**Day 1**

**15-03-2021**

**Full Stack :**

Program

Input a=10, b=20

Process sum = a+a

Output write sum

class Abc {

public int add(int x, int y) {

int sum = x+y;

return sum;

}

}

**Testing: We are going to check our code working fine or not without main method or functions.**

**Testing : It use to find the error or defect or bugs in application.**

**Testing divided into 2 types**

1. **Black box testing**
2. **White box testing**

**Black box testing**

**Input -----🡪 Output**

**White box testing**

**Input -🡪 Process -🡪 Output**

**Unit Testing : Unit testing is a type of white box testing.**

**Unit is where we write the set of line of coding to perform a specific task. Like function/ method or class or modules etc.**

**To achieve unit testing. Third party vendor provide API(Application Programming interface)**

**jUnit :jUnit is small testing framework open source framework provide by vendor which help to do the java programming unit testing.**

**To do jUnit testing it required jar file.**

**jUnit Test Case : jUnit Test case is class which contains more than one test methods with Annotation @Test.**

**Every @Test method can contains more than assertXXX() methods to check actual and expectation values.**

**jUnit Test Suite: Test Suite combines more than one test case classes.**

**JDBC :**

**Database :**

**Account Table : In Database**

**AccNo(PK), Name, Amount**

**1120 Ravi 1200**

**1121 Ramesh 1500**

**1122 Ajay 1600**

**commit;**

**AccountDao (Data Access Object)**

**class AccountDao {**

**public float getBalance(int accnot) {**

**JDBC logic**

**Get balance amount**

**}**

**}**

**Class AccountDao {**

**@Test void testGetBalance(){**

**}**

**}**

**Maven :**

**Gradle**

**Git :**

**Agile :**

**Docker :**

**Kubernetes :**

**Maven : Maven is known a build tools.**

**Build tool is responsible to compile the program, run the program, creating jar, war or ear file, downloading dependencies or jar files depending upon the application requirements. Creating the documentation for the projects.**

**Before Maven We were using ANT tool.**

**Netbean**

**Eclipse :**

**MyEclipse**

**RAD**

**Etc**

**Maven use POM.xml file.**

**POM is known as Project Object Model. It is a type of xml file which hold all configuration details for the maven projects.**

**Creating sample project using maven command prompt**

**mvn archetype:generate**

**Maven Project : For all we have to give groupName using Id.**

**groupId : name for more than project with id identifiers.**

**groupId : groupNameforMoreThanOneproject**

**artifactId : actualProjectName:**

**Maven Life cycle :**

**3 life cycle :**

1. **clean**
2. **build (default )**
3. **site**

**Every life cycle contains more than one phase :**

**Phase : phase is a group of orders goals.**

**Goal : Goal is a single unit of task which does some real work.**

**Goal 🡪**

**Phase 🡪**

**Life cycle 🡪**

**Maven phase :**

**mvn validate : mvn validate : it check the pom.xml file code.**

**mvn clean**

**mvn compile**

**mvn test**

**mvn install**

**mvn deploy**

**mvn exec:java –Dexec.mainClass=com.App**

**Day 2**

**16-03-2021**

**Creating the maven project using eclipse IDE**

**Maven tool use repository(Folder or directory) concept to hold to hold/ store dependencies(vendor Jar files) required for the application.**

**Maven uses local and remote repository.**

**Local repository:**

**Default path :**

**C:\Users\91990\.m2\repository**

**We have to add few tags in pom.xml file to download the external dependencies jar file. First maven search that version jar file from local repository part of m2 folder. If that version jar file available then that jar add the eclipse or maven projects. If that jar is not available in local repository it download from remote repository and keep in local repository for features reference.**

**If we want to download the external Dependencies**

<dependencies>

<dependency>

<groupId></groupId>

<artifactId></artifactId>

<version></version>

</dependency>

<dependency>

<groupId></groupId>

<artifactId></artifactId>

<version></version>

</dependency>

</dependencies>

**Here we have to remember groupId, artifactid and version of external jar files.**

**Maven documentation provide all jar file groupId,artifactid and version.**

**Creating jar or war file for user-defined project using maven.**

**mvn package ( This command we have to execute in that directory where pom.xml file present)**

**This command is use to create jar or war file depending upon type of project.**

**If you want user-defined name for the jar file**

**Then we have to write few tags in pom.xml file**

<build>

<finalName>MyJar</finalName>

</build>

Running jar file through command prompt

**mvn install : This command is use to add our local jar file into local repository part of .m2 folder.**

**Web Technologies**

**https://**[**www.google.com**](http://www.google.com) **URL**

**http/https(req)**

**---------🡪 Req---🡪**

**Client Server**

**🡨-----Res(http/https)-----**

**http: hyper text transfer protocol :**

**set of rules which help to communicate more than one machine or device.**

**s : secure**

**www : world wide web**

**google : domain or server**

**com : commercial**

**URL : Uniform resource locator :**

**HTML/HTML5 : display the contents on browser**

**CSS/CSS3 : apply good look and feel for web contents or presentation on contents.**

**JS(JavaScript): It is use to do action on contents.**

* 1. **: HTML,CSS and JavaScript**

**HTML : Hyper text mark up language. It is use to create the web page. Web page means display the contents in browser in different formats.**

**HTML contains lot of pre-defined tags or elements.**

**Syntax**

**<tagName> opening tag**

**</tagName> closing tag**

**Few contains doesn’t contains closing tag**

**<tagName/> self closing tags.**

**HTML is case insensitive.**

**Means we can write tags in upper case or lower case or any case.**

1. **Html**
2. **Head**
3. **Body**
4. **Title tag : This tag is use to display the title message on title bars. It must in between head tags.**

**HTML 4 version**

**<!doctype HTML public URL=”………….dtd”**

**Document type definition : This file contains the rule for the HTML file.**

**What the root tag name.**

**That tag contains how many child tag**

**Head and body**

**Body tags contains how many paragraph tags**

**0 to infinity**

**.xhtml**

**HTML 5**

**<!DOCTYPE html > : giving the instruction to browser we are going to write HTML 5 features.**

**. without this we can use all HTML5 Features.**

**In HTML5 they introduce more tags.**

**We can’t achieve all HTML5 features alone we have to depends upon JS and servers.**

**Java**

**Asp.net**

**Php**

**VS Code ( Visual Studio Code).**

**Hyper link <a href=”demo.html”>Click here</a>**

**External Hyper link**

**Internal Hyper link (book mark)**

**Image tag**

**List tag UL, OL etc**

**Table tags**

**Forms tags**

**HTML form internally use Query param concept**

**URL?user=Ravi&pass=1234**

**By default form use get method. If method is get the information send through URL using query PARAM concept.**

**Then we can use post() if we want to send the data with security.**

**If the method is post then information or data send through body part of request.**

**Performance wise get is faster than post method.**

**And in get we can send maximum 255 character data only.**

**CSS : Cascading Style sheet**

**CSS provide lot of property in the form of key-value pairs which help apply good look and feel for the web page which not possible or difficult with alone html.**

**If we are going control the contents as well as formatting style in html page. The maintain that application become very difficult.**

**Separation concern(Actual content and formatting style must be separate).**

**CSS mainly divided into 3 types.**

1. **Inline CSS**
2. **Internal css or embedded css**
3. **External css**

**Inline CSS**

**Syntax**

**<tagName style=”property:value;property:value;”></tagName>**

**tagName : p, h1 to h6, div, b, I, form, table etc**

**Internal or Embedded CSS**

**<style type=”text/css”>**

**Selector {property:value;property:value}**

**</style>**

**Selector**

1. **Universal selector : \***
2. **Specific selector : tagName**
3. **Multi specific selector : tagName,tagName**
4. **Class selector (local) : tagName.className**
5. **Class selector (global) .className**
6. **Id selector : #idName**

**Class selector Vs Id selector**

**Two tags may have tagName, same name attribute can be belong to same class.**

**Class : collection of more than one tags.**

**Id must unique for tags.**

**Using Id only we can read, write and update the DOM(Document Object Model).**

**Day 3**

**17-03-2021**

**External CSS**

**HTML5 Semantic tags**