JAVA

**1)What is applet?**

* *The applet program is a small java program that does not consists of main() to start the execution.Instead it consists of several methods of the Applet class which controls its life-cycle.*
* *This program is not executed using java interpreter.Instead it needs to be embedded into the webpage created using HTML and executed using the browser that supports java or using appletviewer.*
* *Applets can be used to make the webpage dynamic by adding multimedia contents.*

*Example:*

1. //First.java
2. **import** java.applet.Applet;
3. **import** java.awt.Graphics;
4. **public** **class** First **extends** Applet{
6. **public** **void** paint(Graphics g){
7. g.drawString("welcome",150,150);
8. }
10. }

//myapplet.html

1. <html>
2. <body>
3. <applet code="First.class" width="300" height="300">
4. </applet>
5. </body>
6. </html>

***2.a).What is synchronization ?***

*Synchronization:*

* *Threads use their own data and method inside their run().*
* *Strange results arise when threads try to use data and methods outside themselves.*
* *Eg: One thread trying to read a record from a file while another is still writing to the same file.*
* *Keyword ‘synchronized’ used before function or a part of a function.*

*Syntax:*

*synchronized void update(){}*

*class A extends Thread*

*{*

*public void run()*

*{*

*void update(); //Calling an outside function.*

*}*

*}*

*Class B extends Thread*

*{*

*public void run()*

*{*

*void update(); //Calling an outside function.*

*}*

*}*

*Whenever a thread has completed working with synchronized method , control transfers to next thread that is ready to use the same resource.*

***2.b)What is deadlock?***

*Deadlock:*

* *When Two or more threads are waiting to gain control of a resource And both depend on each other , then deadlock occurs.*
* *The condition on which waiting threads rely to gain control ,*

*does not happen.*

*Syntax:*

*Thread A*

*synchronized method2()*

*{*

*synchronized method1()*

*{*

*}*

*}*

*Thread B*

*synchronized method1()*

*{*

*synchronized method2()*

*{*

*}*

*}*

*To avoid deadlock condition either remove ‘synchronized ’ keyword from any function or don’t call another synchronized().*

***3.a)What is Multithreading?***

*Multithreading:*

* *It is a programming paradigm where a large program is broken down in smaller subparts(called thread) and these are executed in parallel.*
* *This feature is similar to the concept of multitasking in OS.*
* *The processor switch over the control to these threads in such a way that they appear as if they are running concurrently.*

*Thread:*

* *Each such subpart is called the thread and a thread is like a tiny program which has a starting point,it’s own body (whose execution is done consequetively) and finally it’s own end point.*
* *Using this methodology the performance of the program improves as the threads will not have to wait for the termination of another thread which takes more time for execution.(Round robin/equal time slots/time slicing/reduced average WT).*

***3.b)What are the two ways to create a thread?***

* *Extending the thread class:*

*Eg:*

*class A extends Thread*

*{public void run(){}}*

*class B extends Thread*

*{public void run(){}}*

*-Create own class and extend Thread class of java.lang package.*

*-Then override the run() method.*

* *Implementing the runnable interface:*

*Eg:*

*class A implements Runnable*

*{public void run()*

*{}}*

*class B implements Runnable*

*{public void run()*

*{}}*

*class TestRunnable*

*{*

*psvm(String args[])*

*{*

*A ob1=new A();*

*B ob2=new B();*

*Thread x=new Thread(ob1);*

*Thread y=new Thread(ob2);*

*x.start();*

*y.start();*

*}}*

*-The Runnable interface of java.lang package contains only the run() which is an abstract method(So override the run method)*

*-The method required to manage the life cycle of a thread are absent in the Runnable interface.So the child class of this Runnable interface must be set as the target object having the run for the object of the Thread class which is containing the methods required for the life cycle.*

***3.c)What do you mean by Thread priority?***

* *All the threads are given uniform time slots.*
* *But some threads may perform task that need fast attention of the processor.*
* *When we want to schedule the execution of the threads based on the tasks done by them then these threads are allotted the priorities.*
* *Methods are:*

*-setPriority()*

*-getPriority()*

* *Also there are some symbolic constants of Thread class:*

*-Thread.MIN\_PRIORITY //1*

*-Thread.NORM\_PRIORITY //5*

*-Thread.MAX\_PRIORITY //10*

***4a).What is Exception handling?***

* *Run time error:*

*Sometimes the program compile successfully creating the bytecode but may not run properly due to errors that are encountered during runtime by java interpreter,causing the program to terminate abnormally.*

* *Exception:*

*When the java interpreter encounters the run time error which leads to abnormal program termination it is known as exception.*

* *Exception handling:*

*The java interpreter displays an error message and stops normal execution when error is not caught and handled properly. So exceptions need to be handled.*

* *Exception handling mechanism:*

*If we want normal exeution of the remaining code even after exception has been encountered then this can be done by incorporating the separate error handling code called error handling mechanism.*

*TASK:-*

1. *Hit the exception //Detect*
2. *Throw the exception //Inform*
3. *Catch the exception //Receive*
4. *Handle the exception //Correct*

*Eg:*

*try{}*

*catch(Exception e){}*

***4b)What is the use of “throws” keyword?***

* *Try-Catch is explicit exception handling.*
* *Throws is implicit handling*
* *Throws keyword is used after the declaration of that method in the header line itself and then the method body is defined.*

*Eg:*

*static int Divide(int x,int y)throws ArithmeticException*

*{*

*}*

***4c).What is the use of finally keyword?***

* *The finally block consists of those statements whose execution must be done in any situation(Even when the exception is not handled by catch block).*
* *It guarantees the execution of the statement.*

*Eg:*

*Try{}*

*Catch{}*

*Finally{}*

***5a).What is interface?***

* *An interface is just like a class i.e. it is also the collection of fields and methods into a single unit.*
* *But the major difference is that*

*- all the fields in the interface are final*

*- all the methods in the interface are by default abstract methods.*

* *Since the interface consists of the abstract methods =>*

*So it must be implemented in the subclass and overriding of method must be done.*

* *The interface cannot be used to instantiate the object.*
* *MULTIPLE INHERITANCE IN JAVA CAN BE IMPLEMENTED USING INTERFACES ONLY.*
* *Can implement any no of interface but can extend only one class.*

*Eg:*

*interface Shape*

*{*

*int Calculate(int x,int y);*

*}*

*class Circle implements Shape*

*{*

*int Calculate(int x,int y){return PI\*x\*x;}*

*}*

*class Rectangle implements Shape*

*{*

*int Calculate(int x,int y){return x\*y;}*

*}*

*class Test*

*{Circle ob1;Rectangle ob2;Shape S;S=ob1;S=ob2;//Proper Syntax  
}*

***6a).What is package in java?***

*Package:*

* *A package is a collection of classes and/or interfaces.*
* *Types: Java API packages & User defined package.*

*Benefits:*

*- Reusability achieved.(No need to physically copy code)*

*-coding separated from designing.*

*-Hide the classes due to security reason(don’t make it public) .*

*-Classes contained in the package can be made unique with the classes of other package.(fully qualified path name).*

*Eg:*

*package p1;*

*public class Maths*

*{*

*public static int factorial(int N)*

*{}*

*public static int power(int N,int p)*

*{}*

*}*

*import java.util.\*;*

*import p1.\*;*

*class Test*

*{*

*…….*

*res1=Maths.factorial(N);*

*res2=Maths.power(N,p);*

*…….*

*}*

**OOPS**

**1)What is static member of a class?**

* Static field:

-A single copy of this variable is created which is shared by all the member objects.

-It can be used to keep a record count of how many objects have been created for a particular class.

* Static method:

-Use static keyword preceding normal method declaration.

-Can be accessed using class name (and not using object name).

-Static method can access only other static field/method of the class.

-Use of “super” and “this” keyword is not allowed in static methods.

Eg:

class Student

{

…

static int count=0;

…

}

static void totalStud()

{

Sopln(…+count);

}

class Test

{

Student.totalStud();

}

**2)What is method overriding?**

* When the base class and the sub class have same format of method(same: name,argument list,return type).
* In such a situation if we create an object of the subclass and try to invoke the method then
* Everytime the method of the child class gets invoked and dominates the base(parent) class.
* This process is called method overriding.

USEFUL:

-In situation where

-some child class of a particular parent class give response to the same format of the method(but has different behaviour)

-and some child class uses method of the base class in the original form.

Eg:

class Rectangle

{

void compute(int l,int b){Sop(l\*b);}

}

class Cuboid extends Rectangle

{

void compute(int l,int b){Sop(l\*b\*h);}

}

class Test

{//Use proper syntax

Rectangle ob1; ob1.compute(); //Child class called

Cuboid ob2; ob2.compute(); //Child class called

}

NOTE:If you want to call the parent class then use super.method\_name to call method inside child class.

**3).What do you mean by final method, final field, final class?**

* final method: (No Overriding)

-Declare method in parent class as final method.

-This protects the parent class from being overridden in the child class.

class Rectangle

{

final void compute(int l,int b){Sop(l\*b);}

}

class Cuboid extends Rectangle

{

void compute(int l,int b){} //Invalid

void calc(int l,int b){} //Valid

}

* final fields: (Constant)

-Fields of the class are declared using final keyword.

-Final fields are those whose value cannot be changed in any way by any method.

Eg:

class Circle

{

int Radius;

final double PI=3.14;

…..}

* final class: (no inheritance)

-To protect our class from being inherited,

-We declare a class as final class.

Eg:

final class School{}

class Student extends School{} //Can’t be inherited

**4)What is Abstract method and class?**

-Stop overriding: use final keyword.

-Compulsory overriding: base class method declared as abstract.

-Abstract method don’t have an body of itself.(The body must be overridden or defined by its child class).

-Abstract class can never be used to instantiate the object.(Only serves as a base class and provides its members to its child class for reusability.)

-A “static” and “constructor” method cannot be declared as abstract method.

Eg:

abstract class **Shape**

{

**abstract** double Calculate(double x,double y);

final double PI=3.14;

}

class **Circle** extends Shape

{

double Calculate(double x,double y)

{double ar; ar=PI\*x\*x; return ar;} //definition

}

class **Rectangle** extends Shape

{

double Calculate(double x,double y)

{return x\*y;}

}

class Test

{…

Circle ob1; Rectangle ob2; Shape s;//Use proper definition.

s=ob1; s=ob2;}