S.Y. B.Sc. (IT) : Sem. IV

Core Java

Time : 2½ Hrs.] Prelim Question Paper Solution **[Marks : 75**

Q.1 Attempt any THREE of the following : [15]

Q.1(a) Write short note on History of Java language. [5]

Ans.: The history of java starts from Green Team. Java team members (also known as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

For the green team members, it was an advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

James Gosling, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

Originally designed for small, embedded systems in electronic appliances like set-top boxes.

Firstly, it was called **"Greentalk"** by James Gosling.

After that, it was called **Oak** and was developed as a part of the Green project.

Why Oak? Oak is a symbol of strength and choosen as a national tree of many countries like U.S.A., France, Germany, Romania etc.

In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Why had they choosen java name for java language? The team gathered to choose a new name. The suggested words were "dynamic", "revolutionary", "Silk", "jolt", "DNA" etc. They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say.

According to James Gosling "Java was one of the top choices along with **Silk**". Since java was so unique, most of the team members preferred java.

Java is an island of Indonesia where first coffee was produced (called java coffee).

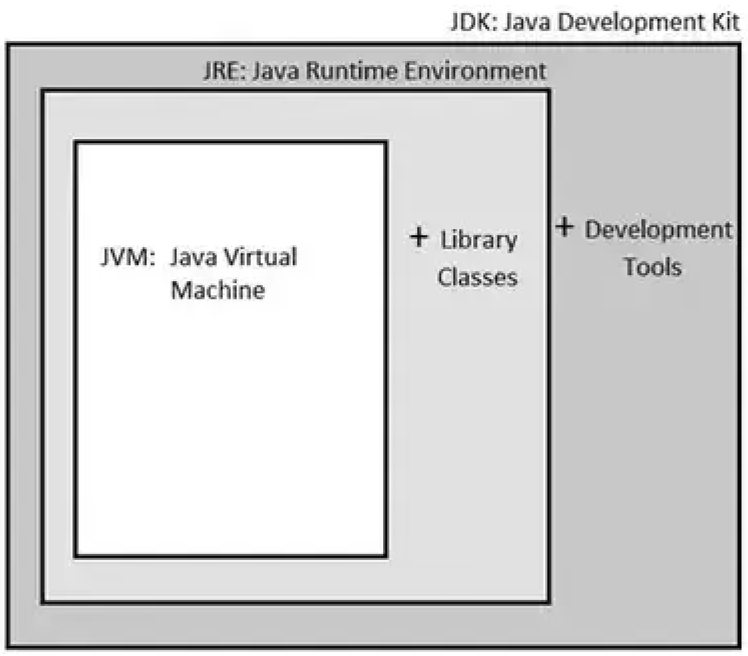
Notice that Java is just a name not an acronym.

Originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.

In 1995, Time magazine called **Java one of the Ten Best Products of 1995**.

JDK 1.0 released in(January 23, 1996).

Q.1(b) Explain difference between JDK, JRE and JVM. [5]
Ans.:


JDK: Java Development Kit
JRE: Java Runtime Environment
JVM: Java Virtual Machine
+ Library Classes
+ Development Tools

JDK = JRE + Development Tools
JRE = JVM + Library Classes

JAVA DEVELOPMENT KIT
The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

JDK is a kit(or package) which includes two things

1. Development Tools(to provide an environment to develop your java programs)
2. JRE (to execute your java program).

Note : JDK is only used by Java Developers.

JAVA RUNTIME ENVIRONMENT
JRE stands for **"Java Runtime Environment"** and may also be written as **"Java RTE."** The Java Runtime Environment provides the minimum requirements for executing a Java application; it consists of the *Java Virtual Machine (JVM), core classes*, and *supporting files*.

Java Runtime Environment (to say JRE) is an installation package which provides environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

JAVA VIRTUAL MACHINE
It is:

A **specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.

An **implementation** is a computer program that meets the requirements of the JVM specification

Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

Java Virtual machine(JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.

Q.1(c) Explain the features of Java. **[5]**

Ans.: **Features of Java Programming Language**

Simple :
- Java is Easy to write and more readable and eye catching.
- Java has a concise, cohesive set of features that makes it easy to learn and use.
- Most of the concepts are drew from C++ thus making Java learning simpler.

Secure :
- Java program cannot harm other system thus making it secure.
- Java provides a secure means of creating Internet applications.
- Java provides secure way to access web applications.

Portable :
- Java programs can execute in any environment for which there is a Java run-time system.(JVM)
- Java programs can be run on any platform (Linux,Window,Mac)
- Java programs can be transferred over world wide web (e.g applets)

Object-oriented :
- Java programming is object-oriented programming language.
- Like C++ java provides most of the object oriented features.
- Java is pure OOP. Language. (while C++ is semi object oriented)

Robust :
- Java encourages error-free programming by being strictly typed and performing run-time checks.

Multithreaded :
- Java provides integrated support for multithreaded programming.

Architecture-neutral :
- Java is not tied to a specific machine or operating system architecture.
- Machine Independent i.e Java is independent of hardware .

Interpreted :
- Java supports cross-platform code through the use of Java bytecode.
- Bytecode can be interpreted on any platform by JVM.

High performance :
- Bytecodes are highly optimized.
- JVM can executed them much faster .

Distributed :
- Java was designed with the distributed environment.
- Java can be transmit,run over internet.

Dynamic :
- Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time.

Q.1(d) Explain any five String methods. **[5]**

Ans.: **String Length**

Methods used to obtain information about an object are known as **accessor methods**. One accessor method that you can use with strings is the length() method, which returns the number of characters contained in the string object.

```
Hint length ()
String s = new String ("Hello").,
int L = s . lengths () ;
Sop ("Length of "+ s +" is " + d);
```

Concatenating Strings

The String class includes a method for concatenating two strings –
string1.concat(string2);
This returns a new string that is string1 with string2 added to it at the end.
Program to concentrate two strings :

```
Class concat
{ psum ( ____ )
{
String tech – "Java" ;
 String author = "Author Sum Microsystems";
Sopln ("Kabgyage ; " + tech + author);
Sopln (tech . concat (author));
}
}
```

O/P :

```
Language : Java Author : Sun Microsystems
Java Author : San Microsystems.
```

Java String toUpperCase() and toLowerCase() method

The java string toUpperCase() method converts this string into uppercase letter and string toLowerCase() method into lowercase letter.

WAP to convert user entered string to UpperCase

```
import java . io. A ;
Class Uppercase
{psum ( ____ ) throws IoException
{ String str ,
BR br = new BR   ( —||— ) ;
Sopln ("Enter a string;"). ,
str = br . readLine ( ) ;
str = str . to UpperCase ( ) ;
Sopln (str) ;
}
}
```

O/P : Enter a string . 'Hi' how are you
HI HOW ARE YOU

Java String trim() method

The string trim() method eliminates white spaces before and after string.

```
1.  String s = " Sachin  ";
2.  System.out.println(s);//  Sachin
3.  System.out.println(s.trim());//Sachin
```

Output:

```
  Sachin
Sachin
```

Java String startsWith() and endsWith() method

```
1.  String s = "Sachin";
2.  System.out.println(s.startsWith("Sa"));//true
3.  System.out.println(s.endsWith("n"));//true
```

Java String charAt() method
The string charAt() method returns a character at specified index.
String s = "Sachin";
1. System.out.println(s.charAt(0));//S
2. System.out.println(s.charAt(3));//h

Java String replace() method
The string replace() method replaces all occurrence of first sequence of character with second sequence of character.
1. String s1 = "Java is a programming language. Java is a platform. Java is an Island.";
2. String replaceString = s1.replace("Java","Kava"); //replaces all occurrences of "Java" to "kava"
System.out.println(replaceString);

Q.1(e) Write short note on Autoboxing and Unboxing. **[5]**

Ans.: **Autoboxing:** Converting a primitive value into an object of the corresponding wrapper class is called autoboxing. For example, converting int to Integer class. The Java compiler applies autoboxing when a primitive value is:
- Passed as a parameter to a method that **expects an object** of the corresponding wrapper class.
- Assigned to a variable of the corresponding **wrapper class**.

Unboxing: Converting an object of a wrapper type to its corresponding primitive value is called unboxing. For example conversion of Integer to int. The Java compiler applies unboxing when an object of a wrapper class is:
- Passed as a parameter to a method that **expects a value** of the corresponding primitive type.
- Assigned to a variable of the corresponding **primitive type**.

The following table lists the primitive types and their corresponding wrapper classes, which are used by the Java compiler for autoboxing and unboxing :

Primitive Type	Wrapper Class
Boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

```
// Java program to illustrate the concept
// of Autoboxing and Unboxing
    import java.io.*;

class GFG
{
    public static void main (String[] args)
{
    // creating an Integer Object
    // with value 10.
```

```
        Integer i = new Integer(10);
    // unboxing the Object
    int i1 = i;

    System.out.println("Value of i: " + i);
    System.out.println("Value of i1: " + i1);

    //Autoboxing of char
    Character gfg = 'a';

    // Auto-unboxing of Character
    char ch = gfg;
    System.out.println("Value of ch: " + ch);
    System.out.println("Value of gfg: " + gfg);

        }
}
```

Output:

```
Value of i: 10
Value of i1: 10
Value of ch: a
Value of gfg: a
```

Q.1(f) How do you compile and interpret Java program? [5]

Ans.: To execute your first Java program, follow the instructions below :

1. Proceed only if you have successfully installed and configured your system for Java as discussed here.
2. Open your preferred text editor — this is the editor you set while installing the Java platform.
   For example, Notepad or Notepad++ on Windows; Gedit, Kate or SciTE on Linux; or, XCode on Mac OS, etc.
3. Write the following lines of code in a new text document:
   ```
   public class HelloWorld {
   public static void main(String[] args) {
           System.out.println("Hello World!");
       }
   }
   ```
4. Save the file as HelloWorld.java — the name of your file should be the same as the name of your class definition and followed by the .java extension. This name is *case-sensitive*, which means you need to capitalize the precise letters that were capitalized in the name for the class definition.
5. Next, open your preferred command-line application.
   For example, Command Prompt on Windows; and, Terminal on Linux and Mac OS.
6. Compile the Java source file using the following command which you can copy and paste in if you want:
   ```
   >>javac HelloWorld.java
   ```
7. Once the compiler returns to the prompt, run the application using the following command:
   ```
   >> java HelloWorld
   ```
8. The above command should result in your command-line application displaying the following result:

   Output
   ```
   Hello World!
   ```

Q.2 Attempt any THREE of the following: [15]

Q.2(a) Write a short note on for each statement with an example. [5]

Ans.: The for-each loop introduced in Java5. It is mainly used to traverse array or collection elements. The advantage of for-each loop is that it eliminates the possibility of bugs and makes the code more readable.

Advantage of for-each loop:

It makes the code more readable.

It elimnates the possibility of programming errors.

Syntax of for-each loop:

```
for(data_type variable : array | collection){}
```

Simple Example of for-each loop for traversing the array elements:

```
class ForEachExample1{
public static void main(String args[]){
int arr[]={12,13,14,44};

for(int i:arr){
System.out.println(i);
  }

 }
}
```

Output:

```
12
13
14
44
```

Q.2(b) Write a short note on Garbage Collection in Java. [5]

Ans.: Java garbage collection is the process by which Java programs perform automatic memory management. Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short. When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

Advantage of Garbage Collection

It makes java memory efficient because garbage collector removes the unreferenced objects from heap memory.

It is automatically done by the garbage collector(a part of JVM) so we don't need to make extra efforts.

How can an object be unreferenced?

There are many ways:

By nulling the reference

By assigning a reference to another

By annonymous object etc.

Q.2(c) What is Method overloading? Explain with the help of a program. [5]

Ans.:
- In same class, defining 2 or more methods with same name
- This is possible, as long as their parameter declarations are different.
- This is called Method Overloading.
- Method Overloading is one way of how Java implements polymorphism.

A class may have as many methods with the same name based on the following three rules.

At least one of the following rules must be true for successful overloading of method.

1. It should have different no. of parameters.
2. If no. of parameters are same, the datatype of parameters should be different.
3. If datatypes are also same, atleast the sequence of parameters should be different otherwise it is a compilation error.

```
class OverloadDemo
{
    void test ( )
    {Sopln("a =" + a) ;
    }
void test (int, a, int b)
{
  Sopln ({"a and b :" + a + "   " + b) ;
}
double test (double a)
{
  Sopln ("double a : " + a) ; return a + a ;
}
class Overload  { psum ( ____ )
{   OverloadDemo ob = new OverloadDemo ( ) ;
    Double result ;
|| call all versions of test ( )
ob . test ( ) ;
ob . test (10) ; ob.test (10, 20) ;
result = ob . test (123.25) ;
Sopln ("Result of ob.test (123.25):" + result);
}
}
```

Output

```
No parameters
a = 10
a and b : 10  20
double a = 123.25
Result of ob.test (123.25) : 15190.5625
```

Q.2(d) Write a short note on Object Oriented Programming. [5]

Ans.: Object Oriented programming is a programming style which is associated with the concepts like class, object, Inheritance, Encapsulation, Abstraction, Polymorphism. Most popular programming languages like Java, C++, C#, Ruby, etc. follow an object oriented programming paradigm.

An object-based application in Java is based on declaring classes, creating objects from them and interacting between these objects.

Below are core concepts of Object oriented Programming in the following sequence:

Inheritance: In OOP, computer programs are designed in such a way where everything is an object that interact with one another. Inheritance is one such concept where the properties of one class can be inherited by the other. It helps to reuse the code and establish a relationship between different classes.

Encapsulation: Encapsulation is a mechanism where you bind your data and code together as a single unit. It also means to hide your data in order to make it safe from any modification. Through encapsulation the methods and variables of a class are well hidden and safe.

Abstraction: Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user.

Polymorphism: Polymorphism means taking many forms, where 'poly' means many and 'morph' means forms. It is the ability of a variable, function or object to take on multiple forms. In other words, polymorphism allows you define one interface or method and have multiple implementations.

Q.2(e) Explain types of constructors in Java. [5]

Ans.: Constructor: is a special method which is used to initialize object. Constructor similar to the methods declarations. This will be invoked during object initialization.

Constructor types:

Constructor are defined as two types.

1) Default parameter(no-argument or zero Parameterized constructor)
2) Parameterized constructor

1) Default parameter:

Default parameter is nothing but a zero argument parameter. This is invoked whenever we create object.

ex:

```
public class Chair {
public Chair() {
System.out.println("Constructor is invoked.");
 }
public static void main(String[] args) {
Chair chair1 = new Chair();
Chair chair2 = new Chair();
    }
}
```

Output:

```
Constructor is invoked.
Constructor is invoked.
```

Parameterized Constructor:

A constructor which has parameters is called "Parameterized Constructor". We can have multiple constructors in one class. All constructors names should be same as class name. This can be overloaded.

This is to initialize the object with different values for different objects.

ex:

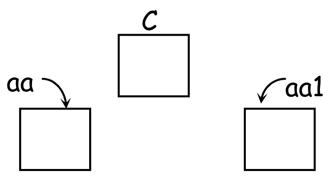
```
public class Chair {
    public Chair(String name) {
        System.out.println("Parameterized Constructor is invoked.");
    }
public static void main(String[] args) {
    Chair chair1 = new Chair("Chair 1");
    Chair chair2 = new Chair("Chair 2");
    }
}
```

Output:

```
Parameterized Constructor is invoked.
Parameterized Constructor is invoked.
```

Q.2(f) Write short note on Static variable in Java. [5]

Ans.:
1. When object of class is created, it contains only list of non–static instance variables.
2. But if the instance variable is declared as static, only one copy of static instance variable exists in the memory, every object of the class shares the same copy of static instance variable.
3. To access non–static instance variable and method of the class, we require object of the same class.
   Without object, these data are not accessible.

Eg. :
```
class A
{int a, b ;
static int c ;
A ( )
     { a = 10, b = 20 ;
       c = a + b ;
       sopln("Default constructor of A") ;
     }
A (int a1 , int = b1)
     { a = a1;
       b = b1 ;
       c = a + b ;
       sopln ("Two integer parameter constructor of A") ;
     }
void show ( )
     { sopln ("a = " + a) ;
       sopln ("b = " + b) ;
     sopln ("c = " + c) ;
     }
Psvn ( _____ )
     { A aa = new A ( ) ;
       aa . show ( ) ; → 10 20 30
       A aa1 = new A ( 100, 200) ;
       aa1 . show ( ) ; 100 200 2000
       aa . show ( ) ; 10 20 20000
     }
}
```
Output:
```
10 20 30
100 200 2000
10 20 20000
```


Q.3 Attempt any THREE of the following: [15]

Q.3(a) Define Inheritance. Explain its types. [5]

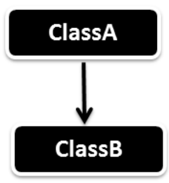
Ans.: Inheritance refers to a feature of Java programming that lets you create classes that are derived from other classes. A class that's based on another class inherits the other class. The class that is inherited is the parent class, the base class, or the superclass.

Below are the different types of inheritance which is supported by Java.
- Single Inheritance
- Multiple Inheritance (Through Interface)
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance (Through Interface)

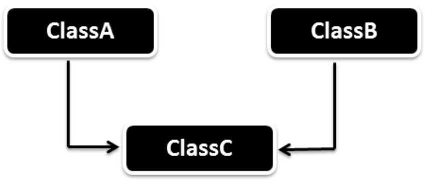
Types of Inheritance in Java – Single,Multiple,Multilevel,Hierarchical & Hybrid
August 17, 2015 by javainterviewpoint 19 Comments

Below are the different types of inheritance which is supported by Java.
- Single Inheritance
- Multiple Inheritance (Through Interface)
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance (Through Interface)

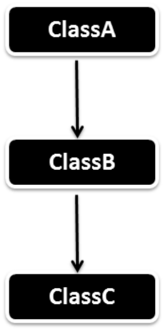
1. Single Inheritance in Java
   Single Inheritance is the simple inheritance of all, When a class extends another class(Only one class) then we call it as Single inheritance. The below diagram represents the single inheritance in java where Class B extends only one class Class A. Here Class B will be the Sub class and Class Awill be one and only Super class.



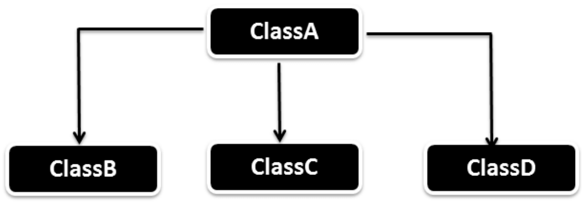
2. Multiple Inheritance in Java
   Multiple Inheritance is nothing but one class extending more than one class. Multiple Inheritance is basically not supported by many Object Oriented Programming languages such as Java, Small Talk, C# etc.. (C++ Supports Multiple Inheritance). As the Child class has to manage the dependency of more than one Parent class. But you can achieve multiple inheritance in Java using Interfaces.



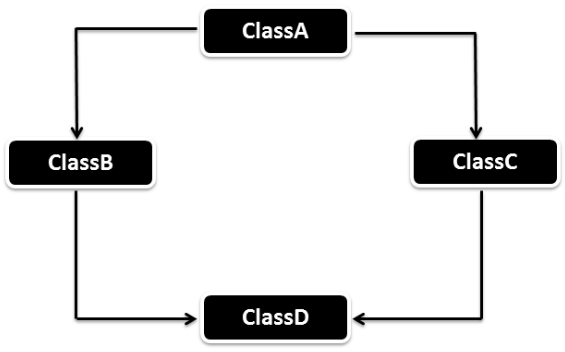
3. Multilevel Inheritance in Java
   In Multilevel Inheritance a derived class will be inheriting a parent class and as well as the derived class act as the parent class to other class. As seen in the below diagram. ClassBinherits the property of ClassA and again ClassB act as a parent for ClassC. In Short Class Aparent for ClassB and ClassB parent for ClassC.



4. Hierarchical Inheritance in Java
   In Hierarchical inheritance one parent class will be inherited by many sub classes. As per the below example ClassA will be inherited by ClassB, ClassC and ClassD. ClassA will be acting as a parent class for ClassB, ClassC and ClassD.



5. Hybrid Inheritance in Java
   Hybrid Inheritance is the combination of both Single and Multiple Inheritance. Again Hybrid inheritance is also not directly supported in Java only through interface we can achieve this. Flow diagram of the Hybrid inheritance will look like below. As you can ClassA will be acting as the Parent class for ClassB & ClassC and ClassB & ClassC will be acting as Parent for ClassD.



Q.3(b) Write a program to demonstrate Multiple Inheritance. [5]

Ans.:

```
interface vehicleone{
    int  speed=90;
    public void distance();
}

interface vehicletwo{
    int distance=100;
    public void speed();
}

class Vehicle  implements vehicleone,vehicletwo{
    public void distance(){
        int  distance=speed*100;
        System.out.println("distance travelled is "+distance);
    }
    public void speed(){
        int speed=distance/100;
    }
}

class MultipleInheritanceUsingInterface{
    public static void main(String args[]){
        System.out.println("Vehicle");
        obj.distance();
        obj.speed();
    }
}
```

Output is :

```
distance travelled is 9000
```

Q.3(c) What are packages? Give the advantages of Packages. [5]

Ans.: Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. Packages are used for:

- Preventing naming conflicts. For example there can be two classes with name Employee in two packages, college.staff.cse.Employee and college.staff.ee.Employee
- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier.
- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.
- Packages can be considered as data encapsulation (or data-hiding).

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
2) Java package provides access protection.
3) Java package removes naming collision.
4) Packages provide reusability of code .
5) To bundle classes and interfaces.
6) We can crate our own Package or extend already available Package.

Q.3(d) What is an Abstract class? Explain with a suitable java code. [5]

Ans.: A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

A class which contains the abstract keyword in its declaration is known as abstract class.

Abstract classes may or may not contain abstract methods,

i.e., methods without body (public void get();)

But, if a class has at least one abstract method, then the class mustbe declared abstract.

If a class is declared abstract, it cannot be instantiated.

To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.

If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Ex:

```
// A simple demonstration of abstract.
Abstract class A {
Abstract void callme ( ) ;
// concrete methods are still allowed in abstract classes
void callmetoo ( ) {
System . out . println ("This is a concrete method.") ;
}
}
class B extends A {
void callme ( ) {
System . out . println ("B's implementation of callme.") ;
}
}
class AbstractDemo {
public static void main (String args [ ] ) {
B b = new B ( ) ;
b . callme ( ) ;
b . callmetoo ( ) ;
}
}
```

Q.3(e) Write short note on "this" keyword in Java. [5]

Ans.: Keyword THIS is a reference variable in Java that refers to the current object.

The various usages of 'THIS' keyword in Java are as follows:

- It can be used to refer instance variable of current class
- It can be used to invoke or initiate current class constructor
- It can be passed as an argument in the method call

Ex:

```
//Java code for using 'this' keyword to
//refer current class instance variables
class Test
{
    int a;
    int b;

    // Parameterized constructor
    Test(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    void display()
    {
        //Displaying value of variables a and b
        System.out.println("a = " + a + " b = " + b);
    }

    public static void main(String[] args)
    {
        Test object = new Test(10, 20);
        object.display();
    }
}
Output:
a = 10 b = 20
```

Q.3(f) Explain use of "super" as a constructor in Java. **[5]**

Ans.: The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
class Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
super();
System.out.println("dog is created");
}
}
class TestSuper3{
public static void main(String args[]){
Dog d=new Dog();
}}
```

Output:

```
animal is created
dog is created
```

Q.4 Attempt any THREE of the following: **[15]**

Q.4(a) Write short note on Vectors. **[5]**

Ans.: **Vector** implements a dynamic array. It is similar to **ArrayList**, but with two differences: **Vector** is synchronized, and it contains many legacy methods that are not part of the collections framework.
Here are the **Vector** constructors:

```
Vector( )
Vector(int size)
Vector(int size, int incr)
```

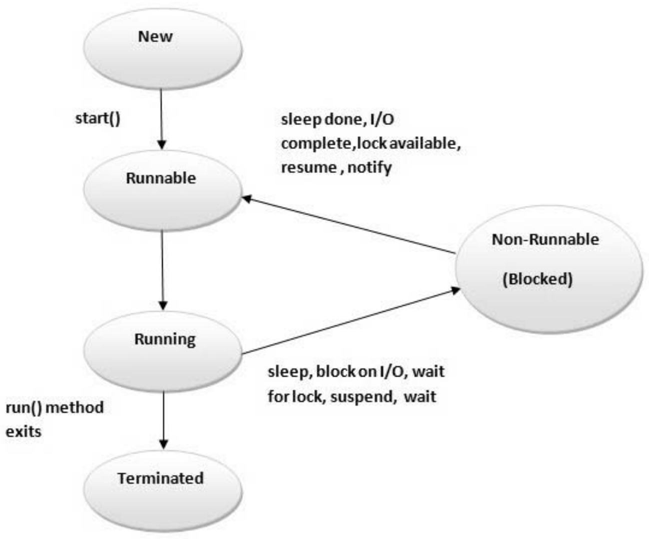
The first form creates a default vector, which has an initial size of 10. The second form creates a vector whose initial capacity is specified by *size.* The third form creates a vector whose initial capacity is specified by *size* and whose increment is specified by *incr.* The increment specifies the number of elements to allocate each time that a vector is resized upward.

All vectors start with an initial capacity. After this initial capacity is reached, the next time that you attempt to store an object in the vector, the vector automatically allocates space for that object plus extra room for additional objects. By allocating more than just the required memory, the vector reduces the number of allocations that must take place. This reduction is important, because allocations are costly in terms of time. The amount of extra space allocated during each reallocation is determined by the increment that you specify when you create the vector. If you don't specify an increment, the vector's size is doubled by each allocation cycle.

Q.4(b) Explain Thread life cycle. [5]

Ans.: A thread can be in one of the five states. According to sun, there is only 4 states in thread life cycle in java new, runnable, non-runnable and terminated. There is no running state.
But for better understanding the threads, we are explaining it in the 5 states.
The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

- New
- Runnable
- Running
- Non-Runnable (Blocked)
- Terminated


New
start()
Runnable
sleep done, I/O complete, lock available, resume, notify
Non-Runnable (Blocked)
Running
sleep, block on I/O, wait for lock, suspend, wait
run() method exits
Terminated

1) New : The thread is in new state if you create an instance of Thread class but before the invocation of start() method.
2) Runnable : The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.
3) Running : The thread is in running state if the thread scheduler has selected it.
4) Non-Runnable (Blocked) : This is the state when the thread is still alive, but is currently not eligible to run.
5) Terminated : A thread is in terminated or dead state when its run() method exits.

Q.4(c) Write a program to accept two numbers from the user and perform division of them. Use multiple try-catch block to catch Arithmetic Exception, NumberFormatException, etc. [5]

Ans.:
```
import java.io. * ;
Class Divide {
public static void main (String args []) throw IOException {
int a = 0, b = 0, res ;
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
String str,
System.out.println ("Enter two numbers :");
try
{
    str = br . readLine ();
    a = Integer. parseInt(str);
    str = br.readLine ();
    b = Integer . parseInt (str);
    res = a/b ;
    System.out.println ("The Quotient=" +res);
}
catch(ArithmeticException ae)
{
    System.out.println("Exception has occurred. You have entered the divisor as zero");
}
catch (NumberFormatException ne)
{
System.out.println ("Invalid number");
}
}
```

Output 1 :

```
Enter two numbers :
5
0
Exception has occurred. You have entered the divisor as zero.
```

Output 2 :

```
Enter two numbers :
4
a
Invalid number
```

Output 3 :

```
Enter two numbers :
5
3
The Quotient = 1
```

Q.4(d) What are Checked and Unchecked Exceptions in Java. [5]

Ans.: Checked Exceptions All checked exceptions are subclasses of class 'Exception'. Checked Exceptions make the programmer to handle the exception that may be thrown. Eg: IOException caused because of readLine() method is a checked exception. There are two ways of dealing with an exception:

1. Indicate that the method throws an exception or
2. Method must catch the exception and take appropriate action using the try catch block.

ex:

Exception	**Meaning**
ClassNotFoundException	Class not found.
CloneNotSupportedException	Attempt to clone an object that does not implement the Cloneable interface.
IllegalAccessException	Access to a class is denied.

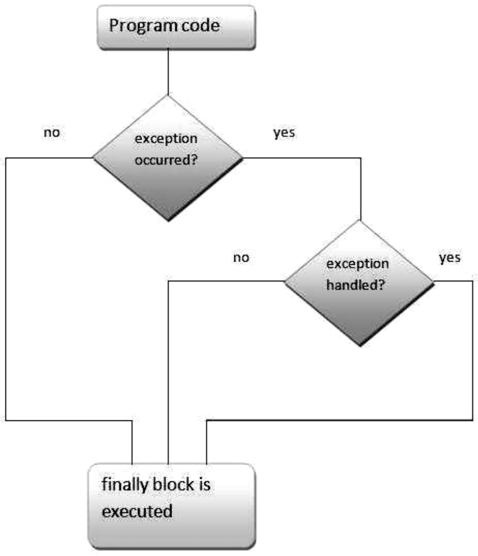
Unchecked Exceptions Unchecked Exceptions are Runtime Exception and its subclasses. These are not subclass of the class Exception. The compiler doesn't check for the unchecked exceptions and hence the compiler doesn't make the compiler handle them. Infact, the programmers may not even know that such an exception would be thrown.

ex:

Exception	**Meaning**
ArithmeticException	Arithmetic error, such as divide–by–zero
ArrayIndexOutOfBoundsException	Array index is out–of–bounds
ArrayStoreException	Assignment to an array element of an incompatible type.

Q.4(e) Explain use of "finally" block in exception handling. [5]

Ans.: When exceptions are thrown, execution in a method takes a rather abrupt, nonlinear path that alters the normal flow through the method. Depending upon how the method is coded, it is even possible for an exception to cause the method to return prematurely. This could be a problem in some methods. For example, if a method opens a file upon entry and closes it upon exit, then you will not want the code that closes the file to be bypassed by the exception-handling mechanism. The finally keyword is designed to address this contingency.



Finally creates a block of code that will be executed after a try/catch block has completed and before the code following the try/catch block. The finally block will execute whether or not an exception is thrown. If an exception is thrown, the finally block will execute even if no catch statement matches the exception. Any time a method is about to return to the caller from inside a try/catch block, via an uncaught exception or an explicit return statement, the finally clause is also executed just before the method returns. This can be useful for closing file handles and freeing up any other resources that might have been allocated at the beginning of a method with the intent of disposing of them before returning. The finally clause is optional. However, each try statement requires at least one catch or a finally clause.

Ex.:
```
class TestFinallyBlock{
    public static void main(String args[])
{
    try{
    int data=25/5;
    System.out.println(data);
     }
    catch(NullPointerException e){System.out.println(e);}
    finally{System.out.println("finally block is always executed");}
    System.out.println("rest of the code...");
    }
}
```

Output:
```
finally block is always executed
        rest of the code...
```

Q.4(f) Write a Java program to implement Multithreading. [5]

Ans.: Write a program demonstrate the making of a thread to print numbers from 1 to 10 by extending the "Thread" class.
```
class Numbers extends Thread
{
    public void run ( )
    {
        int i;
        for(i=1;i<=10;i++)
```

```
            {
                System.out.println(i);
            }
        }
    }
    class Main
    {
        public static void main(String args[])
        {
            Numbers n=new Numbers( );
            Thread t1=new Thread(n);
            t1.start( );
        }
    }
```

Output :

```
1

2

3

4

5

6

7

8

9

10
```

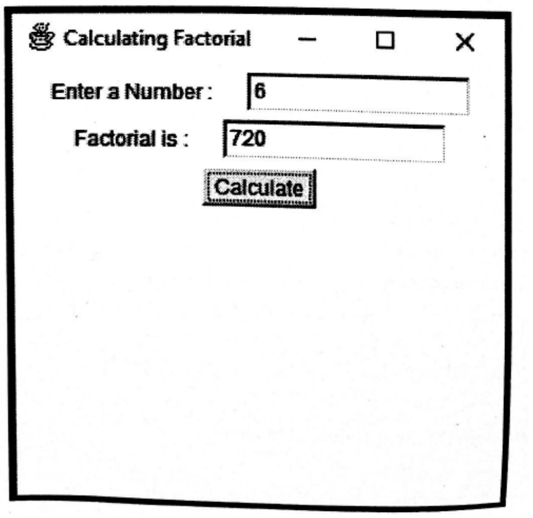
Q.5 Attempt any THREE of the following: [15]

Q.5(a) Design an AWT program to print the factorial of an input value. [5]

Ans.:

```
import java.awt.* ;
import java.awt.event. *;

class FactorialGUI extends Frame implements ActionListener
{
    TextField tf1, tf2 ;
    public FactorialGUI( )
    {

        setLayout(new  FlowLayout ( ));

        Label lb1 = new Label ("Enter a Number :");
        Label lb2 = new Label("Factorial is : ");

        tf1 = new TextField(15) ;
        tf2 = new TextField(15) ;

        Button btnl = new Button ("Calculate") ;

        add(lb1); add(tf1);
        add(lb2) ; add(tf2);
        add(btn1) ;
        btnl.addActionListner(this);
}
public static void main (String ar[ ])
{
    FactorialGUI fr = new FactorialGUI( );
    fr.setSize(300, 300);
    fr.setTitle("Calculating Factorial");
    fr.setVisible(true);
}

Public void actionPerformed(ActionEvent ae)
{

    Int n,f=1, i;

    n = Integer.parseInt(tf1.getText( ));
    for(i=1; i < = n; i ++)
    {
        f = f * i;
    }

    tf2.setText("" + f) ;
    }
}
```

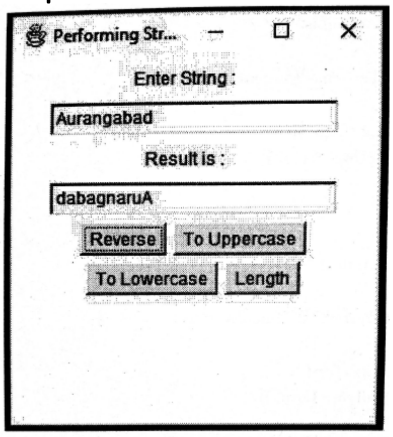
Output :



Q.5(b) Design an AWT program to perform various string operations like string reverse, string concatenation, etc. **[5]**

Ans.:
```
import java.awt.*;
import java.awt.event.*;

class StringOperationsGUI extends Frame
{
    TextField tf1, tf2;
    public StringOperationsGUI( )
    {
            setLayout(newFlowLayout( )) ;

            Label lb1 = new Label("Enter String :");
            Label lb2 = new Label("Result is :");
            tf1 = new TextField(25);
            tf2 = new TestField(25);
            Button btn1 = new Button("Reverse");
            Button btn2 = new Button("To Uppercase");
            Button btn3 = new Button("To Lowercase");
            Button btn4 = new Button("Length");

            add(lb1); add(tf1);
            add(lb2); add(tf2);
            add(btn1); add(btn2); add(btn3); add(btn4);

            btn1.addActionListener(new Inner1( ));
            btn2.addActionListener(new Inner2( ));
            btn3.addActionListener(new Inner3( ));
            btn4.addActionListener(new Inner4( ));
    }
    class Inner1 implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            StringBuffer sb = new StringBuffer(tf1.getText( ));
            tf2.setText(sb.reverse( ).toString( ));

        }
    }
    class Inner2 implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            String s = tf1.getText( );
            tf2.setText(s.toUpperCase( ));

        }
    }
    class Inner3 implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            String s = tf1.getText( );
            tf2.setText(s.toLowerCase( ));

        }
    }
    class Inner4 implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            String s = tf1.getText( );
            tf2.setText("" + s.length( ));

        }
    }
    public static void main(String ar[])
    {
        StringOperationsGUI fr = new StringOperationsGUI( );
        fr.setSize(300,300);
        fr.setTitle("Performing String operations");
        fr.setVisible(true);
    }
}
```

Output :



Q.5(c) What are Applets in Java. **[5]**

Ans.: Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

Advantage of Applet

There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

- Plugin is required at client browser to execute applet.

Lifecycle of Java Applet

- Applet is initialized.
- Applet is started.
- Applet is painted.
- Applet is stopped.
- Applet is destroyed.

Q.5(d) What are Layouts? Explain GridLayout with an example. **[5]**

Ans.: The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers.

- GridLayout position the components in two dimensional grids.

Constructors

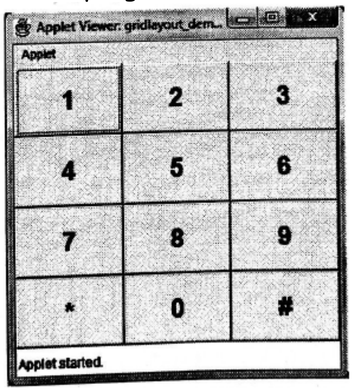
- Here is the list of constructors of GridLayout.
  GridLayout( )
  GridLayout(int rows, int columns)
  GridLayout(int rows, int columns, int horizontal_gap, int vertical_gap)

- First constructor creates a default GridLayout with single row and column.
- Second constructor allow specifying numbers of rows and columns.
- Third form requires numbers of rows and columns as well horizontal and vertical space between components. Following program creates a mobile keybad.

```
import java.awt.*;
import java.applet.*;

/*
<applet code = "gridlayout_demo" width = "500" height = "500" >
</applet>
*/
Public class gridlayout_demo extends Applet
{
    public void init( )
    {
        setFont(new Font("SansSerif" , Font.BOLD, 24));
        GridLayout gl = new GridLayout(4, 3);
        setLayout(gl);
        for(int I = 1; I <–9; i + +)
        {
            Add(new Button ("" + i));
        }
        Button bt1 = new Button ("*");
        Button bt2 = new Button ("0");
        Button bt3 = new Button ("#");
        add(bt1);
        add(bt2);
        add(bt3);
    }
}
```

Output :

Above program will show following output :



Q.5(e) Explain Adapter classes. **[5]**

Ans.:
- While developing different applications to demonstrate event handling, we might feel that only some of the listener methods are useful while others are not.
- But because of implementation of listener interfaces we need to compulsory override all its methods. This takes extra efforts and time to implement even handling.
- The solution for this is, the use of Adapter classes. All the adapter classes are empty implementation of all the methods of an event listener interface.
- We can define a user defined class and extend the required adapter class to it and overriding only those methods that are useful. Refer following table, showing list of adapter classes and the listener interface, it facilitate.

Adapter class	Listener interface
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener

Adapter class	Listener interface
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener

It is not hard to remember adapter classes, simply keep in mind that the listeners shown in above table which has more than one method, has adapter class.

Q.5(f) Explain the delegation event model. [5]

Ans.:
- To begin our study with event, first of all we have to understand how events are handled ? The delegation event model defines the standard background mechanism to process on generated event.
- The mechanism of delegation event model is : the source generates an event and sends it to multiple listeners. Now let us understand each of these parameters one by one.
- Event is an action performed on any source. Event is a result of user's actin which changes the state of a source. Therefore event is also defined as change of state of any source.
- Some of the examples of events are : clicking on a push button, pressing a key from keyboard, clicking mouse button, selecting an item from a list, check or uncheck a checkbox, etc.
- Source is the component object on which action is performed. A source can generated more than one type of events. It is necessary that a source must register listener in order to generate event.
- Each event has its own registration method which registers listener.
- For example : addKeyListener( ) method registers keyboard keys with event listener, addActionListener( ) method registers any select action with event listener.
- Similarly, we can also remove the registration of listener by using removeKeyListener( ) and removeActionListener( ) methods. Hence, add and remove methods are responsible for registering and removing a source with even listener.
- Listener is the object which gets notified when an event occurs. There are several methods which are defined in different interfaces found in java.awt.event.
- For example : MouseListener interface defines methods related to mouse clicked, pressed, released, dragged, moved, entered, exited.
- KeyListener interface defines methods related to keyboard's key pressed, released and typed. Remember that, listener must have been registered with one or more sources to receive notifications.

❑ ❑ ❑ ❑ ❑