Java Concepts

1. **Steps to install java**

Type java in google and then go to the site -> After downloading java - > open exe file and run. We need to set environment variables for our system to understand where the java is present.

Setting Environment variables:

**Step:1**

Go to program files and open java -> copy the entire path -> right click on my computer and click on properties -> click on advance system settings and select environment variables -> select new in system variables, then enter variable name, paste the above copied path and click on ok

**Step:2**

Go to program files and open java -> open jdk and finally open bin folder and copy the entire path -> right click on my computer and click on properties -> click on advance system settings and select environment variables -> edit the path from system variables by adding “;” at the end and paste the above copied path -> click on ok

**--** Now, open cmd and type java -version -> then if you get the details of java then it got successfully installed.

1. **Steps to install eclipse**

Type eclipse in google and then go to the site -> choose Eclipse 64 bit if you download java 64 bit and choose Eclipse 32 bit if you download java 32 bit -> Now download and run the exe file, it will get installed -> It will show the default workspace (we can even change it to different)

1. **Steps to create workspace**

Open Eclipse and click on File -> go to Switch workspace and click on other -> you can choose different workspace and click on ok -> it will refresh and Eclipse will restart again with new workspace.

1. **Steps to create project**

Open Eclipse and click on File -> go to new and select project -> In that we can select our desired project and click on next -> give project name, choose saving location of the project if you don’t wish to save in the default one’s and click on Finish

**-->** You can see src folder and java jars which got imported. package resides in src folder and in that package classes will be there.

**-->** If you directly created a class, then it will automatically saves in default package or we can create our own package first and also desired classes in that package.

1. **create .java file/class**

After creating the project right click on it -> go to new and click on class ->select the desired package -> Enter name and check the public static void main(String[] args) if you want main method -> click on Finish

(or)

Directly right click on the desired package -> go to new and click on class -> Enter name and check the public static void main(String[] args) if you want main method -> click on Finish

1. **how to create packages and what is best way to give name**

After creating the project right click on it -> go to new and click on package -> Enter name -> click on Finish (or) If you directly created a class, then it will automatically saves in default package.

Best way to give name: Always use lowercase letters for package in order to avoid conflicts between classes and interfaces.

1. **what is main method will do?**

**Public static void main(String[] args)**

It is like an entry point for execution.

**public -** it is an access specifier where each and every class have an access.

**static -** it is given because we can invoke without the need of creating an object.

**void** - it doesn’t return any value to the caller(JVM).

**main()** - it is required for the interpreter(JVM), without this it cannot run the program but compiler can convert “.class” file without main.

**String[] args** - it can accept all types of values as arguments.

1. **creating property/data members**

There are two types of data members :

1. Non-static:

They access with respect to object name so, they are like object level data members and memory space is created each and every time whenever an object is created. They are used to store specific values.

1. static:

They access with respect to class name so, they are like class level data members and memory space is created only once irrespective of the creation of objects, They are used to store common values.

code:

public class DataMembers {

// Declaring static data members - they are fixed  
 static String FullName = "SowmyaLakkireddy";

static int rollNo = 25;

public static void main(String[] args) {

// Declaring Instance or Non-static data members - we can change the values  
 String name = "Sowmya";  
 int age = 25;  
 double height = 5.4;  
 }}

1. **what is data type and different data types**

Data type - It is a type of data/information which the variable holds.

Different data types:

1. **primitive data types:**

They are predefined by the language and specified by a keyword. They have fixed size.

Byte - made of 8bits and used to save space in Arrays, which is 4 times smaller than int.

Default value = 0, min = -128(-2^7), max = +127(+2^7-1)

Short - made of 16bit signed 2’s complement integer, which is 2 times smaller than int

Default value = 0, min = -32,768(-2^15), max = +32,767(+2^15-1)

Int - made of 32bit signed 2’s complement integer

Default value = 0, min =- 2,147,483,648(-2^31), max = 2,147,483,647(+2^31-1)

Long - made of 64bit signed 2’s complement integer

Default value = 0, min=-2^63,max=+2^63 -1

Float - It is a single-precision 32-bit IEEE 754 floating point

Default value = 0.0f

Double - It is a double-precision 64-bit IEEE 754 floating point

Default value = 0.0d

Boolean - False and true, Default value=false

Char- It a single 16-bit unicode character, min value = 0, max = 65,535

**b) Reference/Object data types:**

These represents arrays and class objects. So, whenever a new object is created then its reference will be stored in the variable instead of the object and while accessing the code these references will be helpful in locating the object/array

Default value = null

1. **What is variable?**

It is a storage location which holds some information (different data types) in the system’s memory. So, when you create a variable then it reserves some space in the memory based on the data type.

1. **creating method with void**

Void method : It doesn’t return any value

**Code:**

public class Methods {

public static void main(String[] args) {

int a = 1;

int b = 2;

Person(a, b);

}

public static void Person(int a, int b) {

int sum = a + b;

System.out.println(sum);

}

}

1. **creating variable, we can create variables inside method**

We can create non-static variables inside the method but, we cannot create static variable inside the method because they belong to the class level.

**Example:**

public class DataMembers {

// Creating static variables

static String FullName = "SowmyaLakkireddy";

static int rollNo = 25;

public static void main(String[] args) {

// Creating Non-static variables  
 String name = "Sowmya";  
 int age = 25;  
 double height = 5.4;  
 }}

1. **creating method with return data type, int/string/double/float/date**

Step 1- create a class with all methods

public class MethodsReturn {  
  
 public int add(int m, int n) {  
 return m + n;  
 }  
  
 public String names() {  
 return "sowmya";  
 }  
  
 public double substract(int m, int n) {  
 return m - n;  
 }  
  
 public float multiple(int m, int n) {  
 return m \* n;  
 }  
  
 public boolean result() {  
 return false;  
  
 }  
   
 public String date(){  
 return null;  
 }

Step2: now create main class, then create object for the above class and invoke all the methods

import java.util.Date;  
  
public class Methods {  
   
 public static void main(String[] args) {  
 // TODO Auto-generated method stub  
 int a = 1;  
 int b = 2;  
 int m = 10;  
 int n = 5;  
  
 Person(a, b);  
  
 MethodsReturn mr = new MethodsReturn();  
 Date date = new Date();  
 System.out.println(mr.add(m, n));  
 System.out.println(mr.substract(m, n));  
 System.out.println(mr.result());  
 System.out.println(mr.multiple(m, n));  
 System.out.println(date);  
 }  
  
 public static void Person(int a, int b) {  
  
 int sum = a + b;  
 System.out.println(sum);  
 }  
  
}

1. **method that will return hard coded value**

It returns hard coded value sowmya

public String names() {  
 return "sowmya";  
 }  
Output: sowmya

1. **method that will return property value**

Here the method add will return value based on the data members.

public int add(int m, int n) {  
 return m + n;  
 }  
Output: 15

**16. Create default/paramterzied constructors (overloaded constructor)**

**Constructor:**

A constructor is a special method that is used to initialize a newly created object and is called just after the memory is allocated for the object. It can be used to initialize the objects. to required, or default values at the time of object creation. It is not mandatory for the coder to write a constructor for the class.

* It adds lot on readability and usability of class.
* We can use this () to refer the same class and super () to refer parent class
* If we don’t provide constructor then default constructor will be created and it will assign the default values for the variables.
* We cannot inherit constructors from parent to child.

**Rules:**

It has the same name as the class

It should not return a value not even void

Constructors cannot be abstract, static, final or synchronized

It can have all four accessibility modifiers: private , public, protected, default

It can have parameters

It can have throws clause: we can throw exception from constructor.

It can have logic, as part of logic it can have all java legal statement except return statement with value.

**17.Why constructor overloading?**

* Constructor overloading is done to construct object in different ways.
* Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.
* useful when you want to use same class name to initialize variables of different types datatypes or different number of variables.

**Code:**

**public** **class** Constructors {

**private** String name;

**private** **int** age;

**private** **double** height;

**public** Constructors() { //default constructor

}

**public** Constructors(String name, **int** age) { //parameterized constructor with 2 parameters

**this**.name = name;

**this**.age = age;

}

**public** Constructors(String name, **int** age, **double** height) { ////parameterized constructor with 3 parameters

**this**.name = name;

**this**.age = age;

**this**.height = height;

}

**void** display(){

System.***out***.println(name + " " + age + " " + height);

}

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

Constructors d1 = **new** Constructors();

Constructors d2 = **new** Constructors("sowmya",25);

Constructors d3 = **new** Constructors("sruthi",30,5.4);

d1.display();

d2.display();

d3.display();

}

}

**Output:**

null 0 0.0

sowmya 25 0.0

sruthi 30 5.4

**Encapsulation – (hiding of data) – getter and setters**

* The whole idea behind encapsulation is to hide the implementation details from users. If a data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class. However if we setup public getter and setter methods to update (for e.g. void setSSN(int ssn))and read (for e.g. int getSSN()) the private data fields then the outside class can access those private data fields via public methods. This way data can only be accessed by public methods thus making the private fields and their implementation hidden for outside classes. That’s why encapsulation is known as data hiding.
* The reason for using getters and setters instead of making your members public is that it makes it possible to change the implementation without changing the interface.

**Code:**

**package** Introduction;

**public** **class** Encapsulation {

**public** String name;

**public** **int** age;

**public** **double** height;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **double** getHeight() {

**return** height;

}

**public** **void** setHeight(**double** height) {

**this**.height = height;

}

}

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

1. Adding items into integer array

int[] values= new int[3];  
 values[0] = 1;   
 values[1] = 2;  
 values[2] = 3;

(or)

Int[] values = {1,2,3};

1. Adding items into string array

String[] names= new String[4];  
  
 names[0] = "Sowmya";  
 names[1] = "Roja";  
 names[2] = "Pallavi";  
 names[3] = "Vrundha";

(0r)

String[] names = {“Sowmya”,”Roja”,”Pallavi”,”Vrundha”}

**19.** **write code to retrieve items from integer, string array**

Retrieving items from integer array

for (int s : values) {  
  
 System.out.println(s);  
 }

Output:

1

2

3

Retrieving items from string array

Arrays.sort(names);  
 for (String n: names) {  
 System.out.println(n);  
  
 }

Output:

Pallavi

Roja

Sowmya

Vrundha

**20.write code to add items to ArrayList collection**

ArrayList<String> animals = new ArrayList<String>();  
   
 animals.add("dog");  
 animals.add("cat");  
 animals.add("monkey");

ArrayList<Integer> values = new ArrayList<Integer>();

values.add(1);

values.add(2);

values.add(3);

values.add(4);

**21.write code to retrieve items from arraylist (using for each loop)**

System.out.println(animals);

for(String a:animals){

System.out.println(a);

}

Output:

[dog, cat, monkey]

dog  
cat  
monkey

System.out.println(values);

for(int v:values){

System.out.println(v);

}

Output:

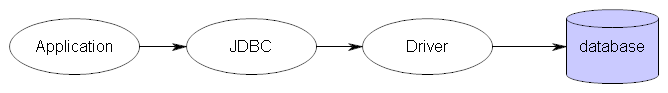
[1, 2, 3, 4]  
1  
2  
3  
4

**22.write code to connect to JDBC to get rows from employee table**

**JDBC: (Java Database Connectivity)**

* JDBC API consists of set of classes and interfaces that help us to connect with database, insert, update and delete the entries in the table.
* With this we can write SQL queries in java code.
* One must import **java.sql** package to run JDBC.

**Flow**:



Application calls JDBC Library -> JDBC loads the driver -> driver talks with database.

**Steps to follow:**

* + Load the driver
  + use DriverManager to get a Connection
  + create a Statement to query the database

3 kinds of statements –

1. **statement**- for executing simple queries without parameters and one time

executeQuery – can be used for running SQLqueries

executeUpdate – can be used for crud operations

2.**prepared statements** – executing queries with parameters many times

3.**callable statements** – executing stored procedures and accept input parameters

* + execute the statement
  + process the ResultSet
  + close the Statement
  + close the Connection

1. **Driver class:**The driver class for the mysql database is **com.mysql.jdbc.Driver**.

At first, we have to load the driver that is Class.forName**(“com.mysql.jdbc.driver”);**

1. Use DriverManager to get a connection. For this we should have
   1. **URL: (database service)**

The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo**  where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, you need to replace the sonoo with your database name.

* 1. **Username:**The default username for the mysql database is **root**.
  2. **Password:**Password is given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

1. Now, once the connection is done we should query the database (to send sql queries to database) for that we have to create a statement

Statement st = conn.createStatement();

1. Once a statement is created we should execute the statement using the statement object which is “**st**”.

st.executeQuery(query) [query = Select \* from employee]

1. Now, whatever the data which we got when we execute the query is sent to the result set so, for that we should create a **ResultSet** object and send the data which we got.

ResultSet **rs** = st.executeQuery(select \* from employee)

1. Now, we have to process the **ResultSet** for that we have many methods like first, beforeFirst, last, afterLast, next, previous. For our scenario, we have to take next method (it moves the cursor to the next row and returns false when there are no more rows )
2. We can also have ResultSet methods for getting the data like dbResultSet.getString(“column name”) – it will return the value of the column name form the current row.

(or)

dbResultSet.getString(2)- returns the value of field in column2 for the current row (indexing starts from 1).

1. Close the statement and close the connection.

**Important:** we have to download jdbc jar file because jdbc is an API and we have to import all jar files to make a connection between java application and database mysql.

**Code:**

package JDBC;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.ArrayList;

public class connectionToDatabase {

ArrayList<Employee> list = new ArrayList<Employee>();

public static void main(String[] args) {

try{

//Class.forName("com.mysql.jdbc.driver"); // loads the driver class

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/demo", "root", "root"); //creates conneection with the database

Statement st = con.createStatement();// create statement

ResultSet result = st.executeQuery("select \* from employee ");//execute the statement and return the result into the ResultSet

while(result.next())//processing the ResultSet

{

System.out.println(result);

}

result.close();

st.close();

con.close();

}catch(Exception ex){

System.out.println("Message " + ex);

}

}

}

**Add employee class to list collection**

ArrayList<Employee> list = new ArrayList<Employee>();

**Create Employee class**

**package** JDBC;

**public** **class** Employee {

**private** **int** idemployee;

**private** String Name;

**private** **int** Age;

**private** String State;

**public** Employee(**int** id, String name, **int** age, String state) {

**this**.idemployee = id;

**this**.Name = name;

**this**.Age = age;

**this**.State = state;

}

**public** **int** getId(**int** id) {

**return** idemployee;

}

**public** **void** setId(**int** idemployee) {

**this**.idemployee = idemployee;

}

**public** String getName(String name) {

**return** Name;

}

**public** **void** setName(String name) {

**this**.Name = name;

}

**public** **int** getAge(**int** age) {

**return** Age;

}

**public** **void** setAge(**int** age) {

**this**.Age = age;

}

**public** String getState(String state) {

**return** State;

}

**public** **void** setState(String state) {

**this**.State = state;

}

}