

**KANDULA SRINIVASA REDDY MEMORIAL COLLEGE OF ENGINEERING
(AUTONOMOUS)**

KADAPA-516003. AP

(Approved by AICTE, Affiliated to JNTUA, Ananthapuramu, Accredited by NAAC)

(An ISO 9001-2008 Certified Institution)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING



VALUE ADDED COURSE

ON

“.NET TECHNOLOGY”

Resource Person: Dr. K. Srinivasa Rao, Professor, Dept. of AIML, KSRMCE

Course Coordinator: Mr. Sunil J. Assistant Professor, Dept. of AIML, KSRMCE

Duration: 30/09/2023 to 23/10/2023

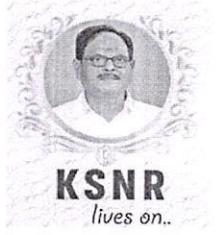


K.S.R.M. COLLEGE OF ENGINEERING
(UGC-AUTONOMOUS)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution



Lr./KSRMCE/AIML/2023-24/

Date:26-09-2023

To
The Principal,
KSRMCE,
Kadapa.

Respected Sir,

Sub: Permission to Conduct Value added Course on “.Net Technology”
30/09/2023 to 23/10/2023–Req- Reg.

The Department of Artificial Intelligence & Machine Learning is planning to offer a Value-Added Course on “.Net Technology” to B. Tech. students. The course will be conducted from 30/09/2023 to 23/10/2023. In this regard, I kindly request you to grant permission to conduct Value Added Course.

Thanking you sir,

*forwarded to
Principal Sir,
KSRMCE
26/9/23*

Yours faithfully

(Mr. Sunil J, Assistant Professor in AIML)

*Permitted
V. S. S. Mm/15*



K.S.R.M. COLLEGE OF ENGINEERING (UGC-AUTONOMOUS)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution



Cr./KSRMCE/AIML/2023-24/

Date:27/09/2023

Circular

The Department of Artificial Intelligence & Machine Learning is offering a Value-Added Course on “.NET TECHNOLOGY” from 30/09/2023 to 23/10/2023 to B. Tech students. In this regard, interested students are requested to register their names for the Value-Added Course with Course Coordinator.

For further information contact Course Coordinator.

Course Coordinator: Mr. Sunil J, Assistant Professor, Dept. of AIML. -KSRMCE.
Contact No: 8978571543

HOD

Dept. of AIML

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

Cc to:

IQAC-KSRMCE

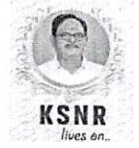


K.S.R.M. COLLEGE OF ENGINEERING (UGC-AUTONOMOUS)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution



Date: 27-09-2023

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

REGISTRATION FORM

Value Added Course

On

“Net Technology” From 30/09/2023 to 23/10/2023

S. No	Roll Number	Full Name	Branch	Semester	Signature
1	229YIA3906	C. Sai Sri Lakshmi	AI&ML	III	
2	229YIA3912	C. Vyshnavi	AI&ML	III	
3	229YIA3919	K. Amaamani	AI&ML	III	
4	229YIA3922	K. Abhila Shma	AI&ML	III	
5	219YIA3909	C. Venkatesh	AI&ML	IV	
6	219YIA3904	Arula Amith	AI&ML	IV	
7	229YIA3915	G. Chandrasekh	AIML	IV	
8	219YIA3931	N. Anjan	AIML	IV	
9	219YIA0509	B. Naveen	CSE	IV	
10	229YIA3903	K. Dhanyush	AIML	IV	
11	219YIA3965	Y. Chenna Keyasa	AIML	IV	
12	229YIA3907	Y. Dileepkasi	AIML	IV	
13	219YIA3901	A. Pradeepkumar	AIML	V	
14	219YIA3902	A. Sharathkumar	AIML	V	
15	219YIA3903	A. Mohammedsufiyan	AIML	V	
16	219YIA3904	A. Amith	AIML	V	
17	219YIA3905	C. Venkatesh	AIML	V	



/ksrmce.ac.in

Follow Us:



/ksrmceofficial

18	21941A3909	Chenna Venkatesh	AI&ML	<u>V</u>	Chenna
19	21941A3910	Dappella Vishnuteja	AI&ML	<u>V</u>	Dappella
20	21941A3911	Deepak Reddy ^{reddy} Srisri	AI&ML	<u>V</u>	Deepak
21	21941A3912	Dada Sathashireddy	AI&ML	<u>V</u>	Dada
22	21941A3919	Kanchasla Ganga ^{Yatisa}	AI&ML	<u>V</u>	Kanchasla
23	21941A0520	B. Chintu	CSE	<u>V</u>	Chintu
24	21941A2052	B. Keerthi	CSE	<u>V</u>	Keerthi
25	21941A3921	Kenqui Umeh	AI&ML	<u>V</u>	Kenqui
26	21941A3922	Krupakaran Karthika	AI&ML	<u>V</u>	Karthika
27	21941A3927	Nayana Arneesh Khan	AI&ML	<u>V</u>	Nayana
28	22941A3928	Meruva Haritha	AI&ML	<u>V</u>	Meruva
29	22941A3929	Mohanlal Junaid ^{Bali}	AI&ML	<u>V</u>	Mohanlal
30	22941A3930	Murthy Meghana	AI&ML	<u>V</u>	Murthy
31	22941A3931	Nagella Arjun	AI&ML	<u>V</u>	Nagella
32	21941A3936	Obureddygaru Manasa	AI&ML	<u>V</u>	Manasa
33	21941A3937	P. LIKITHA	AI&ML	<u>V</u>	Likitha
34	21941A3938	P. Sowanya	AI&ML	<u>V</u>	Sowanya
35	21941A3939	Palleirambai Pravalika	AI&ML	<u>V</u>	Pravalika
36	21941A3940	Pasala Naga Praganka	AI&ML	<u>V</u>	P. Nagaprasanna
37	21941A3949	Shaik Mohammed Abdulmojeed	AIML	<u>V</u>	S.M. Attose
38	21941A3950	Shaik Mohammed Fahemullah	AIML	<u>V</u>	Fahemullah
39	21941A3951	Shaik Mohammed Gaffarali	AIML	<u>V</u>	S.M. Gaffar Ali
40	21941A3952	Shaik Mohammed Saad	AIML	<u>V</u>	S. Saad
41	21941A3961	Vallepu Ajay	AIML	<u>V</u>	V. Ajay
42	21941A3962	Valluvu Vuchitha	AIML	<u>V</u>	V. Vuchitha
43	21941A3963	Vankathar a Jahnavi	AIML	<u>V</u>	V. Jahnavi
44	21941A3964	Vadlani Lakshmi Sauravi	AIML	<u>V</u>	V. Sauravi
45	21941A3965	Venugudipati Divyansha	AIML	<u>V</u>	V. Divyansha



46	229Y5A3905	KOTLO Dhanush	AIML	V	Dhanush
47	229Y5A3905	VADDE Hemathkuma	AIML	V	Hemant
48	229Y5A3906	YERRA Hemant K ^{mn}	AIML	V	Hemant
49	229Y5A3907	YERRAGORLA Dileep ^{Kasi}	AIML	V	Kasi
50	219Y1A0523	Chinngegowda ^{Devendra Prasad}	CSE	V	Devendra
51	219Y1A0527	C. Indra Kumar	CSE	V	Indra
52	219Y1A0528	C. Vasudha	CSE	V	Vasudha
53	219Y1A586	M Raghavul	CSE	V	Raghu
54	219Y1A0509	BALAGANI Naveen	CSE	V	BA
55	219Y1A0510	Bathala Nagaraju	CSE	V	BA
56	219Y1A0511	Bathala Sai Sahaja	CSE	V	SAB
57	219Y1A0520	Chemudu ^{Chandra Sheer} Sathya	CSE	V	CS
58	219Y1A0521	Chimalapenta Gintu	CSE	V	CS
59	219Y1A0522	Chinnannagasi Pullareddy	CSE	V	CS
60	229Y1A3931	Merura Yamini	AIML	V	CS
61	229Y1A3931	Meghal Mohammed Ghans	AIML	V	Meghal


Coordinator



HoD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)



/ksrmce.ac.in

Follow Us:



/ksrmceofficial

Syllabus of Value Added Course

Course Name: .NET TECHNOLOGY

Course Objectives:

1. To learn about basic features of ASP.NET and its controls
2. To create an ASP.NET application using standard .NET Controls
3. To learn about connecting data sources using ADO.NET and managing them.

Course Outcomes:

1. Learners will be able to design web applications using ASP.NET
2. Develop the console and GUI applications using C# .Net.
3. Create the dynamic web page using ASP.NET Controls which interact with databases.
4. Manage cookies and sessions as state management techniques.

UNIT-I

Introduction to .NET & Console Applications:

Introduction to dot net, Dot net overview, .NET framework, MSIL, CLR, CLS, CTS, Namespaces, Assemblies the common language implementation, Assemblies, Garbage collection, Versioning and side-by-side execution, The end to DLL hell, Managed execution.

UNIT-II

C# .NET & Windows Forms and Controls:

C# language fundamentals, Name spaces, Constructor and destructors, Object oriented programming in C# .NET, Modifiers, Property and indexers, Events and delegates, Attributes & reflection API, Debugging and error handling, Files and I/O, String manipulation, Windows forms library, Windows forms and controls, Windows forms properties and events.

UNIT-III

ADO.NET & ASP.NET:

Features of ADO.NET, ADO.NET compared to classic ADO, Datasets, Managed providers, Data binding, Databases and data access using ADO.NET, Data sets and XML, Typed data sets. Introduction to ASP.NET, Difference between ASP and ASP .Net, working with web and HTML controls, Using rich server controls, Configuration overview.

UNIT-IV

Themes and Master pages & Managing State:

Creating a consistent web site, .NET themes: Working with CSS and scene files, Master pages, Working with content place holder and nested master page, Preserving state in web applications, Page-Level state, Using cookies to preserve state, View state, ASP.NET session state, Storing objects in session state.

UNIT-V

Web Services:

XML web services, WSDL, SOAP, UDDI, Creating an XML web service with visual studio, Designing XML web services, publishing and consuming web services, Discovering web services using UDDI.

Advanced concepts .NET:

Introduction to WCF, Hosting and calling WCF services, Introduction to WPF, WPF templates and controls, Silver light.

Text Books/Reference Books:

1. Christian Nagel, "Professional C# .Net", Wrox Publication
2. Matthew Macdonald and Robert Standefer, "ASP.NET Complete Reference", TataMcGrawHills.
3. Vijay Mukhi, "C# The Basics", BPB Publications
4. Nimit Joshi, "Programming ASP.NET MVC 5", C# Corner.

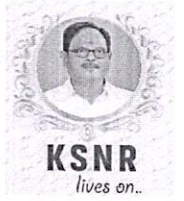


K.S.R.M. COLLEGE OF ENGINEERING (UGC-AUTONOMOUS)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution



SCHEDULE

Department of Artificial Intelligence & Machine Learning

Value Added Course

On

“Net Technology” From 30/09/2023 to 23/10/2023

Date	Timing	Resource Person	Topic to be covered
30/09/2023	9 PM to 12 PM	Dr. K. Srinivasa Rao	Introduction to dot net, Dot net overview, .NET framework, MSIL, CLR
01/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	CLS, CTS, Namespaces, Assemblies the common language implementation, Assemblies
04/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Garbage collection, Versioning and side-by-side execution, The end to DLL hell, Managed execution
05/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	C# language fundamentals, Name spaces, Constructor and destructors, Object oriented programming in C#
06/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	.NET, Modifiers, Property and indexers
06/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Events and delegates, Attributes & reflection API, Debugging and error handling, Files and I/O, String manipulation
07/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Windows forms library, Windows forms and controls, Windows forms properties and events
08/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Features of ADO.NET, ADO.NET compared to classic ADO, Datasets
09/10/2023	9 PM to 12 PM	Dr. K. Srinivasa Rao	Managed providers, Data binding, Databases and data access using ADO.NET, Data sets and XML
11/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Typed data sets. Introduction to ASP.NET, Difference between ASP and ASP .Net, working with web and HTML controls
12/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Using rich server controls, Configuration overview
13/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Creating a consistent web site, .NET themes: Working with CSS and scene

			files, Master pages
14/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Working with content place holder and nested master page, Preserving state in web applications
15/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Page-Level state, Using cookies to preserve state, View state, ASP.NET session state
16/10/2023	9 PM to 12 PM	Dr. K. Srinivasa Rao	XML web services, WSDL, SOAP, UDDI, Creating an XML
18/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	web service with visual studio, Designing XML web services,
19/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Discovering web services using UDDI
20/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Introduction to WCF, Hosting and calling WCF services
21/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	Introduction to WPF
22/10/2023	4 PM to 6 PM	Dr. K. Srinivasa Rao	publishing and consuming web services with an examples
23/10/2023	9 PM to 12 PM	Dr. K. Srinivasa Rao	WPF templates and controls, Silver light.



Resource Person(s)



Coordinator(s)



HoD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

19	219Y1A3937	P LIKITHA (W)	Lik	Lik	Lik	Lik	Lik	Lik	A	A	Lik	Lik	Lik	Lik	Lik	Lik	Lik	Lik	Lik	Lik
20	219Y1A3938	P SOWJANYA (W)	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q	Q
21	219Y1A3939	PALLETI RAM SAI PRAVALLIK A (W)	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro	pro
22	219Y1A3940	PASALA NAGA PRIYANKA (W)	Ri	Ri	Ri	Ri	Ri	Ri	A	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri
23	219Y1A3949	SHAIK MOHAMMED ABDUL MOIZE	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
24	219Y1A3950	SHAIK MOHAMMED FAHEEMULL AH	S	S	A	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
25	219Y1A3951	SHAIK MOHAMMED GAFFAR ALI	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
26	219Y1A3952	SHAIK MOHAMMED SAAD	e	e	e	e	e	e	e	e	e	A	e	e	e	e	e	e	e	e
27	219Y1A3961	VALLEPU AJAY	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj	Aj
28	219Y1A3962	VALLURU RUCHITHA (W)	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
29	219Y1A3963	VANKADHA RA GURU JAHNAVI (W)	a	a	a	a	a	a	a	a	A	A	A	a	a	a	a	a	a	a

53	219Y1A05B7	NADINDLA RAFIUL REHAMAN	A	Reh	Reh	Reh	Reh	Reh	Reh	Reh	A	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	A
54	219Y1A05B8	NADINDLA RIYAZUL REHAMAN	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	Reh	A	Reh	Reh	Reh	Reh
55	219Y1A05B9	NAGELLA NAGA BRAMHAM	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
56	219Y1A05C0	NANDANOO R SASIDHAR	Sasi	Sasi	Sasi	Sasi	A	A	A	A	Sasi	Sasi	Sasi	Sasi	Sasi	Sasi	Sasi	Sasi	Sasi	Sasi	Sasi
57	229Y1A3919	KANTA YAMINI(W)	#	#	A	#	#	#	#	#	#	#	#	#	A	#	#	#	#	#	#
58	229Y1A3921	KODATHALA BHAVANA(W)	Bha	Bha	Bha	Bha	Bha	Bha	Bha	A	Bha	Bha	Bha	Bha	Bha	Bha	Bha	Bha	Bha	Bha	Bha
59	229Y1A3922	KOMALAPA TI ABHILASHA(W)	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh	Abh
60	229Y1A3931	MERUVA YAMINI(W)	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me	Me
61	229Y1A3932	MUGHAL MOHAMMED GHANI BAIG	Mgh	Mgh	Mgh	A	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh	Mgh


Coordinator(s)


HoD



KSRM
COLLEGE OF ENGINEERING

(UGC - Autonomous)
 Kadapa, Andhra Pradesh, India- 516 005
 Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapuramu.



KSNR
lives on..

VALUE ADDED COURSE ON

.NET TECHNOLOGY



Microsoft
.NET



Microsoft
Visual C#



mono



Visual Studio

Microsoft

VB.net



AI&ML



Python Programming Lab
 (AI202)



30.09.2023
 to
 23.10.2023

Resource Person

Dr. K. Srinivasa Rao

Prof, Department of AI&ML

Coordinator

Mr. Sunil J

Assistant professor, Department of AI&ML

Dr. V.S.S. Murthy
 (Principal)

Dr. Kandula Chandra Obul Reddy
 (MD, KCI)

Smt. K.Rajeswari
 (Correspondent, Secretary, Treasurer)

Sri K. Madan Mohan Reddy
 (Vice - Chairman)

Sri K. Raja Mohan Reddy
 (Chairman)

kasmceofficial

www.kasmce.ac.in

8143731980, 8575697569



K.S.R.M. COLLEGE OF ENGINEERING

(UGC-AUTONOMOUS)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution



Report on Value Added Course on “.Net Technology” From 30/09/2023 to 23/10/2023

Target Group	:	B. Tech Students
Details of Participants	:	61 Students
Co-coordinator(s)	:	Mr. Sunil J
Resource Person(s)	:	Dr. K. Srinivasa Rao
Organizing Department	:	Artificial Intelligence & Machine Learning
Venue	:	Python Programming lab(AI-302)

Description:

The Department of Artificial Intelligence & Machine Learning conducted a Value Added Course on “.Net Technology” from 30/09/2023 to 23/10/2023. The course Resource Persons is Sri **Dr. K. Srinivasa Rao**, Professor Department Artificial Intelligence & Machine Learning, KSRMCE.

The main objective of this course is to introduce the fundamental concepts to build desktop, web and mobile applications that can run natively on any operating systems. The .NET system includes tools, libraries, and languages that support modern, scalable, and high-performance software development. An active developer community maintains and supports.

We can write and run .NET apps in C#, F#, or Visual Basic,

- C# is a simple, modern, object-oriented, and type-safe programming language.
- F# is a programming language that makes it easy to write succinct, robust, and performance code.
- Visual Basic is an approachable language with a simple syntax for building type-safe, object-oriented apps.

It supports the following features:

Cross Platform

Whether you're working in C#, F#, or Visual Basic, your code will run natively on any compatible operating system. You can build many types of apps with .NET. Some are cross-platform, and some target a specific set of operating systems and devices.

One consistent API

.NET provides a standard set of base class libraries and APIs that are common to all .NET applications. Each app model can also expose additional APIs that are specific to the operating systems it runs on, or the capabilities it provides. For example, ASP.NET is the cross-platform web framework that provides additional APIs for building web apps that run on Linux or Windows.

Libraries

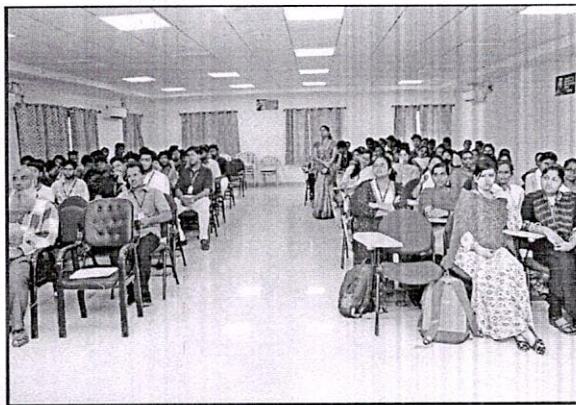
To extend functionality, Microsoft and others maintain a healthy .NET package ecosystem. NuGet is a package manager built specifically for .NET that contains over 100,000 packages.

Application models

- Web Application
- Mobile Applications
- Desktop Applications Micro services
- Cloud Services
- Machine Learning
- Game Development
- Internet of Things

Photos

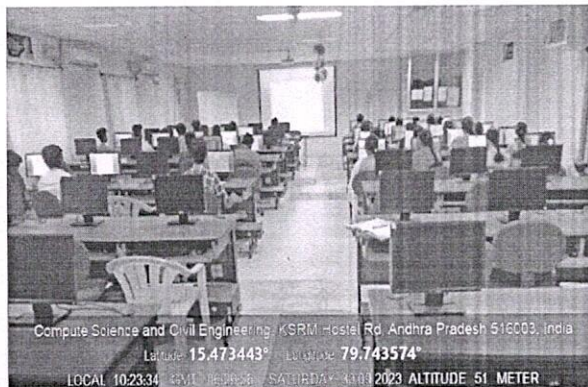
The pictures taken during the course are given below:



Inaugural function of the course



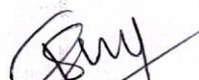
Feedback from the students



Lecture by Resource person Dr. K. Srinivasa Rao




Students while practicing the programs


Coordinator(s)



HoD

Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

 /ksrmce.ac.in

Follow Us:



K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED CERTIFICATE COURSE ON
“.NET TECHNOLOGY” FROM 30/09/2023 to 23/10/2023

ASSESSMENT TEST

Roll Number: _____ **Name of the Student:** _____

Time: 20 Min (Objective Questions) **Max. Marks: 20**

Note: Answer the following Questions and each question carries **one** mark.

1. ASP.NET is a _____. (b)
(a) Client-Side technology (b) Server side Technology (c) Both (d) None
2. Which of the following programming language can be used with ASP.NET? (c)
(a) C#.NET (b) VB.NET (c) Both (d) None
3. Which of the following technologies are also used with ASP.NET? (d)
(a) CSS (b) Java Script (c)AJAX (d) all
4. What is the correct use of the web.config file? (a)
(a) To store global information and variable definitions (b) To configure the domain server
(c) To configure to domain client (d) Store information about web browser
5. Is ASP.NET web form supports an event-driven application model? (a)
(a) True (b) False
6. What namespace is used for ASP.NET Web Form by default? (d)
(a) System.Web.Form (b) System.Web.UI.Page (c) System.Web.GUI.Page (d) System.Web.UI.Form
7. Which term is used for pages that depend on the Master page? (a)
(a) Content Pages (b) Master Pages (c) Web Pages (d) None of the above
8. Which of the following is not an ASP.NET event? (b)
(a) Init (b) Import (c) Load (d) All
9. Which of the following folder is used to store DLL files in the ASP.NET application? (c)
(a) App_Code (b)App_Data (c)Bin (d) App_Local
10. Which of the following method is used to register client script using code dynamically? (a)
(a) Page.ClientScript.RegisterClientScriptBlock
(b)RegisterScript
(c) Page.ClientScript
(d)None

11. Which is the correct basic syntax of Application directive? (a)
- (a) <%@ Application Language="C#" %>
(b) <!-- Application Language="C#" -->
(c) <%@ Application_DIR Language="C#" %>
(d) <%@ DIR @Application Language="C#" %>
12. What is the parent class of all web server control in ASP.Net? (a)
- (a) System.Web.UI.Control (b) System.Web.Control (c) System.Web.UI.WebServer (d) All of the above
13. The Session IDs are stored in ____ by ASP.Net? (d)
- (a) Cache (b) Server (c) Database (d) Cookies
14. Which protocol is used to call a web service? (b)
- (a) SOAP Protocol (b) HTTP Protocol (c) TCP Protocol (d) FTP Protocol
15. What is the difference between session object and application object in ASP.Net? (a)
- (a) The session object is used to maintain the session of each user while an application object is created while a user enters in the application
(b) Session objects are created on the server while application objects are created on the client side.
(c) Session objects are used to handle database communication while application objects are used to handle communication between two different domains. (d) All of the above
16. Which of the following tag is used for the HTML checkbox? (c)
- (a) <check> (b) <checkbox> (c) <input> (d) None of the above
17. What is the correct syntax to create ASP.NET CheckBox control? (b)
- (a) <asp:CB ID="CB1" runat="server" Text="BCA"/>
(b) <asp:Check-Box ID="CB1" runat="server" Text="BCA"/>
(c) <asp_net:CheckBox ID="CB1" runat="server" Text="BCA"/>
(d) <asp:CheckBox ID="CB1" runat="server" Text="BCA"/>
18. Can we apply ToolTip on CheckBox control? (a)
- (a) True (b) False
19. Can we implement an event handler for LinkButton? (a)
- (a) True (b) False
20. Which of the following property is used to set CSS Class to LinkButton control? (b)
- (a) Set Css (b) Css Class (c) Css (d) Class

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED CERTIFICATE COURSE ON
“.NET TECHNOLOGY” FROM 30/09/2023 to 23/10/2023

ASSESSMENT TEST

Roll Number: 219V1A3937 Name of the Student: P. Lileit5a

Time: 20 Min (Objective Questions) **Max. Marks: 20**

Note: Answer the following Questions and each question carries **one** mark.

1. ASP.NET is a _____. (b) ✓
(a) Client-Side technology (b) Server side Technology (c) Both (d) None
2. Which of the following programming language can be used with ASP.NET? (c) ✗
(a) C#.NET (b) VB.NET (c) Both (d) None
3. Which of the following technologies are also used with ASP.NET? (d) ✓
(a) CSS (b) Java Script (c)AJAX (d) all
4. What is the correct use of the web.config file? (a) ✓
(a) To store global information and variable definitions (b) To configure the domain server
(c) To configure to domain client (d) Store information about web browser
5. Is ASP.NET web form supports an event-driven application model? (a) ✓
(a) True (b) False
6. What namespace is used for ASP.NET Web Form by default? (d) ✓
(a) System.Web.Form (b) System.Web.UI.Page (c) System.Web.GUI.Page (d) System.Web.UI.Form
7. Which term is used for pages that depend on the Master page? (a) ✓
(a) Content Pages (b) Master Pages (c) Web Pages (d) None of the above
8. Which of the following is not an ASP.NET event? (b) ✓
(a) Init (b) Import (c) Load (d) All
9. Which of the following folder is used to store DLL files in the ASP.NET application? (c) ✓
(a) App_Code (b)App_Data (c)Bin (d) App_Local
10. Which of the following method is used to register client script using code dynamically? (a) ✓
(a) Page.ClientScript.RegisterClientScriptBlock
(b)RegisterScript
(c) Page.ClientScript
(d)None

19/20
20/20

11. Which is the correct basic syntax of Application directive?

(a) ✓

- (a) <%@ Application Language="C#" %>
- (b) <!-- Application Language="C#" -->
- (c) <%@ Application_DIR Language="C#" %>
- (d) <%@ DIR @Application Language="C#" %>

12. What is the parent class of all web server control in ASP.Net?

(a) ✓

- (a) System.Web.UI.Control (b) System.Web.Control (c) System.Web.UI.WebServer (d) All of the above

13. The Session IDs are stored in ____ by ASP.Net?

(b) ✓

- (a) Cache (b) Server (c) Database (d) Cookies

14. Which protocol is used to call a web service?

(d) ✓

- (a) SOAP Protocol (b) HTTP Protocol (c) TCP Protocol (d) FTP Protocol

15. What is the difference between session object and application object in ASP.Net?

(b) ✓

(a) The session object is used to maintain the session of each user while an application object is created while a user enters in the application

(b) Session objects are created on the server while application objects are created on the client side.

(c) Session objects are used to handle database communication while application objects are used to handle communication between two different domains. (d) All of the above

16. Which of the following tag is used for the HTML checkbox?

(a) ✓

- (a) <check> (b) <checkbox> (c) <input> (d) None of the above

17. What is the correct syntax to create ASP.NET CheckBox control?

(b) ✓

- (a) <asp:CB ID="CB1" runat="server" Text="BCA"/>
- (b) <asp:Check-Box ID="CB1" runat="server" Text="BCA"/>
- (c) <asp_net:CheckBox ID="CB1" runat="server" Text="BCA"/>
- (d) <asp:CheckBox ID="CB1" runat="server" Text="BCA"/>

18. Can we apply ToolTip on CheckBox control?

(a) ✓

- (a) True (b) False

19. Can we implement an event handler for LinkButton?

(a) ✓

- (a) True (b) False

20. Which of the following property is used to set CSS Class to LinkButton control?

(b) ✓

- (a) Set Css (b) Css Class (c) Css (d) Class

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED CERTIFICATE COURSE ON
“.NET TECHNOLOGY” FROM 30/09/2023 to 23/10/2023

ASSESSMENT TEST

Roll Number: 219Y1A3961 **Name of the Student:** V. AJAY

Time: 20 Min **(Objective Questions)** **Max. Marks: 20**

Note: Answer the following Questions and each question carries **one** mark.

1. ASP.NET is a _____. (b) ✓
(a) Client-Side technology (b) Server side Technology (c) Both (d) None
2. Which of the following programming language can be used with ASP.NET? (c) ✓
(a) C#.NET (b) VB.NET (c) Both (d) None
3. Which of the following technologies are also used with ASP.NET? (a) ✗
(a) CSS (b) Java Script (c)AJAX (d) all
4. What is the correct use of the web.config file? (b) ✗
(a) To store global information and variable definitions (b) To configure the domain server
(c) To configure to domain client (d) Store information about web browser
5. Is ASP.NET web form supports an event-driven application model? (a) ✓
(a) True (b) False
6. What namespace is used for ASP.NET Web Form by default? (D) ✓
(a) System.Web.Form (b) System.Web.UI.Page (c) System.Web.GUI.Page (d) System.Web.UI.Form
7. Which term is used for pages that depend on the Master page? (a) ✓
(a) Content Pages (b) Master Pages (c) Web Pages (d) None of the above
8. Which of the following is not an ASP.NET event? (b) ✓
(a) Init (b) Import (c) Load (d) All
9. Which of the following folder is used to store DLL files in the ASP.NET application? (a) ✗
(a) App_Code (b)App_Data (c)Bin (d) App_Local
10. Which of the following method is used to register client script using code dynamically? (a) ✓
(a) Page.ClientScript.RegisterClientScriptBlock
(b)RegisterScript
(c) Page.ClientScript
(d)None

11. Which is the correct basic syntax of Application directive?

(a) x

- (a) <%@ Application Language="C#" %>
- (b) <!-- Application Language="C#" -->
- (c) <%@ Application_DIR Language="C#" %>
- (d) <%@ DIR @Application Language="C#" %>

12. What is the parent class of all web server control in ASP.Net?

(c) x

- (a) System.Web.UI.Control (b) System.Web.Control (c) System.Web.UI.WebServer (d) All of the above

13. The Session IDs are stored in ____ by ASP.Net?

(b) x

- (a) Cache (b) Server (c) Database (d) Cookies

14. Which protocol is used to call a web service?

(a) x

- (a) SOAP Protocol (b) HTTP Protocol (c) TCP Protocol (d) FTP Protocol

15. What is the difference between session object and application object in ASP.Net?

(c) x

(a) The session object is used to maintain the session of each user while an application object is created while a user enters in the application

(b) Session objects are created on the server while application objects are created on the client side.

(c) Session objects are used to handle database communication while application objects are used to handle communication between two different domains. (d) All of the above

16. Which of the following tag is used for the HTML checkbox?

(a) x

- (a) <check> (b) <checkbox> (c) <input> (d) None of the above

17. What is the correct syntax to create ASP.NET CheckBox control?

(b) ✓

- (a) <asp:CB ID="CB1" runat="server" Text="BCA"/>
- (b) <asp:Check-Box ID="CB1" runat="server" Text="BCA"/>
- (c) <asp_net:CheckBox ID="CB1" runat="server" Text="BCA"/>
- (d) <asp:CheckBox ID="CB1" runat="server" Text="BCA"/>

18. Can we apply ToolTip on CheckBox control?

(a) ✓

- (a) True (b) False

19. Can we implement an event handler for LinkButton?

(a) ✓

- (a) True (b) False

20. Which of the following property is used to set CSS Class to LinkButton control?

(b) ✓

- (a) Set Css (b) Css Class (c) Css (d) Class

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED CERTIFICATE COURSE ON
“.NET TECHNOLOGY” FROM 30/09/2023 to 23/10/2023

ASSESSMENT TEST

Roll Number: 21941A0521 Name of the Student: C. Chintu

Time: 20 Min (Objective Questions) **Max. Marks: 20**

Note: Answer the following Questions and each question carries **one** mark.

1. ASP.NET is a _____. (b) ✓
(a) Client-Side technology (b) Server side Technology (c) Both (d) None
2. Which of the following programming language can be used with ASP.NET? (a) ✗
(a) C#.NET (b) VB.NET (c) Both (d) None
3. Which of the following technologies are also used with ASP.NET? (c) ✗
(a) CSS (b) Java Script (c)AJAX (d) all
4. What is the correct use of the web.config file? (a) ✓
(a) To store global information and variable definitions (b) To configure the domain server
(c) To configure to domain client (d) Store information about web browser
5. Is ASP.NET web form supports an event-driven application model? (a) ✓
(a) True (b) False
6. What namespace is used for ASP.NET Web Form by default? (a) ✗
(a) System.Web.Form (b) System.Web.UI.Page (c) System.Web.GUI.Page (d) System.Web.UI.Form
7. Which term is used for pages that depend on the Master page? (b) ✗
(a) Content Pages (b) Master Pages (c) Web Pages (d) None of the above
8. Which of the following is not an ASP.NET event? (c) ✗
(a) Init (b) Import (c) Load (d) All
9. Which of the following folder is used to store DLL files in the ASP.NET application? (d) ✗
(a) App_Code (b)App_Data (c)Bin (d) App_Local
10. Which of the following method is used to register client script using code dynamically? (a) ✓
(a) Page.ClientScript.RegisterClientScriptBlock
(b)RegisterScript
(c) Page.ClientScript
(d)None

11. Which is the correct basic syntax of Application directive?

(a) X

(a) <%@ Application Language="C#" %>

(b) <!-- Application Language="C#" -->

(c) <%@ Application_DIR Language="C#" %>

(d) <%@ DIR @Application Language="C#" %>

12. What is the parent class of all web server control in ASP.Net?

(d) X

(a) System.Web.UI.Control (b) System.Web.Control (c) System.Web.UI.WebServer (d) All of the above

13. The Session IDs are stored in ____ by ASP.Net?

(c) X

(a) Cache (b) Server (c) Database (d) Cookies

14. Which protocol is used to call a web service?

(b) X

(a) SOAP Protocol (b) HTTP Protocol (c) TCP Protocol (d) FTP Protocol

15. What is the difference between session object and application object in ASP.Net?

(a) X

(a) The session object is used to maintain the session of each user while an application object is created while a user enters in the application

(b) Session objects are created on the server while application objects are created on the client side.

(c) Session objects are used to handle database communication while application objects are used to handle communication between two different domains. (d) All of the above

16. Which of the following tag is used for the HTML checkbox?

(c) ✓

(a) <check> (b) <checkbox> (c) <input> (d) None of the above

17. What is the correct syntax to create ASP.NET CheckBox control?

(c) X

(a) <asp:CB ID="CB1" runat="server" Text="BCA"/>

(b) <asp:Check-Box ID="CB1" runat="server" Text="BCA"/>

(c) <asp_net:CheckBox ID="CB1" runat="server" Text="BCA"/>

(d) <asp:CheckBox ID="CB1" runat="server" Text="BCA"/>

18. Can we apply ToolTip on CheckBox control?

(b) X

(a) True (b) False

19. Can we implement an event handler for LinkButton?

(c) X

(a) True (b) False

20. Which of the following property is used to set CSS Class to LinkButton control?

(b) ✓

(a) Set Css (b) Css Class (c) Css (d) Class

K.S.R.M. COLLEGE OF ENGINEERING (AUTONOMOUS), KADAPA-516003
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING
VALUE ADDED COURSE ON
“.NET TECHNOLOGY” FROM 30/09/2023 to 23/10/2023

AWARD LIST

S.NO	Roll Number	Name of the Student	Marks Obtained
1	219Y1A3901	AKULETI PRADEEP KUMAR	16
2	219Y1A3902	ANNEM SHARATH KUMAR REDDY	12
3	219Y1A3903	ATTAR MOHAMMED SUFIYAN	07
4	219Y1A3904	AVULA AMITH	10
5	219Y1A3909	CHENNA VENKATESH	12
6	219Y1A3910	DAPPELLA VISHNU TEJA	14
7	219Y1A3911	DEEPAK REDDY SIRIGIREDDY	04
8	219Y1A3912	DODLA SANTHOSH REDDY	09
9	219Y1A3919	KANCHARLA GANGA YATISH	10
10	219Y1A3920	KAYAPATI KARTHIKEYA BABRUVAHANA	12
11	219Y1A3921	KENGERI UMESH	18
12	219Y1A3922	KRUPAKARAN KARTHIKAN	15
13	219Y1A3927	MAYANA MOHAMMAD AMEER KHAN	16
14	219Y1A3928	MERUVA HARITHA (W)	12
15	219Y1A3929	MOGHAL JUNAID BAIG	17
16	219Y1A3930	MURTHY MEGHANA (W)	12
17	219Y1A3931	NAGELLA ARJUN	12
18	219Y1A3936	OBULAREDDY GARI MANASA (W)	16
19	219Y1A3937	P LIKITHA (W)	19
20	219Y1A3938	P SOWJANYA (W)	12
21	219Y1A3939	PALLETI RAM SAI PRAVALLIKA (W)	18
22	219Y1A3940	PASALA NAGA PRIYANKA (W)	17
23	219Y1A3949	SHAIK MOHAMMED ABDUL MOIZE	10
24	219Y1A3950	SHAIK MOHAMMED FAHEEMULLAH	14
25	219Y1A3951	SHAIK MOHAMMED GAFFAR ALI	12
26	219Y1A3952	SHAIK MOHAMMED SAAD	15
27	219Y1A3961	VALLEPU AJAY	11
28	219Y1A3962	VALLURU RUCHITHA (W)	12
29	219Y1A3963	VANKADHARA GURU JAHNAVI (W)	10
30	219Y1A3964	YADATI LAKSHMISRAVANI (W)	04
31	219Y1A3965	YERUGUDIPADU CHENNA KESAVA	11
32	229Y5A3903	KOTLO DHANUSH	10 10
33	229Y5A3905	VADDE HEMANTHKUMAR	08
34	229Y5A3906	YERRA HEMANTH KUMAR	12
35	229Y5A3907	YERRAGORLA DILEEP KASI	09
36	219Y1A0504	ANNAREDDY BINDUSREE (W)	10
37	219Y1A0505	APKHAN ASIF KHAN	09
38	219Y1A0506	APPUKONDU VENKATA SUBBA REDDY	12
39	219Y1A0507	ARAVETI RUSHIKESAVA REDDY	10
40	219Y1A0509	BALAGANI NAVEEN	16

41	219Y1A0510	BATHALA NAGARAJU	12
42	219Y1A0511	BATTHALA SAI SAHAJA (W)	19
43	219Y1A0512	BHARATHAKAVI KEERTHI (W)	02
44	219Y1A0520	CHEMUDURU SATHYA CHAKRADHAR	11
45	219Y1A0521	CHIMALAPENTA CHINTU	06
46	219Y1A0522	CHINNANNAGARI PULLAREDDY	15
47	219Y1A0523	CHINNEGOWLLA DEVENDRA PRASAD	16
48	219Y1A0527	CHINTHAKUNTA INDRA KUMAR	10
49	219Y1A0528	CHODUBOYINA VASUDHA (W)	07
50	219Y1A0529	CHOWDAPPAGOLLA RAVI	11
51	219Y1A0530	CHUKKASANDEEPAKUMAR	14
52	219Y1A05B6	MUTYALA RAGHAVENDRA	12
53	219Y1A05B7	NADINDLA RAFIUL REHAMAN	14
54	219Y1A05B8	NADINDLA RIYAZUL REHAMAN	10
55	219Y1A05B9	NAGELLA NAGA BRAMHAM	12
56	219Y1A05C0	NANDANOOOR SASIDHAR	12
57	229Y1A3919	KANTA YAMINI(W)	01
58	229Y1A3921	KODATHALA BHAVANA(W)	18
59	229Y1A3922	KOMALAPATI ABHILASHA(W)	10
60	229Y1A3931	MERUVA YAMINI(W)	15
61	229Y1A3932	MUGHAL MOHAMMED GHANI BAIG	12


Coordinator


HoD



K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.



KSNR
lives on..


Certificate of Completion

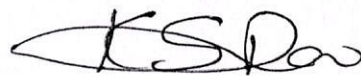
This to certify that Mr/Mrs. v Ajay Bearing


the Roll Number 21941A3961 has Successfully Completed Value Added

Course on ".NET Technology" from 30th September 2023 to 23 October 2023,

Organized by Department of AIML, KSRMCE, Kadapa.


Coordinator


HOD
Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)


Principal



K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.



KSNR
lives on..

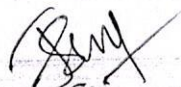
Certificate of Completion

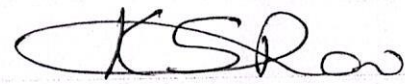
This to certify that Mr/Mrs. C. Sandeep Kumar Bearing

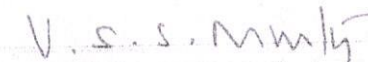
the Roll Number 21941A0530 has Successfully Completed Value Added

Course on ".NET Technology" from 30th September 2023 to 23 October 2023,

Organized by Department of AIML, KSRMCE, Kadapa.


Coordinator


HOD
Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)


Principal



K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India- 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.



KSNR
lives on..

Certificate of Completion


This to certify that Mr/Mrs. K. Umesh Bearing

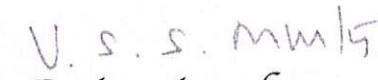
the Roll Number 21941A3921 has Successfully Completed Value Added

Course on ".NET Technology" from 30th September 2023 to 23 October 2023,


Organized by Department of AIML, KSRMCE, Kadapa.


Coordinator


HOD
Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)


Principal

Feedback form on Value Added Course ".NET Technology" from **30/09/2023** to **23/10/2023**

 ksr@ksrmce.ac.in (not shared) Switch account



* Required

Roll Number *

Your answer

Name of the Student *

Your answer

The objectives of the Value Added Course were met (Objective) *

- Excellent
- Good
- Satisfactory
- Poor



The Value Added Course satisfy my expectation as a value added Programme (Course Satisfaction) *

- Excellent
- Satisfactory
- Good
- Poor

Any Issues

Your answer

Submit

Clear form


Never submit passwords through Google Forms.

This form was created inside of KSRM College of Engineering. [Report Abuse](#)

Google Forms



Feedback form on Value Added Course ".NET Technology" from 30/09/2023 to 23/10/2023

 ksr@ksrmce.ac.in (not shared) Switch account



* Required

Roll Number *

Your answer

Name of the Student *

Your answer

The objectives of the Value Added Course were met (Objective) *

- Excellent
- Good
- Satisfactory
- Poor



The content of the course was organized and easy to follow (Delivery) *

- Excellent
- Good
- Satisfactory
- Poor

The Resource Persons were well prepared and able to answer any question (Interaction) *

- Excellent
- Good
- Satisfactory
- Poor

The exercises/role play were helpful and relevant (Syllabus Coverage) *

- Excellent
- Good
- Satisfactory
- Poor



K.S.R.M. COLLEGE OF ENGINEERING

Demartment of Artificial Intelligence & Machine Learning, Feedback report on Value Added Course on .NET Technology

Timestamp	Name	Roll Number	Email	Branch	The objectives of the Value Added Course were met	The content of the course was organized and easy to follow	The Resource Persons were well prepared and able to	The exercises/role play were helpful and relevant (Syllabus)	The Value Added Course satisfy my expectation as a value	Any Issues
10/24/2023 14:12:00	ATTAR MOHAMMA D USMAN	229Y1A3903	229Y1A3903 @ksrmce.ac. in	AIML	Excellent	Good	Satisfactory	Good	Good	No
10/24/2023 14:13:33	BARREMUKA LA ESWAR	229Y1A3904	229Y1A3904 @ksrmce.ac.	AIML	Good	Satisfactory	Good	Excellent	Excellent	
10/24/2023 14:16:23	BOYA POOJITHA(W)	229Y1A3905	229Y1A3905 @ksrmce.ac.	AIML	Excellent	Good	Good	Satisfactory	Good	
10/24/2023 14:16:30	C SAI SRI LAKSHMI(W)	229Y1A3906	229Y1A3906 @ksrmce.ac.	AIML	Good	Good	Excellent	Good	Good	
10/24/2023 14:17:34	ATTAR MOHAMMA D USMAN	229Y1A3903	229Y1A3903 @ksrmce.ac. in	AIML	Satisfactory	Good	Excellent	Excellent	Good	
10/24/2023 14:17:35	BARREMUKA LA ESWAR	229Y1A3904	229Y1A3904 @ksrmce.ac.	AIML	Excellent	Good	Excellent	Good	Good	
10/24/2023 14:17:40	BOYA POOJITHA(W)	229Y1A3905	229Y1A3905 @ksrmce.ac.	AIML	Excellent	Excellent	Good	Satisfactory	Good	
10/24/2023 14:18:20	C SAI SRI LAKSHMI(W)	229Y1A3906	229Y1A3906 @ksrmce.ac.	AIML	Excellent	Satisfactory	Good	Excellent	Satisfactory	
10/24/2023 14:18:22	CHINNA VENKATAGA RI	229Y1A3912	229Y1A3912 @ksrmce.ac. in	AIML	Excellent	Satisfactory	Excellent	Excellent	Satisfactory	
10/24/2023 14:18:27	DEVISETTY BHAVYASREE	229Y1A3913	229Y1A3913 @ksrmce.ac.	AIML	Satisfactory	Good	Satisfactory	Good	Satisfactory	
10/24/2023 15:19:40	GOLLA CHANDRASH EKAR	229Y1A3915	229Y1A3915 @ksrmce.ac. in	AIML	Excellent	Good	Satisfactory	Good	Good	
10/24/2023 15:20:30	CHINNA VENKATAGA RI	229Y1A3912	229Y1A3912 @ksrmce.ac. in	AIML	Good	Good	Good	Excellent	Good	
10/24/2023 15:41:42	DEVISETTY BHAVYASREE	229Y1A3913	229Y1A3913 @ksrmce.ac.	AIML	Good	Good	Satisfactory	Excellent	Excellent	

10/24/2023 16:20:22	GOLLA CHANDRASH EKAR	229Y1A3915	229Y1A3915@ksrmce.ac.in	AIML	Excellent	Good	Satisfactory	Good	Good	
10/24/2023 16:21:44	KANTA YAMINI(W)	229Y1A3919	229Y1A3919@ksrmce.ac.in	AIML	Satisfactory	Satisfactory	Good	Good	Good	
10/24/2023 16:32:23	KATIGANDLA SOWMYA(W)	229Y1A3920	229Y1A3920@ksrmce.ac.in	AIML	Satisfactory	Excellent	Good	Excellent	Good	
10/24/2023 16:32:46	KODATHALA BHAVANA(W)	229Y1A3921	229Y1A3921@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Satisfactory	Good	
10/24/2023 16:40:20	KOMALAPAT I ABHILASHA	229Y1A3922	229Y1A3922@ksrmce.ac.in	AIML	Good	Satisfactory	Good	Good	Good	
10/24/2023 16:44:32	KORAPALA ABHINAYA SRI(W)	229Y1A3924	229Y1A3924@ksrmce.ac.in	AIML	Good	Good	Excellent	Excellent	Excellent	
10/24/2023 16:46:49	LAKKINENI NIRANJAN RAJU	229Y1A3925	229Y1A3925@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Excellent	Excellent	
10/24/2023 17:41:74	KANTA YAMINI(W)	229Y1A3919	229Y1A3919@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Excellent	Excellent	
10/24/2023 17:43:51	KATIGANDLA SOWMYA(W)	229Y1A3920	229Y1A3920@ksrmce.ac.in	AIML	Excellent	Good	Good	Good	Satisfactory	
10/24/2023 17:41:52	KODATHALA BHAVANA(W)	229Y1A3921	229Y1A3921@ksrmce.ac.in	AIML	Excellent	Good	Good	Excellent	Satisfactory	
10/24/2023 17:43:35	KOMALAPAT I ABHILASHA	229Y1A3922	229Y1A3922@ksrmce.ac.in	AIML	Excellent	Good	Satisfactory	Excellent	Good	
10/24/2023 17:51:54	KORAPALA ABHINAYA SRI(W)	229Y1A3924	229Y1A3924@ksrmce.ac.in	AIML	Excellent	Satisfactory	Good		Good	
10/24/2023 18:41:55	RUDRA SATHVIKA(W)	229Y1A3944	229Y1A3944@ksrmce.ac.in	AIML	Good	Good	Good	Good	Good	
10/24/2023 18:41:55	SANDIBOINA YASASWANI	229Y1A3945	229Y1A3945@ksrmce.ac.in	AIML	Excellent	Excellent	Good	Good	Good	
10/24/2023 18:41:55	SESHUGAND LA RAJESH	229Y1A3946	229Y1A3946@ksrmce.ac.in	AIML	Good	Excellent		Excellent	Excellent	
10/24/2023 18:41:58	RUDRA SATHVIKA(W)	229Y1A3944	229Y1A3944@ksrmce.ac.in	AIML	Good	Satisfactory	Good	Good	Good	

10/24/2023 18:44:59	SHAIK MOHAMME D ZAHEER	229Y1A3953	229Y1A3953@ksrmce.ac.in	AIML	Good	Good	Good	Satisfactory	Good	
10/24/2023 18:44:59	SHAIK SAFIYA	229Y1A3954	229Y1A3954@ksrmce.ac.in	AIML	Good	Good	Satisfactory	Good	Satisfactory	
10/24/2023 18:45:30	SHAIK SHAHEENA SULTANA(W)	229Y1A3955	229Y1A3955@ksrmce.ac.in	AIML	Excellent	Excellent	Excellent	Satisfactory	Excellent	
10/24/2023 18:45:42	SIDDU ABBAIAGARI GURU MOHAN	229Y1A3956	229Y1A3956@ksrmce.ac.in	AIML	Good	Excellent	Excellent	Good	Good	
10/24/2023 18:46:20	SIRIGI REDDY NANDINI(W)	229Y1A3957	229Y1A3957@ksrmce.ac.in	AIML	Good	Good	Good	Satisfactory	Good	
10/24/2023 18:46:21	SHAIK MOHAMME D ZAHEER	229Y1A3953	229Y1A3953@ksrmce.ac.in	AIML	Good	Good	Excellent	Good	Excellent	
10/24/2023 18:46:21	VALLEPU AKHILA(W)	229Y1A3961	229Y1A3961@ksrmce.ac.in	AIML	Good	Good	Good	Good	Excellent	
10/24/2023 18:47:34	VELLATUR SWETHA(W)	229Y1A3962	229Y1A3962@ksrmce.ac.in	AIML	Excellent	Good	Good	Satisfactory	Excellent	
10/24/2023 18:47:37	VENNAPUSA GOWTHAMI	229Y1A3963	229Y1A3963@ksrmce.ac.in	AIML	Good	Good	Excellent	Good	Good	
10/24/2023 18:47:42	VALLEPU AKHILA(W)	229Y1A3961	229Y1A3961@ksrmce.ac.in	AIML	Good	Good	Excellent	Satisfactory	Excellent	
10/24/2023 18:47:50	VELLATUR SWETHA(W)	229Y1A3962	229Y1A3962@ksrmce.ac.in	AIML	Excellent	Satisfactory	Excellent	Satisfactory	Excellent	
10/24/2023 19:32:50	AKULETI PRADEEP KUMAR	219Y1A3901	219Y1A3901@ksrmce.ac.in	AIML	Good	Satisfactory	Excellent	Satisfactory	Good	
10/24/2023 19:40:21	ANNEM SHARATH KUMAR	219Y1A3902	219Y1A3902@ksrmce.ac.in	AIML	Excellent	Excellent	Good	Good	Excellent	
10/24/2023 19:50:50	ATTAR MOHAMME D SUFIYAN	219Y1A3903	219Y1A3903@ksrmce.ac.in	AIML	Good	Good	Good	Excellent	Good	
10/24/2023 19:54:21	AVULA AMITH	219Y1A3904	219Y1A3904@ksrmce.ac.in	AIML	Good	Good	Good	Good	Good	

10/24/2023 19:60:74	BHUMIREDDY VENKATA RAVI KUMAR	219Y1A3905	219Y1A3905@ksrmce.ac.in	AIML	Excellent	Good	Excellent	Excellent	Good	
10/24/2023 19:61:75	AKULETI PRADEEP KUMAR	219Y1A3901	219Y1A3901@ksrmce.ac.in	AIML	Satisfactory	Excellent	Excellent	Satisfactory	Good	
10/24/2023 19:61:75	ANNEM SHARATH KUMAR	219Y1A3902	219Y1A3902@ksrmce.ac.in	AIML	Excellent	Excellent	Excellent	Excellent	Excellent	
10/24/2023 19:61:75	ATTAR MOHAMMED SUFIYAN	219Y1A3903	219Y1A3903@ksrmce.ac.in	AIML	Excellent	Good	Good	Good	Satisfactory	
10/24/2023 20:01:21	CHENNA VENKATESH	219Y1A3909	219Y1A3909@ksrmce.ac.in	AIML	Excellent	Satisfactory	Good	Good	Good	
10/24/2023 20:01:21	DAPPELLA VISHNU TEJA	219Y1A3910	219Y1A3910@ksrmce.ac.in	AIML	Excellent	Good	Excellent	Good	Excellent	
10/25/2023 14:12:00	DEEPAK REDDY	219Y1A3911	219Y1A3911@ksrmce.ac.in		Excellent	Good	Good	Good	Satisfactory	
10/25/2023 14:13:33	DODLA SANTHOSH REDDY	219Y1A3912	219Y1A3912@ksrmce.ac.in		Excellent	Satisfactory	Good	Good	Good	
10/25/2023 14:16:23	CHENNA VENKATESH	219Y1A3909	219Y1A3909@ksrmce.ac.in		Excellent	Good	Excellent	Good	Excellent	
10/25/2023 14:16:30	KAYAPATI KARTHIKEYA BABRUVAHANA	219Y1A3920	219Y1A3920@ksrmce.ac.in		Excellent	Good	Good	Good	Satisfactory	
10/25/2023 14:17:34	KENGERI UMESH	219Y1A3921	219Y1A3921@ksrmce.ac.in		Excellent	Satisfactory	Good	Good	Good	
10/25/2023 14:17:35	KRUPAKARAN	219Y1A3922	219Y1A3922@ksrmce.ac.in		Excellent	Good	Excellent	Good	Excellent	
10/25/2023 14:17:40	KURUVA MADHUKRISHNA	219Y1A3923	219Y1A3923@ksrmce.ac.in		Good	Good	Good	Excellent	Good	
10/25/2023 14:18:20	MALIREDDY SAI CHARAN REDDY	219Y1A3924	219Y1A3924@ksrmce.ac.in		Good	Good	Good	Good	Good	

10/25/2023 14:18:22	SHAIK SHUAIB	219Y1A3953	219Y1A3953 @ksrmce.ac.		Excellent	Good	Excellent	Excellent	Good	
10/25/2023 14:18:27	SIRIGIREDDY REDDAIAH	219Y1A3954	219Y1A3954 @ksrmce.ac.		Satisfactory	Excellent	Excellent	Satisfactory	Good	
10/25/2023 14:18:28	KOTLO DHANUSH	229Y5A3903	@ksrmce.ac. in		Excellent	Excellent	Excellent	Excellent	Excellent	

Spem
Co-ordinator

KSRao
HOD
Dr. K. SRINIVASA RAO, M.Tech., Ph.D.
Professor & HOD AIML
K.S.R.M. College of Engineering
(Autonomous)
KADAPA- 516 005. (A.P.)

Unit-1

Introduction to .Net framework

1.1 Introduction to Dot Net Framework

The .NET is the technology from Microsoft, on which all other Microsoft technologies will be depending on in future.

It is a major technology change. Just like the computer world moved from DOS to Windows, now they are moving to .NET. But don't be surprised if you find anyone saying that "I do not like .NET and I would stick with the good old COM and C++". There are still lot of people who like to use the bullock-cart instead of the latest Honda car.

.NET technology was introduced by Microsoft, to catch the market from the SUN's Java. Few years back, Microsoft had only VC++ and VB to compete with Java, but Java was catching the market very fast. With the world depending more and more on the Internet/Web and java related tools becoming the best choice for the web applications, Microsoft seemed to be loosing the battle. Thousands of programmers moved to java from VC++ and VB. To recover the market, Microsoft announced .NET.

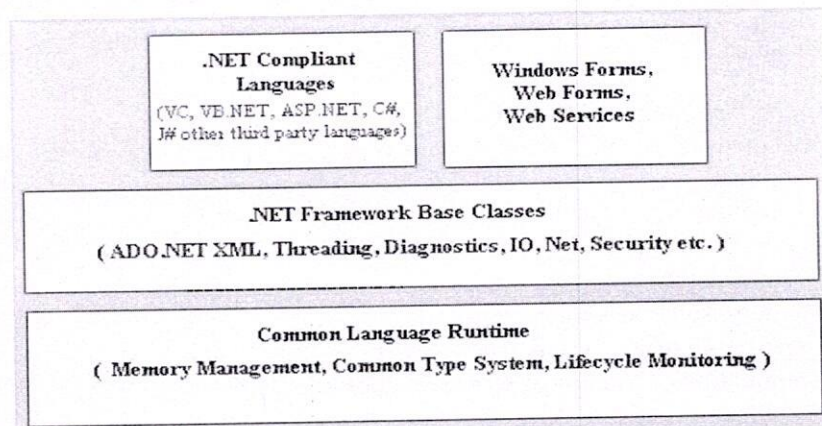
.NET framework comes with a single class library. And thats all programmers need to learn!! Whether they write the code in C# or VB.NET or J#, it doesn't matter, you just use the .NET class library. There is no classes specific to any language. There is nothing more you can do in a language, which you can't do in any other .NET language. You can write code in C# or VB.NET with the same number of lines of code, same performance and same efficiency, because everyone uses same .NET class library.

Features of .NET

- It is a platform neutral framework.
- It is a layer between the operating system and the programming language.
- It supports many programming languages, including VB.NET, C# etc.
- .NET provides a common set of class libraries, which can be accessed from any .NET based programming language. There will not be separate set of classes and libraries for each language. If you know any one .NET language, you can write code in any .NET language.
- In future versions of Windows, .NET will be freely distributed as part of operating system and users will never have to install .NET separately.

Major Components of .NET

The diagram given below describes various components of .NET Framework.



The .NET framework can only be exploited by languages that are compliant with .NET. Most of Microsoft languages have been made to fully comply with .NET.

.NET also introduces Web Forms, Web Services and Windows Forms. The reason why they have been shown separately and not as a part of a particular language is that these technologies can be used by any .NET compliant language. For example Windows Forms is used by VC, VB.NET, C# all as a mode of providing GUI.

The next component of .NET is the .NET Framework Base Classes. These are the common class libraries (much like Java packages) that can be used by any .NET compliant language. These classes provide the programmers with a high degree of functionality that they can use in their programs. For example there are classes to handle reading, writing and manipulating XML documents, enhanced ADOs etc.

The bottom most layer is the CLR - the common runtime language.

Origin of .net Technology

1. OLE Technology
2. COM Technology
3. .net Technology

OLE Technology(Object Linking and Embedding)

- Easy interprocess communication
- Embed documents from one application into another application
- To enable one application to manipulate objects located in another application
- Ex: interoperability between various products such as MS word and MSExcel

COM Technology(Component Object Model)

- Monolithic approach leads to many problem of maintainability and testing
- A program is broken into number of independent components where each one offers a particular service
- Each component can be developed and tested independently and then integrated into main system.
- Benefits:
 - Reduces the overall complexity of software.
 - Enables distributed development across multiple organization or departments.
 - Enhances software maintainability

.net Technology

- Third generation component model
- IPC in COM is replaced by Intermediate Language(IL or MSIL)
- Interoperability by compiling code into IL.
- Metadata

1.2 Common Language Runtime (CLR)

The CLR is the heart of .NET framework. It is .NET equivalent of Java Virtual Machine (JVM). It is the runtime that converts a MSIL (Micro Soft Intermediate Language) code into the host machine language code, which is then executed appropriately.

The CLR provides a number of services that include:

- Loading and execution of codes
- Memory isolation for application
- Verification of type safety
- Compilation of IL into native executable code
- Providing metadata
- Automatic garbage collection
- Enforcement of Security

- Interoperability with other systems
- Managing exceptions and errors
- Provide support for debugging and profiling

1.3 Common Type System (CTS)

The language interoperability, and .NET Class Framework, are not possible without all the language sharing the same data types. What this means is that an "int" should mean the same in VB, VC++, C# and all other .NET compliant languages. Same idea follows for all the other data types. This is achieved through introduction of Common Type System (CTS).

CTS, much like Java, defines every data type as a Class. Every .NET compliant language must stick to this definition. Since CTS defines every data type as a class; this means that only Object-Oriented (or Object-Based) languages can achieve .NET compliance. Given below is a list of CTS supported data types:

Data Type	Description
System.Byte	1-byte unsigned integer between 0-255
System.Int16	2-bytes signed integer in the following range: 32,678 to 32,767
System.Int32	4-byte signed integer containing a value in the following range: -2,147,483,648 to 2,147,483,647
System.Int64	8-byte signed integer containing a value from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
System.Single	4-byte floating point. The value limits are: for negative values: -3.402823E38 to -1.401298E-45 for positive values: 1.401298E-45 TO 30402823E38
System.Double	8-bytes wide floating point. The value limits are: for negative values: -1.79769313486231E308 to -4.964065645841247E-324 for positive values: 4.964065645841247E-324 to 1.79769313486232E308
System.Object	4-bytes address reference to an object
System.Char	2-bytes single Unicode Character.
System.String	string of up to 2 billion Unicode characters.
System.Decimal	12-bytes signed integer that can have 28 digits on either side of decimal.
System.Boolean	4-Bytes number that contains true(1) or false (0)

1.4 Common Language Specification (CLS)

One of the obvious themes of .NET is unification and interoperability between various programming languages. In order to achieve this; certain rules must be laid and all the languages

must follow these rules. In other words we can not have languages running around creating their own extensions and their own fancy new data types. CLS is the collection of the rules and constraints that every language (that seeks to achieve .NET compatibility) must follow. Microsoft has defined three level of CLS compatibility/compliance. The goals and objectives of each compliance level have been set aside. The three compliance levels with their brief description are given below:

Compliant producer

The component developed in this type of language can be used by any other language.

Consumer

The language in this category can use classes produced in any other language. In simple words this means that the language can instantiate classes developed in other language. This is similar to how COM components can be instantiated by your ASP code.

Extender

Languages in this category can not just use the classes as in CONSUMER category; but can also extend classes using inheritance.

Languages that come with Microsoft Visual Studio namely Visual C++, Visual Basic and C#; all satisfy the above three categories. Vendors can select any of the above categories as the targeted compliance level(s) for their languages.

1.5 Microsoft Intermediate Language (MSIL)

A .NET programming language (C#, VB.NET, J# etc.) does not compile into executable code; instead it compiles into an intermediate code called Microsoft Intermediate Language (MSIL). As a programmer one need not worry about the syntax of MSIL - since our source code is automatically converted to MSIL. The MSIL code is then sent to the CLR (Common Language Runtime) that converts the code to machine language which is then run on the host machine.

MSIL is similar to Java Byte code. A Java program is compiled into Java Byte code (the .class file) by a Java compiler, the class file is then sent to JVM which converts it into the host machine language.

Managed Code

The role of CLR doesn't end once we have compiled our code to MSIL and a JIT compiler has compiled this to native code. Code written using the .NET framework, is managed code when it is executed. This stage is usually referred to as being at runtime. This means that the CLR looks after our applications, by managing memory, handling security, allowing cross language debugging and so on. By contrast, applications that do not run under the control of the CLR are said to be unmanaged and certain languages such as C++ can be used to write such applications, that for example, to access low level functions of the operating systems. However in C# we can only write code that runs in a managed environment.

Unified classes

The term .NET framework refers to the group of technologies that form the development foundation for the Microsoft .NET platform. The key technologies in this group are the run time and the class libraries.

The run time is responsible for managing your code and providing services to it while it executes, playing a role similar to that of the Visual Basic 6.0 run time.

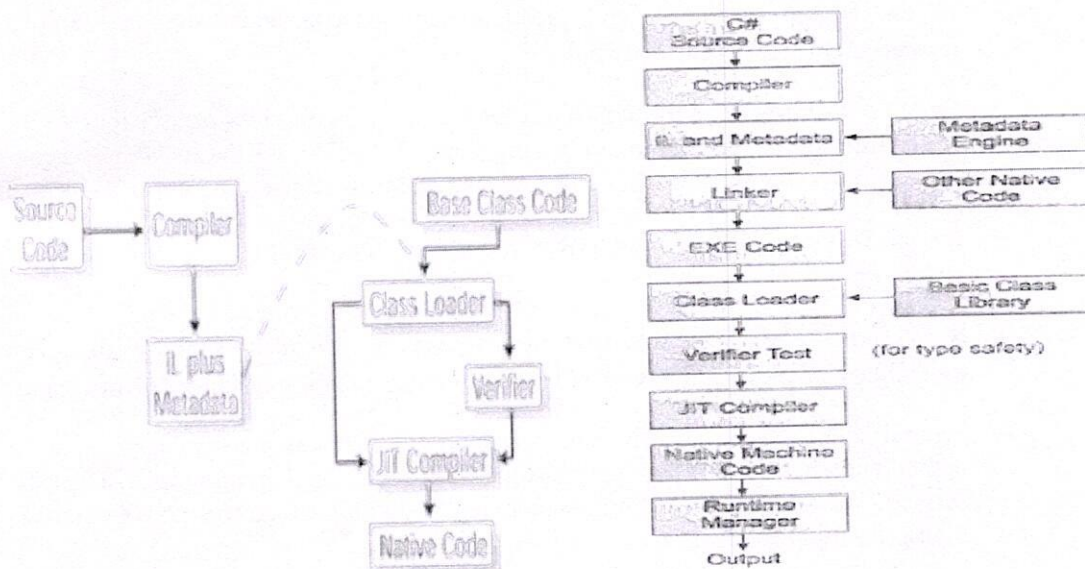
The .NET programming languages including Visual Basic .NET, Microsoft Visual C# and C++ managed extensions and many other programming languages from various vendors utilize .NET services and features through a common set of unified classes.

The .NET unified classes provide foundation of which you build your applications, regardless of the language you use. Whether you simply concating a string, or building a windows Services or a multiple-tier web-based applications, you will be using these unified classes.

The unified classes provide a consistent method of accessing the platform's functionality. Once you learn to use the class library, you'll find that all tasks follow the same uniform architecture, you no longer need to learn and master different API architecture to write your applications. By building your applications on a unified, integrated framework, you maximize your return on the time you spend learning this framework, and you end up with more robust applications that are easy to deploy and maintain.

1.6 Just In Time Compiler

- To make it easy for language writers to port their languages to .NET, Microsoft developed a language akin to assembly language called Microsoft intermediate language (MSIL). To compile applications for .NET, compilers take source code as input and produce MSIL as output.
- MSIL itself is a complete language that you can write applications in. However, as with assembly language, you would probably never do so except in unusual circumstances. Because MSIL is its own language, each compiler team makes its own decision about how much of the MSIL it will support. However, if you're a compiler writer and you want to create a language that does interoperate with other languages, you should restrict yourself to features specified by the CLS.



- You write source code in C# and compile it using the C# compiler (csc.exe) into an EXE.
- The C# compiler outputs the MSIL code and a manifest into a read-only part of the EXE that has a **standard PE (Win32-portable executable) header**. When the compiler creates the output, it also imports a function named **_CorExeMain** from the .NET runtime.
- When the application is executed, the operating system loads the PE, as well as any dependent dynamic-link libraries (DLLs), such as the one that exports the **_CorExeMain** function (mscorlib.dll), just as it does with any valid PE.

1.7 Framework Base Classes

The .NET Framework has an extensive set of class libraries. This includes classes for:

- **Data Access:** High Performance data access classes for connecting to SQL Server or any other OLEDB provider.
- **XML Supports:** Next generation XML support that goes far beyond the functionality of MSXML.
- **Directory Services:** Support for accessing Active Directory/LDPA using ADSI.
- **Regular Expression :** Support for above and beyond that found in Perl 5.
- **Queuing Supports:** Provides a clean object-oriented set of classes for working with MSMQ.

These class libraries use the CLR base class libraries for common functionality.

Base Class Libraries

The Base class library in the .NET Framework is huge. It covers areas such as:

- **Collection :** The System.Collections namespace provides numerous collection classes.
- **Thread Support:** The System.Threading namespace provides support for creating fast, efficient, multi-threaded application.
- **Code Generation:** The System.CodeDOM namespace provides classes for generating source files in numerous language. ASP.NET uses these classes when converting ASP.NET pages into classes, which are subsequently compiled.
- **IO:** The System.IO provides extensive support for working with files and all other stream types.
- **Reflection:** The System.Reflection namespace provides support for load assemblies, examining the type within assemblies, creating instances of types, etc.
- **Security:** The System.Security namespace provides support for services such as authentication, authorization, permission sets, policies, and cryptography. These base services are used by application development technologies like ASP.NET to build their security infrastructure.

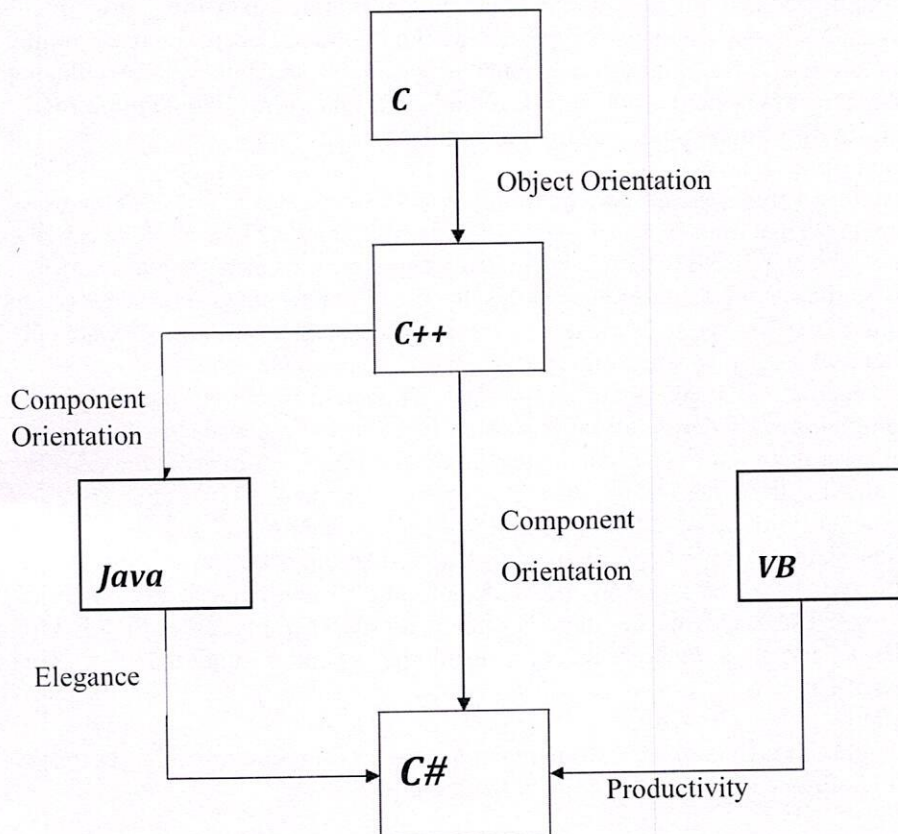
The list of support base classes goes on forever in .NET, but if you ever find yourself lost looking for a specific class, you can use the WinCV tool to locate it. You can run this from the Start bar Run menu. The file is typically located in the c:\program files\Microsoft.Net\FrameworkSDK\Bin directory.

Unit-II C-Sharp Language (C#)

2.1 Introduction

Microsoft Corporation, developed a new computer programming language C# pronounced as 'C-Sharp'. C# is a simple, modern, object oriented, and type safe programming language derived from C and C++. C# is a purely object-oriented language like as Java. It has been designed to support the key features of .NET framework.

Like Java, C# is a descendant language of C++ which is descendant of C language.



C# modernize C++ by enhancing some of its features and adding a few new features. C# borrows Java's features such as grouping of classes, interface and implementation together in one file so the programmers can easily edit the codes. C# also handles objects using reference, the same way as Java.

C# uses VB's approach to form designing, namely, dragging controls from a tool box, dropping them onto forms, and writing events handlers for them.

Comparing C# to C++ and Java

C# versus Java

C# and Java are both new-generation languages descended from a line including C and C++. Each includes advanced features, like garbage collection, which remove some of the low level maintenance tasks from the programmer. In a lot of areas they are syntactically similar.

Both C# and Java compile initially to an intermediate language: C# to Microsoft Intermediate Language (MSIL), and Java to Java bytecode. In each case the intermediate language can be run -

by interpretation or just-in-time compilation - on an appropriate 'virtual machine'. In C#, however, more support is given for the further compilation of the intermediate language code into native code.

C# contains more primitive data types than Java, and also allows more extension to the value types. For example, C# supports 'enumerations', type-safe value types which are limited to a defined set of constant variables, and 'structs', which are user-defined value types.

Unlike Java, C# has the useful feature that we can overload various operators.

Like Java, C# gives up on multiple class inheritance in favour of a single inheritance model extended by the multiple inheritance of interfaces. However, polymorphism is handled in a more complicated fashion, with derived class methods either 'overriding' or 'hiding' super class methods. C# also uses 'delegates' - type-safe method pointers. These are used to implement event-handling. In Java, multi-dimensional arrays are implemented solely with single-dimensional arrays (where arrays can be members of other arrays). In addition to jagged arrays, however, C# also implements genuine rectangular arrays.

C# versus C++

Although it has some elements derived from Visual Basic and Java, C++ is C#'s closest relative. In an important change from C++, C# code does not require header files. All code is written inline. As touched on above, the .NET runtime in which C# runs performs memory management, taking care of tasks like garbage collection. Because of this, the use of pointers in C# is much less important than in C++. Pointers can be used in C#, where the code is marked as 'unsafe', but they are only really useful in situations where performance gains are at an absolute premium.

Speaking generally, the 'plumbing' of C# types is different from that of C++ types, with all C# types being ultimately derived from the 'object' type. There are also specific differences in the way that certain common types can be used. For instance, C# arrays are bounds checked unlike in C++, and it is therefore not possible to write past the end of a C# array.

C# statements are quite similar to C++ statements. To note just one example of a difference: the 'switch' statements has been changed so that 'fall-through' behavior is disallowed.

As mentioned above, C# gives up on the idea of multiple class inheritance. Other differences relating to the use of classes are: there is support for class 'properties' of the kind found in Visual Basic, and class methods are called using the . operator rather than the :: operator.

Features of C#

1. Simplicity

All the Syntax of java is like C++. There is no preprocessor, and much larger library. C# code does not require header files. All code is written inline.

2. Consistent behavior

C# introduced an unified type system which eliminates the problem of varying ranges of integer types. All types are treated as objects and developers can extend the type system simply and easily.

3. Modern programming language

C# supports number of modern features, such as:

- Automatic Garbage Collection
- Error Handling features
- Modern debugging features
- Robust Security features

4. Pure Object- Oriented programming language

In C#, every thing is an object. There are no more global functions, variable and constants. It supports all three object oriented features:

- Encapsulation

- Inheritance
- Polymorphism

5. Type Safety

Type safety promotes robust programming. Some examples of type safety are:

- All objects and arrays are initialized by zero dynamically
- An error message will be produced, on use of any uninitialized variable
- Automatic checking of array out of bound and etc.

6. Feature of Versioning

Making new versions of software module work with the existing applications is known as versioning. Its achieved by the keywords **new** and **override**.

7. Compatible with other language

C# enforces the .NET common language specifications (CLS) and therefore allows inter-operation with other .NET language.

8. Inter-operability

C# provides support for using COM objects, no matter what language was used to author them. C# also supports a special feature that enables a program to call out any native API.

A Simple C# Program

Let's begin in the traditional way, by looking at the code of a Hello World program (note that the tabulation and line numbers are included just for the sake of readability).

```

1.  Using System;
2.  public class HelloWorld
3.  {
4.      public static void Main()
5.      {
6.          // This is a single line comment
7.          /* This is a
8.             multiple
9.             line comment */
10.         Console.WriteLine("Hello World! ");
11.     }
12. }
```

- The first thing to note about C# is that it is case-sensitive. You will therefore get compiler errors if, for instance, you write 'console' rather than 'Console'.
- The second thing to note is that every statement finishes with a semicolon (;) or else takes a code block within curly braces.

Explanation of Program

Line 1 : using System;

we are using the System namespace (namespaces are also covered in chapter 7). The point of this declaration is mostly to save ourselves time typing. Because the 'Console' object used in line 10 of the code actually belongs to the 'System' namespace, its fully qualified name is 'System.Console'. However, because in line 1 we declare that the code is using the System namespace, we can then leave off the 'System.' part of its name within the code.

Line 2: public class HelloWorld

As C# is an object-oriented language, C# programs must be placed in classes (classes are discussed in chapter 5 but if you are new to object orientation we suggest that you first read some introductory material). This line declares the class to be named 'HelloWorld'.

Line 4: public static void Main()

When compiled and run, the program above will automatically run the 'Main' method declared and begun in this line. Note again C#'s case-sensitivity - the method is 'Main' rather than 'main'.

Line 3,11 and 5,12 :

These lines are uses the '{' for starting braces and '}' for closing braces of block. **Lines 6-9 :** Comments

('//' uses for single line and '/* --- */' uses for multiple line comments)

These lines of the program are ignored by the compiler, being comments entered by the programmer for his own benefit.

Line 6 shows a single line comment, in which everything on the line after the two forward slashes is ignored by the compiler.

Lines 7-9 demonstrate a multi-line comment, in which everything between the opening /* and closing */ is ignored, even when it spans multiple lines.

Line 10:

The statement on this line calls the 'WriteLine' method of the Console class in the System namespace. It should be obvious how this works in the given example - it just prints out the given string to the 'Console' (on PC machines this will be a DOS prompt).

Instruction for Saving the Program

In order to run the program, it must first be saved in a file. Unlike in Java, the name of the class and the name of the file in which it is saved do not need to match up, although it does make things easier if you use this convention. In addition, you are free to choose any extension for the file, but it is usual to use the extension '.cs'.

Writing program in Computer

There are two ways of program writing in
computer • Using Text Editor
Using Visual Studio.NET

2.2 Data Types. Identifiers, Variables, Constants and Literals

Identifiers & Variables

Identifiers refer to the names of variables, functions arrays, classes, etc. created by programmer. They are fundamental requirement of any language. Each language has its own rules for naming these identifiers.

To name the variables of your program, you must follow strict rules. In fact, everything else in your program must have a name.

There are some rules you must follow when naming your objects. On this site, here are the rules we will follow:

- The name must start with a letter or an underscore
- After the first letter or underscore, the name can have letters, digits, and/or underscores
- The name must not have any special characters other than the underscore
- The name cannot have a space

C# is case-sensitive. This means that the names Case, case, and CASE are completely different. For example, the main function is always written Main.

C# Keywords

C# uses a series of words, called keywords, for its internal use. This means that you must avoid naming your objects using one of these keywords. They are:

<u>abstract</u>	<u>const</u>	<u>extern</u>	<u>int</u>	<u>out</u>	<u>short</u>	<u>typeof</u>
<u>as</u>	<u>continue</u>	<u>false</u>	<u>interface</u>	<u>override</u>	<u>sizeof</u>	<u>uint</u>
<u>base</u>	<u>decimal</u>	<u>finally</u>	<u>internal</u>	<u>params</u>	<u>stackalloc</u>	<u>ulong</u>
<u>bool</u>	<u>default</u>	<u>fixed</u>	<u>is</u>	<u>private</u>	<u>static</u>	<u>unchecked</u>
<u>break</u>	<u>delegate</u>	<u>float</u>	<u>lock</u>	<u>protected</u>	<u>string</u>	<u>unsafe</u>
<u>byte</u>	<u>do</u>	<u>for</u>	<u>long</u>	<u>public</u>	<u>struct</u>	<u>ushort</u>
<u>case</u>	<u>double</u>	<u>foreach</u>	<u>namespace</u>	<u>readonly</u>	<u>switch</u>	<u>using</u>
<u>catch</u>	<u>else</u>	<u>goto</u>	<u>new</u>	<u>ref</u>	<u>this</u>	<u>virtual</u>
<u>char</u>	<u>enum</u>	<u>if</u>	<u>null</u>	<u>return</u>	<u>throw</u>	<u>void</u>
<u>checked</u>	<u>event</u>	<u>implicit</u>	<u>object</u>	<u>sbyte</u>	<u>true</u>	<u>volatile</u>
<u>class</u>	<u>explicit</u>	<u>in</u>	<u>operator</u>	<u>scaled</u>	<u>try</u>	<u>while</u>

Data types

C# is a type-safe language. Variables are declared as being of a particular type, and each variable is constrained to hold only values of its declared type.

Variables can hold either value types or reference types, or they can be pointers. Here's a quick recap of the difference between value types and reference types.

- where a variable v contains a value type, it directly contains an object with some value. No other variable v' can directly contain the object contained by v (although v' might contain an object with the same value).

- where a variable v contains a reference type, what it directly contains is something which refers to an object. Another variable v' can contain a reference to the same object referred to by v.

Value Types

C# defines the following value types:

- Primitives int i;
- Enum enum state { off, on }
- Struct struct Point { int x, y; }

It is possible in C# to define your own value types by declaring enumerations or structs. These user-defined types are mostly treated in exactly the same way as C#'s predefined value types, although compilers are optimized for the latter. The following table lists, and gives information about, the predefined value types. Because in C# all of the apparently fundamental value types are in fact built up from the (actually fundamental) object type, the list also indicates which System types in the .Net framework correspond to these pre-defined types.

C# Type	.Net Framework (System) type	Signed?	Bytes Occupied	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	-2147483648 to 2147483647
long	System.Int64	Yes	8	-9223372036854775808 to 9223372036854775807
byte	System.Byte	No	1	0 to 255
ushort	System.UInt16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to 4294967295

ulong	System.UInt64	No	8	0 to 18446744073709551615
float	System.Single	Yes	4	Approximately $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ with 7 significant figures
double	System.Double	Yes	8	Approximately $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	Approximately $\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character (16 bit)
bool	System.Boolean	N/A	1 / 2	true or false

In the following lines of code, two variables are declared and set with integer values.

```
int x = 10;
int y = x;
y = 20; // after this statement x holds value 10 and y holds value 20
```

Reference Types

The pre-defined reference types are object and string, where object - is the ultimate base class of all other types. New reference types can be defined using 'class', 'interface', and 'delegate' declarations. There fore the reference types are :

Predefined Reference Types

- Object
- String

User Defined Reference Types

- Classes
- Interfaces
- Delegates
- Arrays

Reference types actually hold the value of a memory address occupied by the object they reference. Consider the following piece of code, in which two variables are given a reference to the same object (for the sake of the example, this object is taken to contain the numeric property 'myValue').

```
object x = new
object(); x.myValue =
10; object y = x ;
y.myValue = 20; // after this statement both
x.myValue // and y.myValue equal 20
```

This code illustrates how changing a property of an object using a particular reference to it is reflected in all other references to it. Note, however, that although strings are reference types, they work rather more like value types. When one string is set to the value of another, eg

```
string s1 = "hello";
string s2 = s1;
```

Then s2 does at this point reference the same string object as s1. However, when the value of s1 is changed, for instance with

```
s1 = "goodbye";
```

what happens is that a new string object is created for s1 to point to. Hence, following this piece of code, s1 equals "goodbye", whereas s2 still equals "hello".

The reason for this behaviour is that string objects are 'immutable'. That is, the properties of these objects can't themselves change. So in order to change what a string variable references, a new string object must be created.

Boxing

C# allows you convert any value type to a corresponding reference type, and to convert the resultant 'boxed' type back again. The following piece of code demonstrates boxing. When the second line executes, an object is initiated as the value of 'box', and the value held by i is copied across to this object. It is interesting to note that the runtime type of box is returned as the boxed value type; the 'is' operator thus returns the type of box below as 'int'.

```
int i = 123;
object box = i;
if (box is int)
{Console.WriteLine("Box contains an int");} // this line is printed
```

When boxing occurs, the contents of value type are copied from stack into memory allocated into the managed heap. The new reference type created contains a copy of the value type, and can be used by other types that expect an object reference. The value contained in the value type and the created reference types are not associated in any way (except that they contain the same values). If we change the original value type, the reference type is not affected.

The following code explicitly **unboxes** a reference type into a value type:

```
object o;
int i = (int) o;
```

When unboxing occurs, memory is copied from the managed heap to the stack.

2.3 Array and Strings

Arrays

An array is a group or collection of similar values. An array contains a number of variables, which are accessed through computed indices. The various value contained in an array are also called the elements of array. All elements of an array have to be of same type, and this type is called the element type of the array. The element of an array can be of any type including an array type.

An array has a rank that determines the number of indices associated with each array elements. The rank of an array is also referred as the dimension of the array. An array may be :

- Single Dimensional
- Multi Dimensional

An array with a rank of one is called single-dimensional array, and an array with a rank greater than one is called a multi dimensional array.

Each dimension of array has an associated length, which is an integer number greater than or equal to zero. For a dimension of length n, indices can range from 0 to n-1. In C#, array types are categorized under the reference types alongside with classes and interfaces.

Single Dimensional Array

Single -dimensional arrays have a single dimension (ie, are of rank 1). The process of creation of arrays is basically divided into three steps:

1. Declaration of Array
2. Memory Allocation for Array
3. Initialization of Array

Declaration of Array

To declare an array in C# place a pair of square brackets after the variable type. The syntax is given below :

```
type[] arrayname;
```

For Example:

```
int[] a; float[]  
marks;  
double[] x;  
int[] m,n;
```

You must note that we do not enter the size of the arrays in the declaration.

Memory Allocation for Array

After declaring an array, we need to allocate space and defining the size. Declaring arrays merely says what kind of values the array will hold. It does not create them. Arrays in C# are objects, and you use the new keyword to create them. When you create an array, you must tell the compiler how many components will be stored in it. Here is given the syntax:

```
arrayname = new type[size];
```

For Example:

```
a = new int[5];  
marks = new float[6];  
x = new double[10];  
m = int[100];  
n = int [50];
```

It is also possible to combine the two steps, declaration and memory allocation of array, into one as shown below:

```
int[] num = new int [5];
```

Initialization of Array

This step involves placing data into the array. Arrays are automatically assigned the default values associated with their type. For example, if we have an array of numerical type, each element is set to number 0. But explicit values can be assigned as and when desired.

Individual elements of an array are referenced by the array name and a number that represents their position in the array. The number you use to identify them are called subscripts or indexes into the array.

Subscripts are consecutive integers beginning with 0. thus the array “num” above has components **num[0], num[1], num[2], num[3], and num[4]**.

The initialization process is done using the array subscripts as shown:

```
arrayname[subscript] = value;
```

For Example:

```
num[0] = 5;  
num[1] = 15;  
num[2] = 52;  
num[3] = 45;  
num[4] = 57;
```

We can also initialize arrays automatically in the same way as the ordinary variables when they are declared, as shown below:

```
type[] arrayname = { list of values };
```

the list of variables separated by commas and defined on both ends by curly braces. You must note that no size is given in this syntax. The compiler space for all the elements specified in the list.

For Example:

```
int[] num = {5,15,52,45,57};
```

You can combine all the steps, namely declaration, memory allocation and initialization of arrays like as:

```
int[] num = new int [5] {5,15,52,45,57};
```

You can also assign an array object to another. For Example

```
int[] a = { 10, 20,  
30}; int[] b;  
b=a;
```

The above example is valid in C#. Both the array will have same values.

Example

```
using System;
class Number
{
    public static void Main()
    {
        int [] num = {10, 20, 30, 40,
                    50}; int n = num.Length;
        // Length is predefined attribute to access the size of
        // array Console.WriteLine(" Elements of array are :");
        for(int i=0; i<n; i++)
        {
            Console.WriteLine(num[i]);
        }

        int sum =0;
        for(int i=0; i<n; i++)
        {
            sum = sum + num[i];
        }
        Console.WriteLine(" The sum of elements :"+sum);
    }
}
```

OUTPUT:

Elements of array
are: 10 20 30 40 50

The sum of elements :150

Multi Dimensional Array

C# supports two types of multidimensional arrays:

- Rectangular Array
- Jagged Array

Rectangular Arrays

A rectangular array is a single array with more than one dimension, with the dimensions' sizes fixed in the array's declaration. The following code creates a 2 by 3 multi-dimensional array:

```
int[,] squareArray = new int[2,3];
```

As with single-dimensional arrays, rectangular arrays can be filled at the time they are declared. For instance, the code

```
int[,] squareArray = {{1, 2, 3}, {4, 5, 6}};
```

creates a 2 by 3 array with the given values. It is, of course, important that the given values do fill out exactly a rectangular array.

The **System.Array** class includes a number of methods for determining the size and bounds of arrays. These include the methods **GetUpperBound(int i)** and **GetLowerBound(int i)**, which

return, respectively, the upper and lower subscripts of dimension *i* of the array (note that *i* is zero based, so the first array is actually array 0).

For instance, since the length of the second dimension of `squareArray` is 3, the

```
expression squareArray.GetLowerBound(1)
```

returns 0, and the expression

```
squareArray.GetUpperBound(1)
```

returns 2.

System.Array also includes the method **GetLength(int i)**, which returns the number of elements in the *i*th dimension (again, zero based).

The following piece of code loops through `squareArray` and writes out the value of its elements.

```
for(int i = 0; i < squareArray.GetLength(0); i++)
    for (int j = 0; j < squareArray.GetLength(1); j++)
        Console.WriteLine(squareArray[i,j]);
```

A `foreach` loop can also be used to access each of the elements of an array in turn, but using this construction one doesn't have the same control over the order in which the elements are accessed.

Jagged Arrays

Using jagged arrays, one can create multidimensional arrays with irregular dimensions. This flexibility derives from the fact that multidimensional arrays are implemented as arrays of arrays. The following piece of code demonstrates how one might declare an array made up of a group of 4 and a group of 6 elements:

```
int[][] jag = new int[2][];
jag[0] = new int [4];
jag[1] = new int [6];
```

The code reveals that each of `jag[0]` and `jag[1]` holds a reference to a single-dimensional `int` array. To illustrate how one accesses the integer elements: the term `jag[0][1]` provides access to the second element of the first group.

To initialise a jagged array whilst assigning values to its elements, one can use code like the following:

```
int[ ][ ] jag = new int[ ][ ] {new int[ ] {1, 2, 3, 4}, new int[ ] {5, 6, 7, 8, 9, 10}};
```

Be careful using methods like `GetLowerBound`, `GetUpperBound`, `GetLength`, etc. with jagged arrays. Since jagged arrays are constructed out of single-dimensional arrays, they shouldn't be treated as having multiple dimensions in the same way that rectangular arrays do.

To loop through all the elements of a jagged array one can use code like the following:

```
for (int i = 0; i < jag.GetLength(0); i++)
    for (int j = 0; j < jag[i].GetLength(0);
        j++) Console.WriteLine(jag[i][j]);
```

or

```
for (int i = 0; i < jag.Length; i++)
    for (int j = 0; j < jag[i].Length; j++)
        Console.WriteLine(jag[i][j]);
```

Strings

A string is an empty space, a character, a word, or a group of words that you want the compiler to consider "as is", that is, not to pay too much attention to what the string is made of, unless you explicitly ask it to. This means that, in the strict sense, you can put in a string anything you want. Primarily, the value of a string starts with a double quote and ends with a double-quote. An example of a string is "Welcome to the World of C# Programming!". You can include such a string in the **Console.Write()** method to display it on the console. Here is an example:

Example using

```
System; class
```

```
BookClub
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.WriteLine("Welcome to the World of C# Programming!");
```

```
    }
```

```
}
```

OUTPUT:

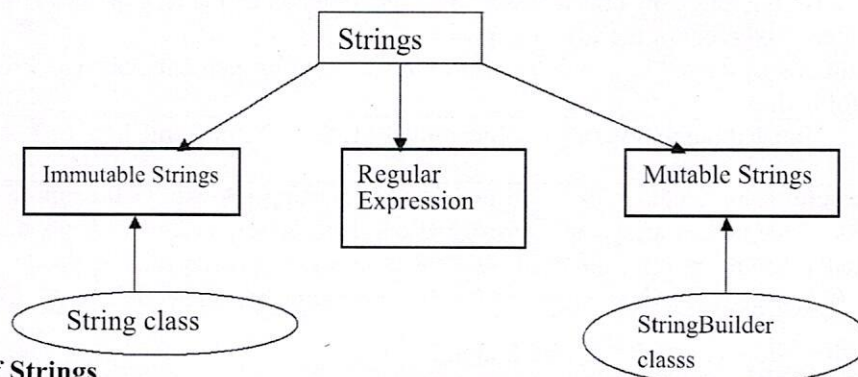
```
Welcome to the World of C# Programming!
```

Types of String

There are two types of string in C#:

- 1) **Immutable strings**
- 2) **Mutable strings**

The immutable strings are can't be modify and mutable strings are modifiable. C# also supports a feature of *regular expression* that can be used for complex strings manipulations and pattern matching.



Handling of Strings

We can create immutable strings using **string** or **String** objects in number of ways. There are some techniques to handling the immutable strings:

Assigning String

```
string s1;  
s1 = "Welcome";
```

or

```
string s1 = "Welcome";
```

Copying String

```
string s2 = s1;
```

or

```
string s2 = string.Copy(s1);
```

Concatenating Strings

```
string s3 = s1 + s2;
```

or

```
string s3 = string.Concat(s1,s2);
```

Reading from Console

```
string s1 = Console.ReadLine();
```

Converting Number to String

```
int num = 100;  
string s1 = num.ToString();
```

Inserting String

```
string s1 = Wel;  
string s2 = s1.insert(3,"come");  
// s2 = Welcome  
string s3 = s1.insert(3,"don");  
// s3 = Weldon;
```

Comparing Strings

```
int n = string.Compare(s1,s2);
```

This statement will perform case-sensitive comparison and returns integer values for different conditions. Such as:

- If s1 is equal to s2 it will return zero.
- If s1 is greater than s2 it will return positive integer (1).
- If s1 is less than s2 it will return negative integer(-

1). Or you can use following statement:

```
bool a = s2.Equals(s1); bool  
b = string.Equal(s1,s2);
```

Above statements will return a Boolean value **true** (if equal) or **false**(if not equal).

Or you can also use the “==” operator for comparing the strings. Like as:

```
if ( s1 == s2)  
Console.Write(“ both are equal”);
```

In this statement, it will return a Boolean value **true** (if equal) or **false**(if not equal).

Mutable String

Mutable strings are those strings, which can be modify dynamically. This type of strings are created using **StringBuilder** class. For Example:

```
StringBuilder s1 = new StringBuilder(“Welcome”);  
StringBuilder s2 = new StringBuilder( );
```

The string **str1** is created with an initial size of seven characters and **str2** is created as an empty string. They can grow dynamically as more character added to them. Mutual string are referred as a *dynamic strings*.

The StringBuilder class supports many methods that are useful for manipulating dynamic strings. Some of the most common methods are listed below:

Method	Operation
Append()	Append a string
AppendFormat()	Append string using specific format
EnsureCapacity()	Ensure sufficient size
Insert()	Insert a string at a specified position
Remove()	Remove specified character
Replace()	Removes previous string with new one

StringBuilder also provides some attributes to access some properties of strings, such as:

Attributes	Purpose
Capacity	To retrieve or set the number of characters the object can hold
Length	To retrieve or set the length
MaxCapacity	To retrieve maximum capacity of the object
[]	To get or set a character at a specified position

Example

```
using System.Text; //For using StringBuilder
using System;

class StrMethod
{
    public static void Main( )
    {
        StringBuilder s = new StringBuilder("C");
        Console.WriteLine(" Stored String is :"+ s);
        Console.WriteLine("Length of string is :"+s.Length);

        s.Append("Sharp ");
        // appending the string s

        Console.WriteLine(" After Append String is :"+ s);
        Console.WriteLine("Length of string is :"+s.Length);

        s.Insert(7,"Language");
        // inserting the string at last in s

        Console.WriteLine("After Insertion String is:"+ s);
        Console.WriteLine("Length of string is :"+s.Length);

        int n = s.Length;

        s[n] = "!";
    }
}
```

```

namespace ex6
{
    class Program
    {
        static void Main(string[] args)
        {
            // Numeric Formatting Examples
            // integer
            int i = 5678;
            string s = string.Format("{0,10:D}", i); // Into a string right aligned 10 width
            Console.WriteLine("{0,10:D}", i); // or output direct
            Console.WriteLine("{0,10:D7}", i); // or output direct with leading 0

            // double and currency in various
            // formats double d=47.5;
            double bigd = 19876543.6754;
            Console.WriteLine("{0,15:C2}", d); // In Uk = £47.50 right aligned in 15 width

            Console.WriteLine("{0,15:N10}", bigd); // Number
            Console.WriteLine("{0,15:E3}", d); // Scientific 4.750E+001
            Console.WriteLine("{0,15:F5}", d); // Fixed Point 47.50000
            Console.WriteLine("{0,15:G4}", d); // Compact 47.5
            Console.WriteLine("{0,10:P2}", d/100.0); // %
            Console.WriteLine("{0,15:R}", bigd); // Roundtrip - not a digit lost

            // Hex-a-diddly-decimal
            Console.WriteLine("{0,10:x8}", i); // lowercase 0000162e
            Console.WriteLine("{0,10:X8}", i); // uppercase 0000162E

            // Date formats
            DateTime dt = DateTime.Now;
            // Standards
            Console.WriteLine("{0:O}", dt); // O or o yyyy'-MM'-dd'T'HH':'mm':'ss'.fffffffz
            Console.WriteLine("{0:R}", dt); // R or r ddd, dd MMM yyyy HH':'mm':'ss 'GMT'
            Console.WriteLine("{0:s}", dt); // s yyyy'-MM'-dd'T'HH':'mm':'ss
            Console.WriteLine("{0:u}", dt); // u yyyy'-MM'-dd HH':'mm':'ss'Z'

            // Using date/time specifiers
            Console.WriteLine("{0:t}", dt); // short time
            Console.WriteLine("{0:T}", dt); // long time
            Console.WriteLine("{0:d}", dt); // short date
            Console.WriteLine("{0:D}", dt); // long date
            Console.WriteLine("{0:f}", dt); // long date / short time
            Console.WriteLine("{0:F}", dt); // long date / long time
            Console.WriteLine("{0:g}", dt); // short date / short time
            Console.WriteLine("{0:G}", dt); // short date / long time
            Console.WriteLine("{0:o}", dt); // Round Trip

            // roll your own...
        }
    }
}

```

Listing A Directory

```
class DirTest1 {
static void Main(){
System.Console.WriteLine("sub directories in this directory
"); string[] dirs = Directory.GetDirectories("C:\\");
int count = dirs.Length;
for (int i=0; i<count; i++)
System.Console.WriteLine(dirs[i]);
System.Console.WriteLine("Files in this directory are");
string[] files = Directory.GetFiles("C:\\");
int count1 = files.Length;
for (int i=0; i<count1; i++)
System.Console.WriteLine(files[i]);
}}
```

Formatted Strings in C#

The parameter placeholder {0}, {1} etc in Console.Write is handled by the string formatting. The placeholder has to have the index of the parameter but can also include formatting information. This is used in Console.WriteLine and also string.format hence its inclusion here.

Layout of the Placeholder

The three parts are

index, alignment : format

So {0:X} (x means Hexadecimal) or {0,10} meaning output in a width of 10. The alignment value specifies the width and left or right alignment by using - (left aligned) or + (right aligned) numbers. 10 Means right aligned in a width of 10, -6 means left aligned in a width of 6. Alignment is ignored if the output exceeds the width.

This provides a large number of formatting examples.

List of Numeric Formats

- C or c - For Currency. Uses the cultures currency symbol.
- D or d - Integer types. Add a number for 0 padding eg D5.
- E or e - Scientific notation.
- F or f - Fixed Point.
- G or g - Compact fixed-point/scientific notation.
- N or n - Number. This can be enhanced by a NumberFormatInfo object.
- P or p - Percentage.
- R or r - Round-trip. Keeps exact digits when converted to string and back.
- X or x - Hexadecimal. x - uses abcdef, X use ABCDEF.

Dates can also be specified either using Standard Format strings

- O or o - YYYY-MM-dd:mm:ss:ffffffzz
- R or r - RFC1123 eg ddd, dd MMM yyyy HH:ss GMT
- s - sortable . yyyy-MM-ddTHH:mm:ss
- u - Universal Sort Date - yyyy-MM-dd HH:mm:ssZ or Format specifiers.

Example

```
using System;
using System.Text;
using System.Globalization;
```

- **Directory Class:** Exposes static methods for creating, moving, and enumerating through directories and subdirectories. This class cannot be inherited.
- **DirectoryInfo Class:** Exposes instance methods for creating, moving, and enumerating through directories and subdirectories. This class cannot be inherited.
- **File Class:** Provides static methods for the creation, copying, deletion, moving, and opening of files, and aids in the creation of FileStream objects.
- **FileInfo :** Provides properties and instance methods for the creation, copying, deletion, moving, and opening of files, and aids in the creation of FileStream objects. This class cannot be inherited.
- **FileStream:** Exposes a Stream around a file, supporting both synchronous and asynchronous read and write operations.
- **IOException Class**
- **Path Class**
- **Stream Class:** Provides a generic view of a sequence of bytes.
- **StreamReader:** Implements a TextReader that reads characters from a byte stream in a particular encoding.
- **StreamWriter :** Implements a TextWriter for writing characters to a stream in a particular encoding.
- **StringReader :** Implements a TextReader that reads from a string.
- **StringWriter:** Implements a TextWriter for writing information to a string. The information is stored in an underlying StringBuilder.
- **TextReader:** Represents a reader that can read a sequential series of characters.
- **TextWriter:** Represents a writer that can write a sequential series of characters. This class is abstract.

Creating and writing text on a File

```

namespace IOtest{
class Program {
    static void Main(string[] args)    {
        StreamWriter sw;
        sw= File.CreateText("d:/workspace/Hello.txt");
        sw.WriteLine("Hello Mca Students This is your basic
        IO"); //sw.Flush();
        //sw.Close();
        Console.WriteLine("Please show the file in d drive ");
    } }

class TextFileWriter {
    static void Main(string[] args)    {
        TextWriter tw = new StreamWriter("date.txt");
        tw.WriteLine(DateTime.Now);
        tw.Close();
    } }

class TextFileReader {
    static void Main(string[] args)    {
        Textreader tr = new StreamReader("date.txt");
        Console.WriteLine(tr.ReadLine()); tr.Close();
    }
}
}

```


Example

```
using System;
using System.Text;
class Program
{
    static void Main()
    {
        StringBuilder builder = new StringBuilder();
        // Append to StringBuilder.
        for (int i = 0; i < 10; i++)
        {
            builder.Append(i).Append(" ");
        }
        Console.WriteLine(builder);
    }
}
```

3.2 Input-Output

C# programs perform I/O through streams. A *stream* is an abstraction that either produces or consumes information. A stream is linked to a physical device by the I/O system. All streams behave in the same manner, even if the actual physical devices they are linked to differ. Thus, the I/O classes and methods can be applied to many types of devices. For example, the same methods that you use to write to the console can also be used to write to a disk file.

Byte Streams and Character Streams

At the lowest level, all C# I/O operates on bytes. This makes sense because many devices are byte oriented when it comes to I/O operations. Frequently, though, we humans prefer to communicate using characters. Recall that in C#, **char** is a 16-bit type, and **byte** is an 8-bit type. If you are using the ASCII character set, then it is easy to convert between **char** and **byte**; just ignore the high-order byte of the **char** value. But this won't work for the rest of the Unicode characters, which need both bytes (and possibly more). Thus, byte streams are not perfectly suited to handling character-based I/O. To solve this problem, the .NET Framework defines several classes that convert a byte stream into a character stream, handling the translation of **byte-to-char** and **char-to-byte** for you automatically.

The Predefined Streams

Three predefined streams, which are exposed by the properties called **Console.In**, **Console.Out**, and **Console.Error**, are available to all programs that use the **System** namespace. **Console.Out** refers to the standard output stream. By default, this is the console. When you call **Console.WriteLine()**, for example, it automatically sends information to **Console.Out**. **Console.In** refers to standard input, which is, by default, the keyboard. **Console.Error** refers to the standard error stream, which is also the console by default. However, these streams can be redirected to any compatible I/O device. The standard streams are character streams. Thus, these streams read and write characters.

System.IO Namespace

- **BinaryReader Class**: Reads primitive data types as binary values in a specific encoding.
- **BinaryWriter Class** : Writes primitive types in binary to a stream and supports writing strings in a specific encoding.
- **BufferedStream Class** : Adds a buffering layer to read and write operations on another stream. This class cannot be inherited.

```

str = Console.ReadLine();
s2 = Double.Parse(str);
hypot = Math.Sqrt(s1*s1 + s2*s2);
Console.WriteLine("Hypotenuse is " + hypot);
}
}

```

Sorting and Searching, Reverse, Copy Arrays

Using `Sort()`, you can sort an entire array, a range within an array, or a pair of arrays that contain corresponding key/value pairs. Once an array has been sorted, you can efficiently search it using `BinarySearch()`.

```

using System; class
SortDemo { static
void Main() {
int[] nums = { 5, 4, 6, 3, 14, 9, 8, 17, 1, 24, -1, 0
}; Console.Write("Original order: ");
foreach(int i in nums)
Console.Write(i + " "); Console.WriteLine();
Array.Sort(nums);
Console.Write("Sorted order: ");
foreach(int i in nums)
Console.Write(i + " "); Console.WriteLine();
int idx = Array.BinarySearch(nums, 14);
Console.WriteLine("Index of 14 is " + idx); } }

```

The IComparable and IComparable<T> Interfaces

- Many classes will need to implement either the `IComparable` or `IComparable<T>` interface because they enable one object to be compared to another (for the purpose of ordering) by various methods defined by the .NET Framework
- **`IComparable` is especially easy to implement because it consists of just this one method:**
- `int CompareTo(object obj)`
- This method compares the invoking object against the value in *obj*. *It returns greater than zero if the invoking object is greater than obj, zero if the two objects are equal, and less than zero if the invoking object is less than obj.*

StringBuilder in C#

Once created a string cannot be changed. A `StringBuilder` can be changed as many times as necessary. It yields astonishing performance improvements. It eliminates millions of string copies. Many C# programs append or replace strings in loops. There the `StringBuilder` type becomes a necessary optimization. It uses the new keyword for `StringBuilder`. Use the new keyword to make your `StringBuilder`. This is different from regular strings. `StringBuilder` has many overloaded constructors. continuing on it calls the instance `Append` method. This method adds the contents of its arguments to the buffer in the `StringBuilder`. Every argument to `StringBuilder` will automatically have its `ToString` method called. It calls `AppendLine`, which does the exact same thing as `Append`, except with a new line on the end. Next, `Append` and `Append Line` call themselves. This shows terse syntax with `StringBuilder`. Finally `ToString` returns the buffer. You will almost always want `ToString`. It will return the contents as a string.

Unit-III

C# Using Libraries

3.1 Namespace- System

System Namespace in fundamental namespace for c# application. It contain all the fundamental classes and base classes which are required in simple C# application. These classes and sub classes defines reference data type, method and interfaces. Some classes provide some other feature like data type conversion, mathematical function.

Some functionality provided by System namespace

- Commonly-used value
- Mathematics
- Remote and local program invocation
- Application environment management
- Reference data types
- Events and event handlers
- Interfaces Attributes Processing exceptions
- Data type conversion
- Method parameter manipulation

Some Classes provide by System namespace

- AccessViolationException
- Array
- ArgumentNullException
- AttributeUsageAttribute
- Buffer
- Console
- Convert
- Delegate
- Exception
- InvalidCastException

Some interfaces provided by System namespace

- Public interface ICloneable
- Public interface IComparable
- Public interface IComparable<T>
- Public interface IConvertible
- Public interface ICustomFormatter
- Public interface IDisposable
- Public interface IEquatable<T>
- Public interface IFormatProvider

MATH EXAMPLE

```
using System; class
Pythagorean { static
void Main() {
double s1;
double s2;
double hypot;
string str;
Console.WriteLine("Enter length of first side:
"); str = Console.ReadLine();
s1 = Double.Parse(str);
Console.WriteLine("Enter length of second side: ");
```

- When s is a null reference, it will return 0 rather than `throwArgumentNullException`.
- If s is other than an integer value, the out variable will have 0 rather than `FormatException`.
- When s represents a number less than `MinValue` or greater than `MaxValue`, the outvariable will have 0 rather than `OverflowException`.
- `success = Int32.TryParse(s1, out result);` *//-- success => true; result => 1234*
- `success = Int32.TryParse(s2, out result);` *//-- success => false; result => 0*
- `success = Int32.TryParse(s3, out result);` *//-- success => false; result => 0*
- `success = Int32.TryParse(s4, out result);` *//-- success => false; result => 0*
- `Convert.ToInt32` is better than **`Int32.Parse`** since it returns 0 rather than an exception. But again, according to the requirement, this can be used. `TryParse` will be the best since it always handles exceptions by itself.

Unboxing

- Unboxing is a mechanism in which reference type is converted into value.
- It is explicit conversion process.

Example

```
int i, j;
object obj;
string s;
i = 32;
obj = i; // boxed copy!
i = 19;
j = (int) obj; // unboxed!
s = j.ToString(); // boxed!
s = 99.ToString(); // boxed!
```

Difference Between Int32.Parse(), Convert.ToInt32(), and Int32.TryParse(),(int)

Int32.Parse (string s) method converts the string representation of a number to its 32-bit signed integer equivalent.

- When s is a null reference, it will throw `ArgumentNullException`.
 - If s is other than integer value, it will throw `FormatException`.
 - When s represents a number less than `MinValue` or greater than `MaxValue`, it will throw `OverflowException`.
- `(int)` will only convert types that can be represented as an integer (ie double, long, float, etc) although some data loss may occur.
 - `string s1 = "1234";`
 - `string s2 = "1234.65";`
 - `string s3 = null;`
 - `string s4 = 123456789123456789123456789123456789123456789123456789123456789";`
 - `int result;`
 - `bool success;`
 - `result = Int32.Parse(s1); //-- 1234`
 - `result = Int32.Parse(s2); //-- FormatException`
 - `result = Int32.Parse(s3); //-- ArgumentNullException`
 - `result = Int32.Parse(s4); //-- OverflowException`

Convert.ToInt32(string): This method converts the specified string representation of 32-bit signed integer equivalent. This calls in turn **Int32.Parse ()** method.

- When s is a null reference, it will return 0 rather than throw `ArgumentNullException`.
 - If s is other than integer value, it will throw `FormatException`.
 - When s represents a number less than `MinValue` or greater than `MaxValue`, it will throw `OverflowException`. For example:
- `result = Convert.ToInt32(s1); //-- 1234`
 - `result = Convert.ToInt32(s2); //-- FormatException`
 - `result = Convert.ToInt32(s3); //-- 0`
 - `result = Convert.ToInt32(s4); //-- OverflowException`

Int32.TryParse(string, out int): This method converts the specified string representation of 32-bit signed integer equivalent to out variable, and returns true if it is parsed successfully, false otherwise.

2.8 Type conversion

There are two types of conversions:

1. Implicit Conversion
2. Explicit Conversion

Implicit Conversion : In implicit conversion the compiler will make conversion for us without asking.

char -> int -> float is an example of data compatibility. Compiler checks for type compatibility at compilation.

Explicit Conversion: In explicit conversion we specifically ask the compiler to convert the value into another data type. CLR checks for data compatibility at runtime. Explicit conversion is carried out using casts. When we cast one type to another, we deliberately force the compiler to make the transformation. Casting of big data type into small may lead to losing of data.

Microsoft .NET provides three ways of type conversion:

1. Parsing(`int.Parse()`)
2. Convert Class(`Convert.ToInt32()`)
3. Explicit Cast Operator ()

Parsing

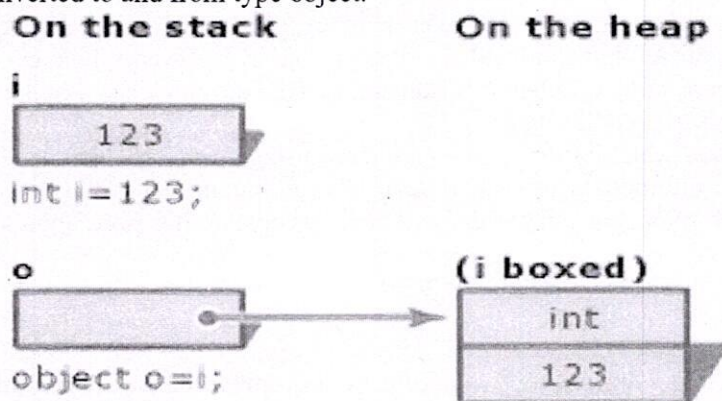
Parsing is used to convert string type data to primitive value type. For this we use parse methods with value types.

Convert Class: One primitive type to another primitive type.

This class contains different static methods like `ToInt32()`, `ToInt16()`, `ToString()`, `ToDateTime()` etc used in type conversion.

Boxing and unboxing

Boxing and unboxing is an important concept in C# type system. With Boxing and unboxing one can link between value-types and reference-types by allowing any value of a value-type to be converted to and from type object.



Boxing

- Boxing is a mechanism in which value type is converted into reference type.
- It is implicit conversion process in which object type (super type) is used.
- In this process type and value both are stored in object type

2. Class 2 wants to respond to E-events with its event-handling method M. It therefore adds onto D a reference to M.

3. When Class 1 wants to issue an E-event, it calls D. This invokes all of the methods which have subscribed to the event, including M.

The 'event' keyword is used to declare a particular multicast delegate (in fact, it is usual in the literature to just identify the event with this delegate). The code below shows a class EventIssuer, which maintains an event field myEvent. We could instead have declared the event to be a property instead of a field. To raise the myEvent event, the method onMyEvent is called (note that we are checking in this method to see if myEvent is null - trying to trigger a null event gives a run-time error).

```
public class EventIssuer
{
    public delegate void EventDelegate(object from, EventArgs
    args); public event EventDelegate myEvent;
    protected virtual void onMyEvent(EventArgs args)
    {
        if (myEvent!=null)
            myEvent(this, args);
    }
}
```

A class which wanted to handle the events issued by an EventIssuer ei with its method handleEvents would then subscribe to these events with the code:

```
ei.myEvent += new EventIssuer.EventDelegate(handleEvents);
```

Tips For Events

The code above demonstrates some points about event-handling which are not enforced by the language architecture, but are used throughout the .Net framework as good practice.

1. When you want to raise an event in code, you don't tend to trigger the class's event object directly. Rather, you call a 'protected, virtual' method to trigger it (cf. the onMyEvent method above).
2. By convention, when events are raised they pass two objects to their subscribers. The first is a reference to the class raising the event; the second is an instance of the System.EventArgs class which contains any arbitrary data about the event.
3. If an event is not interested in passing data to subscribers, then its defining delegate will still reference an EventArgs object (but a null value will be passed by the event). If an event should pass data to its subscribers, however, then it is standard to use a specific class which derives from the EventArgs class to hold this data.
4. When you write a class which inherits from an event-raising base class, you can 'intercept' an event by overriding the method used to raise it. The following code illustrates such an intercept - classes which subscribe to the event will never receive notifications about it.

```
protected override void onMyEvent(EventArgs args)
{
    Console.WriteLine("hello"); }
}
```

If you want subscribers to continue to receive notifications despite such an 'intercepting' method, however, then you can call the base class method as in the following:

```
protected override void onMyEvent(EventArgs
args) { Console.WriteLine("hello");
    base.onMyEvent(args); }
```

Environment

This folder has the details for the production or the dev environment. The folder contains two files.

- environment.prod.ts
- environment.ts

Both the files have details of whether the final file should be compiled in the production environment or the dev environment.

The additional file structure of Angular 4 app folder includes the following –
favicon.ico : This is a file that is usually found in the root directory of a website.

index.html : This is the file which is displayed in the browser.

```
<!doctype html>
<html lang = "en">
  <head>
    <meta charset = "utf-8">
    <title>HTTP Search Param</title>
    <base href = "/">
    <link href = "https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <link href = "https://fonts.googleapis.com/css?family=Roboto|Roboto+Mono"
rel="stylesheet">
    <link href = "styles.c7c7b8bf22964ff954d3.bundle.css" rel="stylesheet">
    <meta name = "viewport" content="width=device-width, initial-scale=1">
    <link rel = "icon" type="image/x-icon" href="favicon.ico">
  </head>

  <body>
    <app-root></app-root>
  </body>
</html>
```

The body has **<app-root></app-root>**. This is the selector which is used in **app.component.ts** file and will display the details from app.component.html file.

main.ts is the file from where we start our project development. It starts with importing the basic module which we need. Right now if you see angular/core, angular/platform-browser-dynamic, app.module and environment is imported by default during angular-cli installation and project setup.

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';
if (environment.production) {
  enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule);
```

The **platformBrowserDynamic().bootstrapModule(AppModule)** has the parent module reference **AppModule**. Hence, when it executes in the browser, the file that

is called is index.html. Index.html internally refers to main.ts which calls the parent module, i.e., AppModule when the following code executes –

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

When AppModule is called, it calls app.module.ts which further calls the AppComponent based on the bootstrap as follows –

```
bootstrap: [AppComponent]
```

In app.component.ts, there is a **selector: app-root** which is used in the index.html file. This will display the contents present in app.component.html.

polyfill.ts

This is mainly used for backward compatibility.

styles.css

This is the style file required for the project.

test.ts

Here, the unit test cases for testing the project will be handled.

tsconfig.app.json

This is used during compilation, it has the config details that need to be used to run the application.

tsconfig.spec.json

This helps maintain the details for testing.

typings.d.ts

It is used to manage the TypeScript definition.

Components

Major part of the development with Angular 4 is done in the components. Components are basically classes that interact with the .html file of the component, which gets displayed on the browser. We have seen the file structure in one of our previous chapters. The file structure has the app component and it consists of the following files –

- **app.component.css**
- **app.component.html**
- **app.component.spec.ts**
- **app.component.ts**
- **app.module.ts**

The above files were created by default when we created new project using the angular-cli command.

If you open up the **app.module.ts** file, it has some libraries which are imported and also a declarative which is assigned the appcomponent

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
```

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

To create a new Component:

```
ng g component new-cmp
```

installing component

```
create src\app\new-cmp\new-cmp.component.css // CSS
create src\app\new-cmp\new-cmp.component.html // HTML
create src\app\new-cmp\new-cmp.component.spec.ts // unit testing
create src\app\new-cmp\new-cmp.component.ts // module, properties, etc
update src\app\app.module.ts // Updation of root app
```

In the app.component.html

```
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
<!--To include the design of the new components -->
  <app-new-cmp></app-new-cmp>
  <app-my-new-component></app-my-new-component>
</div>
```

Modules

Module in Angular refers to a place where you can group the components, directives, pipes, and services, which are related to the application.

In case you are developing a website, the header, footer, left, center and the right section become part of a module.

To define module, we can use the **NgModule**. When you create a new project using the Angular `-cli` command, the `ngmodule` is created in the `app.module.ts` file by default

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
```