**Notes: First you prepare how to write a code for syntax and write a code**

**Java: (day one)**

1. **Steps to install java**

Ans:1) Go to //www.oracle.com/technetwork/java/javase/downloads

2) Click on Java Platform (JDK) 8u111 / 8u112 and download the file according to our OS by accepting the license agreement.

3) Download and run the JDK installer by clicking on Run as Admin, Next and Finish.

**Steps to set JDK path.**

1. Open C Drive🡪 Program files🡪Java🡪jdk.101🡪bin🡪 copy path
2. Right click on My PC 🡪 Advance Settings🡪 Environment Variables🡪 Path🡪 Edit🡪New🡪Pate🡪 Ok
3. Repeat same process for jre.101

1. Steps to install eclipse
2. Go to <http://www.eclipse.org/downloads/eclipse-packages/>
3. Click on download [Eclipse IDE for Java EE Developers](http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/neon1a)
4. Run and Install
5. Open Eclipse folder from downloads and extract files to C drive.
6. In C drive, open eclipse🡪 eclipsec🡪send to desktop
7. Click on eclipse Application🡪 send to Desktop
8. Steps to create workspace
9. Open Eclipse application
10. After Eclipse starts up it displays a dialog titled "Workspace Launcher".
11. Select a filesystem directory for workspace which creates selenium workspace to create and customize projects.
12. Steps to create project
13. Click on File🡪 New🡪 Java Project🡪Give a project name

By default, Eclipse specifies a /src subdirectory for source files and the /bin sub-directory for compile class files

5. create .java file/class

1) Right click on our Project 🡪 new class and give extension .java

When you create a class it will create class name with keyword class and name of class

Class will contain method and properties

Ex: Class Employee {

}

1. How to create packages and what is best way to give name:
2. Right click on src library 🡪 New🡪 Package
3. Package name should start with lower case. Break down your packages according to their functionality or modules.

e.g: com.company.product.modules

1. what is main method will do?
2. Main method is starting point of any java program. when you execute your class, anything in the **main method** runs.
3. in the Java language, when you execute a class , the runtime system starts by calling the class's main() method. The main() method then calls all the other methods required to run your application.
4. What is variable?
5. Variable is a name of memory location. It will store the value in memory
6. There are three types of variables in java: local, instance and static.

#### Local Variable- A variable which is declared inside the method is called local variable.

#### Instance Variable- A variable which is declared inside the class but outside the method, is called instance variable. It is not declared as static.

#### Static variable- A variable that is declared as static is called static variable. It cannot be local.

Example:

**class** A{

**int** data=50;//instance variable

**static** **int** m=100;//static variable

**void** method(){

**int** n=90;//local variable

}

}//end of class

To create variable we specify

Variablename datatype;

1. What is data type and different data types`

Data types represent the different values to be stored in the variable. In java, there are two types of data types:

1. Primitive data types – Primitive datatypes are predefined by the language and named by a keyword.

Types: int, short, long, double, float, byte, boolean, char

1. Non-primitive data types **are** not defined by the programming language, but are instead created by the programmer. They are sometimes called "reference variables," or "object references," since they reference a memory location, which stores the **data**.

creating property/data members : we create properties at class level

int salary

String firstname

creating method with void : we write methods in

Ex: Void MethodName();

creating method with void and parameter

Ex: Void MethodName(int param1);

creating method with return data type

Ex: int MethodName();

     String MethodName();

creating method with return data type and parameter

int MethodName(int id) {

Int id;

Return id;

}

String MethodName();

creating variable

Int var1;

creating static property: All instances shared the value http://crunchify.com/java-static-methods-variables-static-block-and-class-with-example/

creating static method

http://crunchify.com/java-static-methods-variables-static-block-and-class-with-example/

12. creating object:

An object is an instance of a class. It has state and behavior and stores entire class information.

<classname> <objectname> = new <classname>();

13. calling method with no return and no parameter

Ex:

package examplepack;

public class Mathematics

{

Void sum() /\* No arg \*/

{ int a=10;

int b=20;

int c=a+b;

System.out.println(“addition value is:” +c);

}

public static void main(String[] args)

{

Mathematics obj=new Mathematics();

obj.Sum(); /\* no return type\*/

}

}

14. calling method with return and no parameter

package examplePack;

public class Mathematicaly

{

int Sum()

{

int a=10;

int b=20;

int c=a+b;

return c;

}

public static void main(String[] args)

{

Mathematicaly obj=new Mathematicaly();

int r=obj.Sum(); /\* No parameter with return type \*/

System.out.println("Addition values : "+r);

}

}

15. with parameter and no return type:

package examplePack;

public class Mathematicaly

{

void Sum(int x,int y)

{

int a=x;

int b=y;

int c=a+b;

System.out.println("Addition values : "+c);

}

public static void main(String[] args)

{

Mathematicaly obj=new Mathematicaly();

obj.Sum(100,300); /\*parameter with no return\*/

}

}

16. calling method with return and parameter

package examplePack;

public class Mathematicaly

{

int Sum(int x,int y)

int a=x;

int b=y;

int c=a+b;

return c;

}

public static void main(String[] args)

{

// Declare the class object

Mathematicaly obj=new Mathematicaly();

int r=obj.Sum(100,300); /\*parameter with return type\*/

System.out.println("Addition value : "+r);

}

}

calling static method

using static property: it will maintain

**Ans:** public static int empid;

**Java: Day2**

create classes under multiple packages

calling classes under different packages

18. write code to handle exceptions with try/catch/finally

public class Exmpl {

public static void main(String args[]) {

int a[] = new int[4];

try {

System.out.println("Access element three :" + a[5]);

}catch(ArrayIndexOutOfBoundsException e) {

System.out.println("Exception thrown :" + e);

}finally {

a[0] = 10;

System.out.println("First element value: " + a[0]);

System.out.println("The finally statement is executed");

}

}

}

19. what is final keyword

Ans: The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable
2. method
3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only

20. write code for interface and create class to implement that interface

Ans :

interface Employee {

public void eno( );

public void salary( );

}

**Implementation:**

interface MyInterface

{ public void method1();

public void method2();

}

Class Example implements MyInterface

{

public void method1()

{ System.out.println(“implementation of method1”); }

public void method2()

{

System.out.println(“implementation of method2”);

}

public static void main(String arg[])

{ Myinterface obj = new Example();

obj.method1(); }

21. write code for creating abstract class

**Ans:**

**abstract** **class** Bike{

**abstract** **void** run();

}

**class** Honda **extends** Bike{

**void** run(){ System.out.println("running safely.."); }

**public** **static** **void** main(String args[]){

 Bike obj = **new** Honda();

obj.run();

}

}

22. implement method overloading

Ans : class Display

{

public

Ans :

class DisplayOverloading

{

public void disp(char c)

{

System.out.println(c);

}

public void disp(char c, int num)

{

System.out.println(c + " "+num);

}

}

class Sample

{

public static void main(String args[])

{

DisplayOverloading obj = new DisplayOverloading();

obj.disp('a');

obj.disp('a',10);

}

}

implement method overriding

implementing polymorphism

write a code to save data into excel file and read from excel file (POI and jexcel API)

**Read from excel:**

public static void main(String[] args) throws IOException {

        String excelFilePath = "Books.xlsx";

        FileInputStream inputStream = new FileInputStream(new File(excelFilePath));

        XSSFWorkbook workbook = new XSSFWorkbook(inputStream);

        XSSFSheet firstSheet = workbook.getSheetAt(0);

        Iterator<Row> iterator = firstSheet.iterator();

        while (iterator.hasNext()) {

            Row nextRow = iterator.next();

            Iterator<Cell> cellIterator = nextRow.cellIterator();

            while (cellIterator.hasNext()) {

                Cell cell = cellIterator.next();

                switch (cell.getCellType()) {

                    case Cell.CELL\_TYPE\_STRING:

                        System.out.print(cell.getStringCellValue());

                        break;

                    case Cell.CELL\_TYPE\_BOOLEAN:

                        System.out.print(cell.getBooleanCellValue());

                        break;

                    case Cell.CELL\_TYPE\_NUMERIC:

                        System.out.print(cell.getNumericCellValue());

                        break;

                }

                System.out.print(" - ");

            }

            System.out.println();

        }

        workbook.close();

        inputStream.close();

    }

**Write into Excel :**

XSSFWorkbook workbook = **new** XSSFWorkbook();

XSSFSheet sheet = workbook.createSheet("Sample sheet");

Map<String, Object[]> data = **new** TreeMap<String, Object[]>();

data.put("1", **new** Object[] {"empNo.", "name", "salary"});

data.put("2", **new** Object[] {1, "John", 1500000d});

data.put("3", **new** Object[] {2, "Sam", 800000d});

data.put("4", **new** Object[] {3, "Dean", 700000d});

Set<String> keyset = data.keySet();

**int** rownum = 0;

**for** (String key : keyset) {

Row row = sheet.createRow(rownum++);

Object [] objArr = data.get(key);

**int** cellnum = 0;

**for** (Object obj : objArr) {

Cell cell = row.createCell(cellnum++);

**if**(obj **instanceof** Integer)

cell.setCellValue((Integer)obj);

**else** **if**(obj **instanceof** String)

cell.setCellValue((String)obj);

**else** **if**(obj **instanceof** Double)

cell.setCellValue((Double)obj);

}

}

**try** {

//new excel file created by fileoutput stream object

FileOutputStream out =

**new** FileOutputStream(**new** File("C:\\Users\\Public\\Neelima2.xlsx"));

workbook.write(out);

out.close();

System.***out***.println("Excel written successfully..");

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

how to update the data into XML file and read data from XML file

**Read XML file data:**

**public** **static** **void** main(String argv[]) {

**try** {

File fXmlFile = **new** File("C:\\Users\\sindhu\\Desktop\\Sample.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

//optional, but recommended

//read this - http://stackoverflow.com/questions/13786607/normalization-in-dom-parsing-with-java-how-does-it-work

doc.getDocumentElement().normalize();

System.***out***.println("Root element :" + doc.getDocumentElement().getNodeName());

NodeList nList = doc.getElementsByTagName("staff");

System.***out***.println("----------------------------");

**for** (**int** temp = 0; temp < nList.getLength(); temp++) {

Node nNode = nList.item(temp);

System.***out***.println("\nCurrent Element :" + nNode.getNodeName());

**if** (nNode.getNodeType() == Node.***ELEMENT\_NODE***) {

Element eElement = (Element) nNode;

System.***out***.println("Staff id : " + eElement.getAttribute("id"));

System.***out***.println("First Name : " + eElement.getElementsByTagName("firstname").item(0).getTextContent());

System.***out***.println("Last Name : " + eElement.getElementsByTagName("lastname").item(0).getTextContent());

System.***out***.println("Nick Name : " + eElement.getElementsByTagName("nickname").item(0).getTextContent());

System.***out***.println("Salary : " + eElement.getElementsByTagName("salary").item(0).getTextContent());

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

write code to add items to integer, string **array**

int[] intArray= new int[3];

int[0]=1;

int[1]=2;

int[] intArray= {2,25};

String[] stringArray= new String[2];

String[0]=”Hello”;

String[1]=”World”;

String[] stringArray= {“Hello”,”World”};

write code to retrieve items from integer, string array

**import** java.util.Scanner;

**public** **class** ArrayDemo {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int**[] array = **new** **int**[5];

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.println("Enter the values of the array");

**for**(**int** i=0;i<array.length;i++){

array[i] = scanner.nextInt();

}

System.***out***.println("Array elements are:");

**for**(**int** i=0;i<array.length;i++){

System.***out***.println(array[i]);

}

}

}

write code to add items to ArrayList collection

Ans :

List<String> list= new ArrayList<String>();

List.add(“Hello World”);

write code to retrieve items from arraylist (using for each loop\_

for(String str:list){

System.out.println(str);

}

write code to add items HashMap

Map<Integer,String> students = new HashMap<>();

students.put(1, "Hello");

students.put(2, "World");

write code to retrieve items HashMap

Map<Integer,String> students = new HashMap<>();

students.put(1, "Hello");

students.put(2, "World");

//get value by giving key value

System.out.println(students.get(1));

// get all keys

System.out.println(students.keySet());

Write code to add items to hashset

Write code to retrieve items to hasset

import java.util.HashSet;

import java.util.Set;

public class HashSetDemo {

public static void main(String[] args) {

// TODO Auto-generated method stub

//Set cannot take duplicate values

Set<String> set = new HashSet<>();

set.add("What");

set.add("are ");

set.add("you");

set.add("doing");

set.add("world");

System.out.println(set.size());

for(String s : set){

System.out.println(s);

}

set.remove("world");

System.out.println(names.size());

}

}

write code to connect to JDBC to get rows from employee table:

**public** **class** JDBCExample {

// JDBC driver name and database URL

**static** **final** String ***JDBC\_DRIVER*** = "com.mysql.jdbc.Driver";

**static** **final** String ***DB\_URL*** = "jdbc:mysql://localhost/STUDENTS";

// Database credentials

**static** **final** String ***USER*** = "username";

**static** **final** String ***PASS*** = "password";

**public** **static** **void** main(String[] args) {

Connection conn = **null**;

Statement stmt = **null**;

**try**{

Class.*forName*("com.mysql.jdbc.Driver");

System.***out***.println("Connecting to a selected database...");

conn = DriverManager.getConnection(***DB\_URL***, ***USER***, ***PASS***);

System.***out***.println("Connected database successfully...");

System.***out***.println("Creating statement...");

stmt = conn.createStatement();

String sql = "SELECT id, first, last, age FROM Employee";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next()){

**int** id = rs.getInt("id");

**int** age = rs.getInt("age");

String first = rs.getString("first");

String last = rs.getString("last");

System.***out***.print("ID: " + id);

System.***out***.print(", Age: " + age);

System.***out***.print(", First: " + first);

System.***out***.println(", Last: " + last);

}

rs.close();

}**catch**(SQLException se){

se.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}**finally**{

**try**{

**if**(stmt!=**null**)

conn.close();

}**catch**(SQLException se){

}

**try**{

**if**(conn!=**null**)

conn.close();

}**catch**(SQLException se){

se.printStackTrace();

} }

System.***out***.println("Goodbye!");

}

}

Write method to return list of rows code to loop throughs

create Employee class

public class Employee {

String fname;

String lname;

Employee(String fname,String lname){

this.fname = fname;

this.lname = lname;

}

Add employee class to list collection

create method that return list of employee collection

import java.util.ArrayList;

import java.util.List;

static List<Employee> emplist = new ArrayList<>();

public List<Employee> getEmpList(){

return emplist;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Employee e1 = new Employee("sindhu","sura");

Employee e2 = new Employee("prasu","Kanathala");

emplist.add(e1);

emplist.add(e2);

List<Employee> list = e1.getEmpList();

for(Employee e :list){

System.out.println(e.fname+" "+e.lname);

}

}

}

Difference between string, string buffer, string builder with example

String is Immutable and synchronized(Thread safe).

Ex: String str= “Hello”; hello is stored in constant String pool

String buffer is mutable and synchronized

StringBuffer str = new StringBuffer("Hello") ;

String Builder is mutable and not synchronized

StringBuilder str= new StringBuilder("Hello");