**JAVA QUESTIONS**

**Steps to INSTALL JAVA**

1. Go to google and enter JAVA JDK
2. Click on the 1st link i.e. oracle link to download java into your computer.
3. Click on “JAVA PLATFORM JDK”.
4. Now, you can find various options like windows versions and you need to download the one which suits to your computer operating system.
5. Click on the radio button “Click on Accept license agreement “and download.
6. Now, Install java in your system.
7. Double –click on executable file that you can find once download is complete.
8. Click on “Next” till you reach the “finish” option.
9. To check if java is successfully installed, go to cmd prompt and type javac and enter.
10. If it shows javac file is installed, then installation is successfully completed.

**Steps to INSTALL ECLIPSE**

1. Go to google and search for Eclipse IDE Download.
2. Click on the 1st link that shows the eclipse official website.
3. Click on download 64 bit and download.
4. Double-click on executable file that shows once the downloading is completed.
5. Click on “Eclipse IDE for java developer”.
6. It shows install option, click on that to start installation.
7. Click on “Launch”.
8. It shows workspace path where your java projects will get installed.

**Steps to setup workspace in Eclipse**

1. Click to open Eclipse application.
2. For new workspace, we need to browse and select existing folder to be our workspace and click “OK”.
3. Now, Eclipse starts and shows some icons as options.
4. Select “workbench” to start workspace.

**Steps to create project**

1. To start a new project, go to file (top left corner) and choose new option.
2. Choose an option java project and give a project name that suits your project and select “next”.
3. It shows “src” it is where our java classes will store.
4. And finally finish.

**Steps to create java file/class**

Go to file and choose “new”.

Select class and give class name

Check the box to select the main method with the “Public static void main (string args[]).

Now, new class is created**.**

**Steps to create package and what is the best way to give package name?**

Go to src and right click

New-🡪package and give package name

Com.companyname.package use

E.g.:- com.saibersys.services

**What is the use of Main method ()?**

public static void main (string args [])

A Java program needs to start its execution somewhere. A Java program starts by executing the main method of some class. You can choose the name of the class to execute, but not the name of the method. The method must always be called main. Here is how the main method declaration looks when located inside the Java class declaration from earlier:

package myjavacode;

public class MyClass {

public static void main(String[] args) {

}

}

In Java, the main method has to be called first if before any objects hence it is static.

Static🡪 because static keyword can be invoked without creating an object

Void--🡪 because void doesn’t return any values.

Public-🡪because it is a access specifier so that it can be accessed outside the class.

String args[]-🡪It acts as one of the parameters and args accepts any kind of data like string, int, char and so on.

**What is the data type and different types of data type?**

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.

There are two data types available in Java −

Primitive Data Types

Reference/Object Data Types

**Different types of data types are:**

Byte, int, char, string, Boolean, long, float, double

Eg:- int x=5;

**What is a variable?**

Variable is a kind of placeholder. A variable provides us with named storage that our programs can manipulate.

int a, b, c; // Declares three ints, a, b, and c.

int a = 10, b = 10; // Example of initialization

**Creating property/data members:**

Data members are nothing but simple variable that we declare inside a class. So they are called data members of that class.

Example: - public class Student {

int id; // these are the data members of the class student.

String name;

float gpa;

}

**Creating method with void.**

If the method is not returning any value then the method should be **void**

public void go(int x)

{

System.out.println(x);

}

**Creating variable, we can create variables inside method**

Public int display(int x, int y)--🡪here int x and int y are the passing values

{

x= 20;

y= 30;

z=x+y;

}

**Creating method with return data type, we can return int/string/double/float/date etc.**

Public int display(int x, int y)

{

x= 20;

y= 30;

int z=x+y;

return z;

}// here we have mentioned int in place of return type so, here it return integer value. Whatever datatype we are using in place of return type, we can only return those values.

**Creating method that returns hard code value**

Public String display ( )

{

return ” Hello”;

}

Here it is not using any variable but directly return hard code value.

**Creating object**

Public class Test

{

Public static void main(String args[])

{

Test T = new Test();----🡪 here object is created for the class Test

}  
}

**Create default/parameterized constructors**

A constructor with arguments is known as parameterized constructor.

class Example{

//Default constructor

Example(){

System.out.println("Default constructor");

}

/\* Parameterized constructor with

\* two integer arguments

\*/

Example(int i, int j){

System.out.print("parameterized constructor");

System.out.println(" with Two parameters");

}

/\* Parameterized constructor with

\* three integer arguments

\*/

Example(int i, int j, int k){

System.out.print("parameterized constructor");

System.out.println(" with Three parameters");

}

public static void main(String args[]){

//This will invoke default constructor

Example obj = new Example();

/\* This will invoke the constructor

\* with two arguments

\*/

Example obj2 = new Example(12, 12);

/\* This will invoke the constructor

\* with three arguments

\*/

Example obj3 = new Example(1, 2, 13);

}

}

**Method that will return property value**

**Creating method with return data type and parameter**

**Eg**: public class Method1{

Public static void main (String args[]){

Public int add(int x, inty)

{

int z= x\*y;

return z;

**Creating static property:**

static{

int x= 10;

system.out .println(x);

}

**Creating static method**

Public static void go( int x, int y)

{

int z= x + y;

}

**Create static block**

static {

// whatever code is needed for initialization goes here

System.out.println(“ x goes here”);

}

**Creating object**

public class Student{

public static void main(String args[])

{

Student s= new Student();

}

**Calling method with void**

Public class MethodCall{

Public static void main(String args[]){

MethodCall m= new MethodCall();

m.go();

}

Public void go(){

System.out.println(“Inside go”);

}

}

**Calling method with no return and parameter**

Public class Addition{

Public static void main(String args[]){

Addition.a= new Addition();

a.add();

}

public void add(){  
int x= 10;

int y= 20;

int z= x+y;

System.out.println(“the addition of x and y is:” +z);

} };

**Calling method with return and no parameter**

Public class Addition{

Public static void main(String args[]){

Addition.a= new Addition();

a.add();

}

public int add(){  
int x= 10;

int y= 20;

int z= x+y;

return z;

} };

**Calling method with return and parameter**

Public class Addition{

Public static void main(String args[]){

Addition.a= new Addition();

a.add(10,20);

}

public int add(int x, int y){  
int z= x+y;

return z;

} };

**Calling method with return and storing the return data**

**Calling static method**

Public class Addition{

Public static void main(String args[]){

Addition.a= new Addition();

a.add(10,20);

}

public static void add(int x, int y){  
int z= x+y;

System.out println(z);

} };

**Using static property: it will maintain**

**Implement method overloading**

public class DisplayOverloading

{

public void disp(char c)

{

System.*out*.println(c);

}

public void disp(char c, int num)

{

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

}

public static void main(String args[])

{

DisplayOverloading obj = new DisplayOverloading();

obj.disp('a');

obj.disp('a',10);

}

**Implement method overriding**

Public class Bike{

String color;

Public void startBike(){

System.out.println(“kick to start bike”);

}

public class Honda extends Bike{

String color;

public void tyretype(){

system.out.println(“tubeless tyres”);

}

Public void startBike(){ //here startBike method is overriding

System.out.println(“self start”);

}

**Implementing interface**

Public interface Employee{

Public void salary();

}

Public class Department implements Employee{

Public static void main(string args[]){

Employee e= new Department();

e. salary();

}

}

**How to do inheritance in java (using extend keyword)**

Public class Bike{

String color;

Public void startBike(){

System.out.println(“kick to start bike”);

}

public class Honda extends Bike{

String color;

public void tyretype(){

system.out.println(“tubeless tyres”);

}

**Create class under multiple packages**

**calling classes under different packages**

**write code to handle exceptions with try/catch/finally**

public class ExceptionHandling{

public static void main(String args[]){

try{

int a= 100/0;

} catch(exception e){

System.out.println(“….”);

}finally{

System.out.println(“finally called”);

}

}

**What is checked exception/unchecked exception?**

**Checked:** are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throwskeyword.

**Unchecked** are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.  
In Java exceptions under Errorand RuntimeExceptionclasses are unchecked exceptions, everything else under throwable is checked.

**what is final keyword, create final class, final method, final property**

. A final class cannot be sub classed

. A final method cannot be overridden.

. A final variable cannot be initialized more than once.

Example:-

Public final class Student{ // this class cannot be subclasses( extended)

Public static void main(String args[]){

Public final int number= 10;// this variable cannot be initialized once agian

}

}

**write code for creating abstract class**

// any class with abstract methods(which are declared but not defined ) will become abstract class.

public abstract class Phone{

public void call(){

system.out.println(“calling..”);

}

Public abstract class move();

Public abstract class text();

}

Public abstract class AndroidPhone extends Phone{

Public void text(

System.out.println(“text”);

}

Public abstract void move();

}

**implementing polymorphism**

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

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

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

import java.util.ArratList;

public class MyClass{

public static void main(String args[]){

ArrayList<Integer>myList=new ArrayList<Integer>(5);

myList.add(1);

myList.add(8);

}

}

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

import java.util.ArratList;

public class MyClass{

public static void main(String args[]){

ArrayList<Integer>myList=new ArrayList<Integer>(5);

myList.add(1);

myList.add(8);

} for(Integer x:myList)

System.out.println(x);

myList.remove(2);-🡪it removes element from position 2

}

**write code to add items HashMap**

import java.util.Map;

public class HashMap{

public static void main(String args[]){

map<Integer,String> map= new Hashmap<Integer,String>();

map.put(1,”A”);

map.put(2,”B”);

}

**}**

**write code to retrieve items HashMap**

import java.util.Map;

public class HashMap{

public static void main(String args[]){

map<Integer,String> map= new Hashmap<Integer,String>();

map.put(1,”A”);

map.put(2,”B”);

system.out.println(“The elements are:”+map);

map.clear();

system.out.println(“the elements are removed:”+map);

}

**}**

**Write code to add items to hashset**

Import java.util.\*;

Public class HashSet{

Public static void main(String args[]){

HashSet<String>name =new HashSet<String>();

Name.add(“mouni”);

Name.add(“mark”);

Iterator<String>itr= name.iterator();// to access these values we use iterator

While(it.hasNext());

s.o.println(itr.Next());

**}**

**}**

**Write code to retrieve items to hashset**

Import java.util.\*;

Public class HashSet{

Public static void main(String args[]){

HashSet<String>name =new HashSet<String>();

Name.add(“mouni”);

Name.add(“mark”);

s.o.println(“the elements in the set:” +name.size());

name.remove(“mark”);

s.o.println(“the elements in the set:” +name.size());

**}**

**}**

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

**Write method to return list of rows code to loop throughs**

**create Employee class**

public class Employee{

int Eid;

String Ename

---

}

**Add employee class to list collection**

**create method that return list of employee collection**

**Difference between string, string buffer, string builder with example**

**String** is *immutable*  ( once created can not be changed )object  . The object created as a String is stored in the  Constant String Pool  .   
Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously.  
String  once assigned can not be changed.  
  
String  demo = " hello " ;  
// The above object is stored in constant string pool and its value can not be modified.

**StringBuffer**is mutable means one can change the value of the object . The object created through StringBuffer is stored in the heap . StringBuffer  has the same methods as the StringBuilder , but each method in StringBuffer is synchronized that is StringBuffer is thread safe .   
  
Due to this it does not allow  two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .

StringBuffer demo1 = new StringBuffer("Hello") ;  
// The above object stored in heap and its value can be changed .

demo1=new StringBuffer("Bye");  
// Above statement is right as it modifies the value which is allowed in the StringBuffer

**StringBuilder** is same as the StringBuffer , that is it stores the object in heap and it can also be modified . The main difference between the StringBuffer and StringBuilder is that StringBuilder is also not thread safe.   
StringBuilder is fast as it is not thread safe .    
  
  
StringBuilder demo2= new StringBuilder("Hello");  
// The above object too is stored in the heap and its value can be modified  
demo2=new StringBuilder("Bye");   
// Above statement is right as it modifies the value which is allowed in the StringBuilder

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

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