RJS First Grade College

Koramangala, Bengaluru – 560034

Department of Computer Science and Applications

Subject – OBJECT ORIENTED PROGRAMMING USING JAVA

UNIT I – INTRODUCTION TO JAVA, OBJECTS AND CLASSES

2 Marks Questions

1. What is Java API?

Ans: The full form of API is Application Programming Interface. It is a document which gives you the list of all the packages, classes, and interfaces, along with their fields and methods.
Using these API's, the programmer can know how to use the methods, fields, classes, interfaces provided by Java libraries.

2. Why java is simple? Mention any two reasons?

Ans: easy to learn
Java inherits the c/c++ syntax and many object oriented features of c++.

3. What is bytecode? Justify how java is platform independent

Ans: Java bytecode are a special set of machine instruction that are notspecified to any
One processor or computer system .A platform specific byte code interpreter executes the java byte code interpreter is also called as javavirtual machine.

4. Java is platform independent language justify?

Ans: java program doesnot rely on particular os .it can run any os which has JVM.

5. What are string literals?

The string literal is a sequence of characters which has to be double coded and occur on a single line.
Eg "vadivel"

6. What is instance variable .Give an example

Ans: These are declared in a class ,but outside a method, they are also called member or field variable, an instance variable is created when an object is created and destroyed when the object is destroyed, visible in all methods and constructors of defining class.

7. What are the default values of float and character primitive datatypes in java

Ans: float –default value -00.0,rarge,is from -3.4E38 to +3.4E38
Char – default value -/ u0000,range from 0 to 65535 characters.

8. Mention the data types in java

Ans: Eight primitive data types can be put in four groups : Integer types : includes byte,short,int, and long
Floating point types: includes float and doubles Character types: includes char
Boolean type: includes Boolean representing true or false

9. Give the general switch statement

Ans:Syntax:
Switch(expression)
{
Case value1:
Value1 Break;
Case value2:
Value2 Break;
}
Ex:
Default:
Class switchdemo
{
Public static void main(String args[])
{
int a=10,b=20,c=30; int status=-1;
If(a>b && a>c)
{
Status=1;
}
Elseif(b>c)
{
}
Else
{
}
Status=2;
Status=3;
Switch(status)
{
Case1:

```
System.out.println("a is greatest"); Break;
Case2:
System.out.println("b is greatest"); Break;
Case3:
System.out.println("c is greatest"); Break;
 Default:
}
 System.out.println("can not be determined");
}
 }
```

10. What is "labelled break" and "labelled continue"

Ans: break statement is used to break the loops and transfer control to the line immediate outside of the loop while continue is used to escape current execution and transfers control back to start of the loop.

11. What is default constructor and parameterized constructor?

Ans: default constructor :
-> is used to initialise all objects with same data.
 It doesn't have any arguments.
 When data is not passed at the time of creating an object,default constructor will be called.
Parameterized constructor:
->is useful to initialize each object with different data.
-> it will have one or more arguments.
-> when data is passed at the time of creation of an object, parameterized constructor will be called.

12. What is the difference between constructor and method?

Ans:

Constructer	Method
Is used to initialize the instancevariables of a class.	Is used for any general purposetasks like calculations
Constructor name and class nameshould be always same.	The method name and class namecan be same or different.
Constructor is called at the time ofcreating an object.	Method can be called aftercreatin the object.
Constructor is called only onceper object.	Method can be called any numberof times on the object.
Constructor is called and executed automatically.	Method is executed only when wewant it.

13. Differentiate between 'String class' and 'String Buffer' class?

String	String Buffer
It defines and supports character strings.	It also represents the strings of characters.
String str = newString(str);	String buffer = new String buffer(str);

14. What do you mean by command line argument?

Ans: it represent the values of passed to main() method to receive and it store the values, main() method provides argument called as String args[].
Example: java Sumclass 10 20

5 Marks, 8 Marks, 10 Marks Questions

1. Explain the features of Java.

Ans: Features of Java
1) Java is simple
2) Java is object-oriented
3) Java is distributed
4) Java is portable & platform Independent
5) Java is compiled & interpreted
6) Java is architecture neutral
7) Java is secure
8) Java is robust
9) Java is multithreading
10) Java is dynamic
11) Java has high performance

1. Java is simple

- Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:
- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java

2. Java is object-oriented

- Java is an object-oriented programming language. Everything in Java is an object Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.
- What is oop??

- O0P stands for Object-Oriented Programming,
- It Focus on objects that contain both data and functions.
- Classes and objects are the two main aspects of object- oriented programming

3. Java is distributed
- Distributed computing involves several computers on a network working together
- Writing network programs in java is like sending and receiving data to and from a file in network environment.

4. Java is portable & platform independent
- Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.
- Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/, OS, etc

5. Java is compiled & interpreted
- Source code is converted to byte code after compilation.
- Byte code can be executed on any machine which has JVM Java Virtual machine)
- In interpreter converts the byte code into machine code
- Byte Code: Java byte codes are a special set of machine instructions that are not specific to any one processor or computer system.

6. Java is architecture neutral
- Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.
- In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

7. Java is secured

->Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:
- No explicit pointer
- Java Programs run inside a virtual machine sandbox

8. Java is robust
- Robust simply means strong. Java is robust because:
- It uses strong memory management
- There is a lack of pointers that avoids security problems.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

9. Java is multithreaded

- A thread is like a separate program, executing concurrently.
- We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread.
- It shares a common memory area.
- Threads are important for multi-media, Web applications, etc.
- For example, downloading an mp3 file while playing the file would be considered as multithreading.

10. Java is dynamic

- Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C+.
- Java supports dynamic compilation and automatic memory management (garbage collection).

11. Java has high performance

- Java is faster than other traditional interpreted programming languages because Java bytecode is “close” to native code.
- It is still a little bit slower than a compiled language (e.g., C++).
- Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.
- It has JIT [Just in Time] Compiler to increase the speed of execution.

2. Explain structure of java program

Structure of Java Program

Java is an object-oriented programming, platform- independent, and secure programming language that makes it popular. Using the Java programming language, we can develop a wide variety of applications. So, before diving in depth, it is necessary to understand the basic structure of Java program in detail. In this section, we have discussed the basic structure of a Java program. At the end of this section, you will able to develop the Hello world Java program, easily.

Let's see which elements are included in the structure of a Java program. A typical structure of a Java program contains the following elements:

- Documentation Section
- Package Declaration
- Import Statements
- Interface Section
- Class Definition
- Class Variables and Variables
- Main Method Class
- Methods and Behaviors Documentation Section

The **documentation section** is an important section but optional for a Java program. It includes basic information about a Java program. The information includes the author's name, date of creation, version, program name, company name, and description of the program. It improves the readability of the program. Whatever we write in the documentation section, the Java compiler ignores the

statements during the execution of the program. To write the statements in the documentation section, we use comments. The comments may be single- line, multi-line, and documentation comments
Single-line Comment: It starts with a pair of forwarding slash (//). For example:
1. //First Java Program
o Multi-line Comment: It starts with a /* and ends with */. We write between these two symbols. For example:
1. /*It is an example of
2. multiline comment*/
Documentation Comment: It starts with the delimiter (/**) and ends with */. For example:
1. /**It is an example of documentation comment*/

Package Declaration

The package declaration is optional. It is placed just after the documentation section. In this section, we declare the package name in which the class is placed. Note that there can be only one package statement in a Java program. It must be defined before any class and interface declaration. It is necessary because a Java class can be placed in different packages and directories based on the module they are used. For all these classes package belongs to a single parent directory. We use the keyword package to declare the package name. For example:

1. package javatpoint; //where javatpoint is the package name
2. package com.javatpoint; //where com is the root directory and javatpoint is the subdirectory

Import Statements

The package contains the many predefined classes and interfaces. If we want to use any class of a particular package, we need to import that class. The import statement represents the class stored in the other package. We use
the import keyword to import the class. It is written before the class declaration and after the package statement. We use the import statement in two ways, either import a specific class or import all classes of a particular package. In a Java program, we can use multiple import statements. For example:

1. import java.util.Scanner; //it imports the Scanner class only
2. import java.util.*; //it imports all the class of the java.util package

Interface Section

It is an optional section. We can create an interface in this section if required. We use the interface keyword to create an interface. An interface is a slightly different from the class. It contains only constants and method declarations. Another difference is that it cannot be instantiated. We can use interface in classes by using the implements keyword. An interface can also be used with other interfaces by using the extends keyword. For example:

1. interface car 2. {
3. void start();
4. void stop(); 5. }

Class Definition

In this section, we define the class. It is vital part of a Java program. Without the class, we cannot create any Java program. A Java program may conation more than one class definition. We use the class keyword to define the class. The class is a blueprint of a Java program. It contains information

about user- defined methods, variables, and constants. Every Java program has at least one class that contains the main() method. For example:

```
1. class Student //class definition
2. {
3. }
```

Class Variables and Constants

In this section, we define variables and constants that are to be used later in the program. In a Java program, the variables and constants are defined just after the class definition. The variables and constants store values of the parameters. It is used during the execution of the program. We can also decide and define the scope of variables by using the modifiers. It defines the life of the variables. For example:

```
1. class Student //class definition
2. {
3.      String sname; //variable
4.      int id;
5.      double percentage;
6. }
```

Main Method Class

In this section, we define the main() method. It is essential for all Java programs. Because the execution of all Java programs starts from the main() method. In other words, it is an entry point of the class. It must be inside the class. Inside the main method, we create objects and call the methods. We use the following statement to define the main() method:

```
1. public static void main(String args[])
2. {
3. }
```

For example:

```
1. public class Student //class definition
2. {
3. public static void main(String args[])
 4. {
5.      //statements
6. }
7. }
```

You can read more about the Java main() method here. Methods and behaviour

In this section, we define the functionality of the program by using the methods. The methods are the set of instructions that we want to perform. These instructions execute at runtime and perform the specified task. For example:

```
public class Demo //class definition
{
public static void main(String args[])
```

```
{
void display()
{
System.out.println("Welcome to javatpoint");
}
//statements
}
}
```

When we follow and use the above elements in a Java program, the program looks like the following.
CheckPalindromeNumber.java

```
/*Program name: Palindrome*/
//Author's name: Mathew
/*Palindrome is number or string that will remains the same When we write that in reverse order. Some example of palindrome is 393, 010, madam, etc.*/
//imports the Scanner class of the java.util package
import java.util.Scanner;
//class definition
public class CheckPalindromeNumber
{
//main method
public static void main(String args[])
{
//variables to be used in program
 int r, s=0, temp;
int x; //It is the number variable to be checked for palindrome Scanner sc=new Scanner(System.in);
System.out.println("Enter the number to check: ");
//reading a number from the user x=sc.nextInt();
//logic to check if the number id palindrome or not temp=x;
while(x>0)
{
r=x%10; //finds remainder s=(s*10)+r;
x=x/10;
}
if(temp==s)
System.out.println("The given number is palindrome.");
else
System.out.println("The given number is not palindrome.");
}
}
```

3. Explain bitwise operators

Ans: java defines several bitwise operators which can be applied to the integer types long,int,short,char and byte

Operators	Description	Example
&	(bitwise AND operators) Copies the bit to the result if it exist in both operators	(A & B)will give 12 which is 0000 1100
|	(bitwise OR operators) copies a bit if H exist in eitheroperand	(A|B) will give 61 which is 0011 1101
^	(bitwise XOR operator) copies the bit if it is set in one operand but not both	(A^B) will give 49 which is 0011 0001
~	(bitwise 1's compliment operator)is unary andhas the effect of negating bits	(~A) will give 60 which gives 1100 0011
<<	(bitwise left-shift Operators) the left operands value is more left by the number of bits specified by right operand	A<<2 will give 240 which is 1111 0000
>>	(bitwise right –shift operators) the left operands value is moved right by the number of bits specified by right operand	A>>2 will give 15 which is 0000 1111
>>>	(bitwise unsigned right shift operater) the left operands value is moved right by the number bits specified by the right operand and shifted values are filledup	A>>>2 will give 15 which is 0000 1111

	with zeros	

Program: class shiftoperationdemo
{
Public static void main(String args[])
{
Int num1=741;
Int num 2=-858993460;
System.out.println("741<<1="+(num1<<1)); System.out.println("741>>1="+(num1>>1));
System.out.println("741>>>1="+(num1>>>1)); System.out.println("-858993460>>6:" +(num2>>6));
System.out.println("-858993460>>>6:" +(num2>>>6));
}
}

4. What are static variables and static methods?

Ans: Static variables:
Static variables are special type of variables that are not associated with object,they are associated with class
The static variables are also called as class variables.
The static variables can be accessed without an object.
DECLARING STATIC VARIABLES:
Static variables are initialized only once,at the start of execution.
Syntax: <class-name>.<variable-name>

Static methods:
The methods can also be declared as static. A static method is associated with a class rather than the instatnce.
The static methods are also called as class members.

DECLARING STATIC METHOD:
class staticDemo
{
Int x,y; Static int z;
Void static method1()
{
System.out.println (z);
}
}
Syntax : <class-name>.<method-name>(arguments)

5. What is Constructor overloading? Write a program to demonstrate the usage of constructor overloading?

Ans: Constructor Overloading: Writing two or more constructor with the same name but with different number and types of arguments.
Example:

```
Class ConsDemo
{
Int i,j; ConsDemo()
{
i = 10;
j = 20;
}
ConsDemo (int a)
{
i = a;
j = 20;
}
ConsDemo (int a, int b)
{
i = a; j = b;
}
Void display()
{
System.out.println (“ i = “ +i); System.out.println (“ j = “ +j);
}
Public static void main (String args[])
{
ConsDemo a1 = new ConsDemo(); ConsDemo a2 = new ConsDemo(11); ConsDemo a3 = new ConsDemo(111,222); a1.display();
a2.display();
a3.display();
}
}
```

6. Explain any three string methods with example?

Ans: The string methods are:
(i.)String concat():
Creates a new string by appending the contents of a string object passed as argument to the contents of the string on which the method is invoked.
Syntax: String concat(String str);
Example : String s1 =”raja”;
String s2 = “ram”; String s3 = s1.concat(s2); System.out.println(s1); System.out.println(s2); System.out.preintln(s3);

(ii.) String replace();
Create a new string using the same contents as that of the string object on which the method is invoked.
Example: public string replace(char old ,char new) String original +"Java programming"; System.out.println (Original replace ('a','o'));

(iii.) String charAt:
Return type is character.
Example: String s1 = "RJSFGC college"; char C = s1.charAt(0);
c=R.

UNIT II – INHERITANCE AND POLYMORPHISM

2 Marks Questions

1. What is the use of Super keyword and This keyword?

Ans: Super keyword:
The super keyword is used to refer the super class object.
The super keyword is used to call super class method from subclass.
The super keyword is used to call super class constructor from subclass constructor.

This keyword:
This keyword refers to current object.
This keyword used to solve the program of name conflict, and differentiate the instant and local variables.
It points to the object that is executing the block in which this keyword is present.

2. What is the difference between class and abstract class?

Ans: the difference between class and abstract class are:

Class	**Abstract class**
It does not contain abstract method.	It contains abstract method.
It can be instantiated.	It cannot be instantiated.

3. Define a package. Mention its use.

A java package is a group of similar types of classes, interfaces and sub- packages.
Package in java can be categorized in two form, built-in package and user- defined package.
There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.
Advantages of 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. What are the different access modifiers in Java?

Ans. Java provides four types of access modifiers or visibility specifiers i.e., default, public, private, and protected. The default modifier does not have any keyword associated with it. When a class or method or variable does not have an access specifier associated with it, we assume it ishaving default access.

5 Marks, 8 Marks, 10 Marks Questions

1. Define Inheritance .Explain the types of inheritance supported by java.

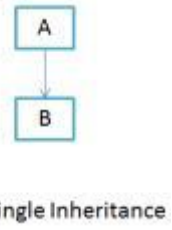
The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called **inheritance**.

Syntax: Inheritance in Java
To inherit a class we use extends keyword. Here class XYZ is child class and class ABC is parent class. The class XYZ is inheriting the properties and methods of ABC class.

```
class XYZ extends ABC
{
}
```

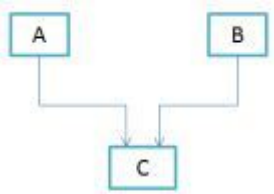
Types of Inheritance

1) Single Inheritance
Single inheritance is damn easy to understand. When a class extends another one class only then we call it a single inheritance.



(a) Single Inheritance

2) Multiple Inheritance
"**Multiple Inheritance**" refers to the concept of one class extending (Or inherits) more than one base class. The problem with "multiple inheritance" is that the derived class will have to manage the dependency on two base classes.



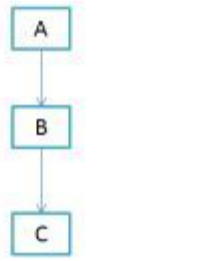
(b) Multiple Inheritance

Note 1: Multiple Inheritance is very rarely used in software projects. Using Multiple inheritance often leads to problems in the hierarchy. This results in unwanted complexity when further extending the class.

Note 2: Most of the new OO languages like **Small Talk, Java, C# do not support Multiple inheritance**. Multiple Inheritance is supported in C++.

3) Multilevel Inheritance

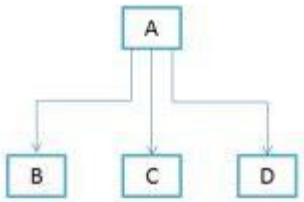
Multilevel inheritance refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class.



(d) Multilevel Inheritance

4) Hierarchical Inheritance

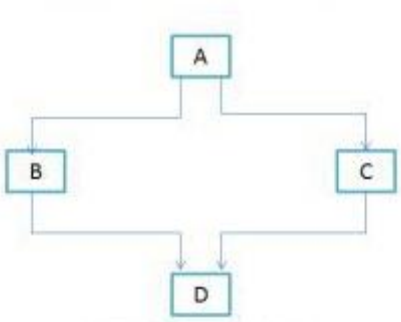
In such kind of inheritance one class is inherited by many **sub classes**. In below example class B,C and D **inherits** the same class A.



(c) Hierarchical Inheritance

5) Hybrid Inheritance

In simple terms you can say that Hybrid inheritance is a combination of **Single** and **Multiple inheritance.** A hybrid inheritance can be achieved in the java in a same way as multiple inheritance can be!! Using interfaces.



(e) Hybrid Inheritance

2. Explain with an example: (i) Method Overloading. (ii)Method overriding. (iii) Abstract method. (iv) Abstract class.

Ans: (i) Method overloading: It deals with multiple methods in the same class with the same name but different method signature.
It can be static or non-static.
Example: import java.lang.*; class methodOverload Demo

```
{
Void add(int x,int y)
System.out.println(+(x+y);
}
int add (int x, int y,int z)
{
return;
}
Void add (double x, double y)
{
System.out.println (+(x+y));
}
}
Class methodOverloadmain
{
Public static void main (String a[])\
{
methodOverloaddemo ob = new methodoverloaddemo(); ob.add (10,20);
System.out.println ("Sum =" + ob.add (10,20,30)); Ob.add (5.5, 2.5);
}
}
```

(ii) Method Overriding: Overriding deals with two methods, one in the parent class other one in the child class.
Method with super class and subclass.

```
class Baseclass
{
Public void getAmount (int rate)
{
.....
}
}
class Myclass extends Baseclass
{
Public void getAmount (int rate)
{
.....
}
```

}
(iii) Abstract method: The method which does not have body.
It just contains only method signature.
Compiler does not allow abstract method tp be private or final.
Example: abstract void method();
(iv) Abstract class: Abstract class is a class which contain 0 or more abstract method.
It cannot be instantiated.
An abstract class does not define a complete implementation.

3. Explain interface with an example

An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class.

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

interface <interface name>{

// declare constant fields

// declare methods that abstract

// by default.

}

Interface fields are public, static and final by default, and the methods are public and abstract.

In this example, the Printable interface has only one method, and its implementation is provided in the A6 class.

interface printable{

void print();

```
}

class A6 implements printable{

public void print(){System.out.println("Hello");}

public static void main(String args[]){ A6 obj = new A6();

obj.print();

}

}
```

4. What is package? How do you create and access a package?

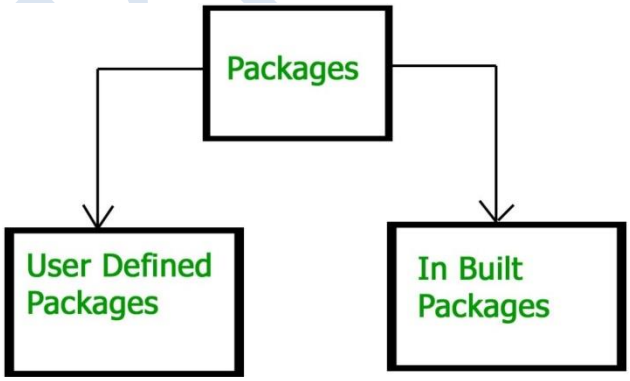
Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.

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). All we need to do is put related classes into packages



Types of packages:

Illustration of user-defined packages:

Creating our first package: File name – ClassOne.java

package package_name; public class ClassOne {

```
public void methodClassOne() {

System.out.println("Hello there its ClassOne");

}

}
```

```
package package_one; public class ClassTwo {

public void methodClassTwo(){ System.out.println("Hello there i am ClassTwo");

}

}
```

```
package package_one;

public class ClassTwo {

public void methodClassTwo(){ System.out.println("Hello there i am ClassTwo");

}

}
```

Making use of both the created packages:

```
import package_one.ClassTwo; import package_name.ClassOne;

public class Testing {

public static void main(String[] args){ ClassTwo a = new ClassTwo(); ClassOne b = new ClassOne();
a.methodClassTwo(); b.methodClassOne();

}

}
```

Important points:

1. Every class is part of some package.

2. If no package is specified, the classes in the file goes into a special unnamed package (the same unnamed

package for all files).

3. All classes/interfaces in a file are part of the same package. Multiple files can specify the same package name.

4. If package name is specified, the file must be in a subdirectory called name (i.e., the directory name must match the package name).

5. We can access public classes in another (named) package using: package-name, class-name

UNIT III – EVENT AND GUI PROGRAMMING AND I/O PROGRAMMING

2 Marks and 5 Marks Questions

1. **What is the difference between a Swing and AWT components?**

AWT components re heavy weight, whereas Swing components are lightweight.

Heavy weight components depend on the local windowing toolkit.

For example, java.awt.Button is a heavy weight component, when it is running on the Java platform for Unix platform, it maps to a real Motif button.

2. **What are AWT peers?**

A component is associated with a standard AWT button object, a peer object and an interfacing button object constructed per the native GUI.

3. **What are the different types of controls in AWT?**

The AWT supports the following types of controls:

Labels, Pushbuttons, Checkboxes, Choice lists, Lists, Scroll bars, Text components

These controls are subclasses of component.

4. **What is the use of the window class?**

The window class can be used to create a plain, bare bones window that does not have a border or menu.

The window class can also be used to display introduction or welcome screens.

5. **What is an applet?**

Applet is a dynamic and interactive program that runs inside a web page displayed by a java capable browser.

6. **What is the difference between applications and applets?**

Application must be run on local machine whereas applet needs no explicit installation on local machine.

Application must be run explicitly within a java compatible virtual machine where as applet loads and runs itself automatically in a java enabled browser.

Application starts execution with its main method whereas applet starts execution with its init method.

Application can run with or without graphical user interface whereas applet must run within a graphical user interface.

7. **When is update method called?**

Whenever we minimize, maximize, restart the applet and explicitly calling the repaint() method in the code.

Repaint() method will implicitly call the update() method.

8. **Which method is called by applet class to load an image?**

getImage(URL object, filename) is used for this purpose.

9. **Why does it take so much time to access an Applet having Swing components the first time?**

Because behind every swing component are many Java objects and resources.

This takes time to create them in memory. JDK 1.3 from Sun has some improvements which may lead to faster execution of Swing applications.

10. **What are the benefits of Swing over AWT?**

Swing components are light weight.

We can have a pluggable look and feel feature which shows us how they appear in other platforms.

We can add images to Swing components. We have toolbars and tooltips in Swing.

11. **What are the component and container class?**

A component is a graphical object. A few examples of components are

Button

Canvas

Checkbox

Choice etc.

12. What is the difference between the paint() and repaint() method?

The paint() method supports painting via a Graphics object.

The repaint() method is used o cause paint() to be invoked by the AWT painting thread.

13. What interface is extended by AWT event listener?

The java.util.EventListener interface is extended by all the AWT event listeners.

14. What is a container in a GUI?

A Container contains and arranges other components through the use of layout managers, which use specific layout policies to determine where components should go as a function of the size of the container.

15. What is the default layout for Applet?

The default layout manager for an Applet is FlowLayout, and the FlowLayout manager attempts to honor the preferred size of any components.

16. What is the difference between a MenuItem and a CheckboxMenuItem?

The CheckboxMenuItem class extends the MenuItem class to support a menu item that may be checked or unchecked.

17. How are the elements of different layouts organized?

FlowLayout: The elements of a FlowLayout are organized in a top to bottom, left to right fashion.

Border Layout: The elements of a BorderLayout are organized at the borders and the centre of a container.

CardLayout: The elements of a CardLayout are stacked, on the top of the other, like a deck of cards.

GridLayout: The elements of a GridLayout are equal size and are laid out using the square of a grid.

GridBagLayout: The elements of a GridBagLayout are organized according to a grid.

18. What is the default layout for Applet?

The default layout manager for an Applet is FlowLayout, and the FlowLayout manager attempts to honor the preferred size of any components.

19. What is the difference between Grid and GridbagLayout?

In Grid layout the size of each grid is constant where as in GridbagLayout grid size can varied.

20. What is a layout manager?

A layout manager is an object that is used to organize components in a container.

21. What is the difference between a window and a frame?

The Frame class extends Window to define a main application window that can have a menu bar.

22. What are the default layouts for a applet, a frame and a panel?

For an applet and a panel, Flow layout, and The FlowLayout manager attempts to honor the preferred size of any components.

23. What is Java I/O ?

Java I/O (Input and Output) is used to process the input and produce the output. Java makes use of the stream concepts to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

24. What is difference between Scanner and BufferedReader?

Scanner is used for parsing tokens from the contents of the stream while BufferedReader just reads the stream and does not do any special parsing. Usually we pass a BufferedReader to a scanner as the source of characters to parse.

25. What is the use of the window class?

The window class can be used to create a plain, bare bones window that does not have a border or menu.

The window class can also be used to display introduction or welcome screens.

26. How to process java.io.InputStream object and produce a String?

```
static String convertStreamToString(java.io.InputStream iStream)
{
   java.util.Scanner scanInput = new java.util.Scanner(iStream).useDelimiter("\\A");
   return scanInput.hasNext() ? scanInput.next() : "";
}
```

27. What are the uses of FileInputStream and FileOutputStream in java?

java.io.FileInputStream and java.io.FileOutputStream were introduced in JDK 1.0. These APIs are used to read and write stream input and output. They can also be used to read and write images.

28. What is the use of the PrintStream class in Java IO?

PrintStream is used to write data on Console, for example, System.out.println(), here out is an object of PrintStream class and we are calling println() method from that class.

29. How do you process a large file in Java?

You can also use both BufferedReader and Scanner to read a text file line by line in Java.

Reading a text file using FileReader

The FileReader is your general-purpose Reader implementation to read a file. It accepts a String path to file or a java.io.File instance to start reading. It also provides a couple of overloaded read() methods to read a character or read characters into an array or into a CharBuffer object.

Here is an example of reading a text file using FileReader in Java:

```
public static void readTextFileUsingFileReader(String fileName) {
   try {
     FileReader textFileReader = new FileReader(fileName);
     char[] buffer = new char[8096];
     int numberOfCharsRead = textFileReader.read(buffer);
     while (numberOfCharsRead != -1) {
       System.out.println(String.valueOf(buffer, 0, numberOfCharsRead));
       numberOfCharsRead = textFileReader.read(buffer);
     }
     textFileReader.close();
   } catch (IOException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
   }
 }
```

30. Explain the use of DataInputStream and DataOutputStream with example

The DataInputStream class read primitive Java data types from an underlying input stream in a machine-independent way. While the DataOutputStream class write primitive Java data types to an output stream in a portable way.

Here is an example to demonstrate the use of DataInputStream and DataOutputStream.

DataInputOutputStreamExample.java

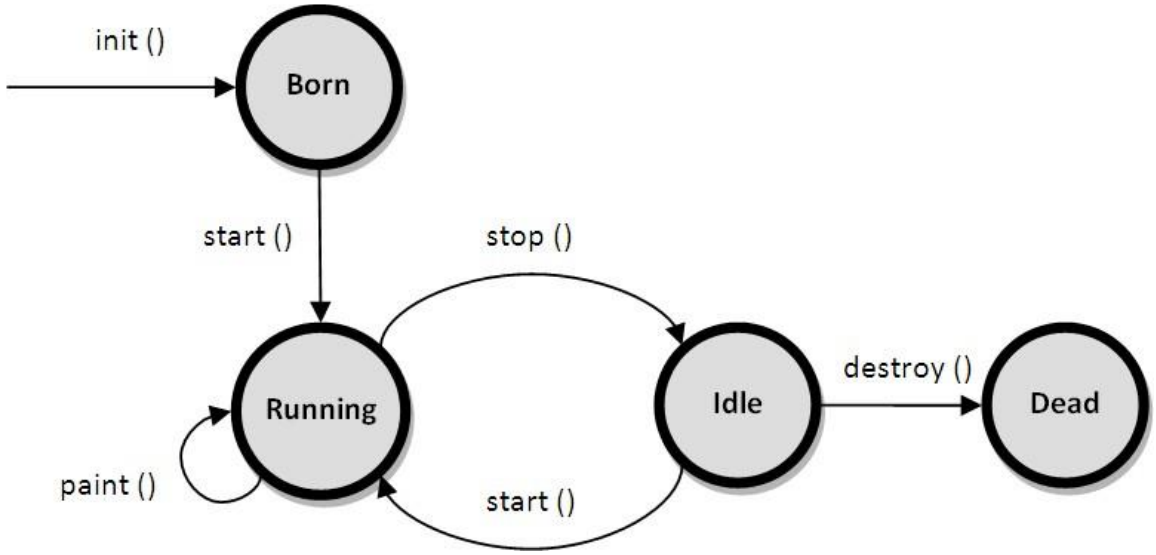
```
package com.boraji.tutorial.io;

import java.io.DataInputStream; import java.io.DataOutputStream; import java.io.File;
```

```
import java.io.FileInputStream; import java.io.FileOutputStream; import java.io.IOException;

/**

*	@author imssbora

*	DataInputOutputStreamExample.java

* Nov 5, 2016

*/

public class DataInputOutputStreamExample {

public static void main(String[] args) { File file = new File("file.txt");

/*Write primitive data type to file*/ FileOutputStream fileOutputStream = null; DataOutputStream dataOutputStream = null; try {

fileOutputStream=new FileOutputStream(file);

dataOutputStream=new DataOutputStream(fileOutputStream);

dataOutputStream.writeInt(50); dataOutputStream.writeDouble(400.25);

dataOutputStream.writeChar('A'); dataOutputStream.flush();

} catch (IOException e) {

e.printStackTrace();

}finally {

try {

if(fileOutputStream!=null){ fileOutputStream.close();

}

if(dataOutputStream!=null){ dataOutputStream.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}
```

```
/*Read primitive data type from file*/ FileInputStream fileInputStream = null; DataInputStream dataInputStream = null; try {

fileInputStream = new FileInputStream(file);

dataInputStream = new

DataInputStream(fileInputStream);

System.out.println(dataInputStream.readInt());      System.out.println(dataInputStream.readDouble()); System.out.println(dataInputStream.readChar());

} catch (IOException e) {

e.printStackTrace();

}finally {

try {

if(fileInputStream!=null){

fileInputStream.close();

}

if(dataInputStream!=null){ dataInputStream.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

}
```

31. Explain the life cycle of applet.

The life cycle of an applet is as shown in the figure below:



As shown in the above diagram, the life cycle of an applet starts with *init()* method and ends with *destroy()* method. Other life cycle methods are *start()*, *stop()* and *paint()*. The methods to execute only once in the appletlife cycle are *init()* and *destroy()*. Other methods execute multiple times.

Below is the description of each applet life cycle method:

init(): The init() method is the first method to execute when the applet is executed. Variable declaration and initialization operations are performed in thismethod.

start(): The start() method contains the actual code of the applet that should run. The start() method executes immediately after the *init()* method. It also executes whenever the applet is restored, maximized or moving from one tab toanother tab in the browser.

stop(): The stop() method stops the execution of the applet. The stop() method executes when the applet is minimized or when moving from one tab to anotherin the browser.

destroy(): The destroy() method executes when the applet window is closed orwhen the tab containing the webpage is closed. *stop()* method executes just before when destroy() method is invoked. The destroy() method removes the applet object from memory.

paint(): The paint() method is used to redraw the output on the applet displayarea. The paint() method executes after the execution of *start()* method and whenever the applet or browser is resized.

The method execution sequence when an applet is executed is:

- init()
- start()
- paint()

The method execution sequence when an applet is closed is:

- stop()
- destroy()

32. Explain the methods of Graphics class methodsCommonly used methods of Graphics class:

1. public abstract void drawString(String str, int x, int y): is used todraw the specified string.
2. public void drawRect(int x, int y, int width, int height): draws arectangle with the specified width and height.
3. public abstract void fillRect(int x, int y, int width, int height): is usedto fill rectangle with the default color and specified width and height.
4. public abstract void drawOval(int x, int y, int width, int height): isused to draw oval with the specified width and height.
5. public abstract void fillOval(int x, int y, int width, int height): is usedto fill oval with the default color and specified width and height.
6. public abstract void drawLine(int x1, int y1, int x2, int y2): is used todraw line between the points(x1, y1) and (x2, y2).
7. public abstract boolean drawImage(Image img, int x, int y,ImageObserver observer): is used draw the specified image.
8. public abstract void drawArc(int x, int y, int width, int height, intstartAngle, int arcAngle): is used draw a circular or elliptical arc.
9. public abstract void fillArc(int x, int y, int width, int height, intstartAngle, int arcAngle): is used to fill a circular or elliptical arc.
10. public abstract void setColor(Color c): is used to set the graphics current color to the specified color.
11. public abstract void setFont(Font font): is used to set the graphicscurrent font to the specified font.

Example of Graphics in applet:

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

public class GraphicsDemo extends Applet{

public void paint(Graphics g){ g.setColor(Color.red); g.drawString("Welcome",50, 50);
    g.drawLine(20,30,20,300); g.drawRect(70,100,30,30); g.fillRect(170,100,30,30);
    g.drawOval(70,200,30,30);

g.setColor(Color.pink); g.fillOval(170,200,30,30); g.drawArc(90,150,30,30,30,270);
    g.fillArc(270,150,30,30,0,180);

}
}
```

33. Explain the use of FileInputStream class and FileOutputStream class.

Java provides I/O Streams to read and write data where, a Stream represents an input source or an output destination which could be a file, i/o devise, other program etc.

There are two types of streams available −

•InputStream − This is used to read (sequential) data from a source.

•OutputStream − This is used to write data to a destination.

FileInputStream

This class reads the data from a specific file (byte by byte). It is usually used to read the contents of a file with raw bytes, such as images.

To read the contents of a file using this class −

•First of all, you need to instantiate this class by passing a String variable or a File object, representing the path of the file to be read.

FileInputStream inputStream = new FileInputStream("file_path");

or,

File file = new File("file_path");

FileInputStream inputStream = new FileInputStream(file);

Then read the contents of the specified file using either of the variants of read() method −

oint read() − This simply reads data from the current InputStream and returns the read data byte by byte (in integer format).

This method returns -1 if the end of the file is reached.

oint read(byte[] b) − This method accepts a byte array as parameter and reads the contents of the current InputStream, to the given array

This method returns an integer representing the total number of bytes or, -1 if the end of the file is reached.

oint read(byte[] b, int off, int len) − This method accepts a byte array, its offset (int) and, its length (int) as parameters and reads the contents of the current InputStream, to the given array.

oThis method returns an integer representing the total number of bytes or, -1 if the end of the file is reached.

UNIT IV – MULTITHREADING, COLLECTIONS IN JAVA AND JAVABEANS AND NETWORK PROGRAMMING

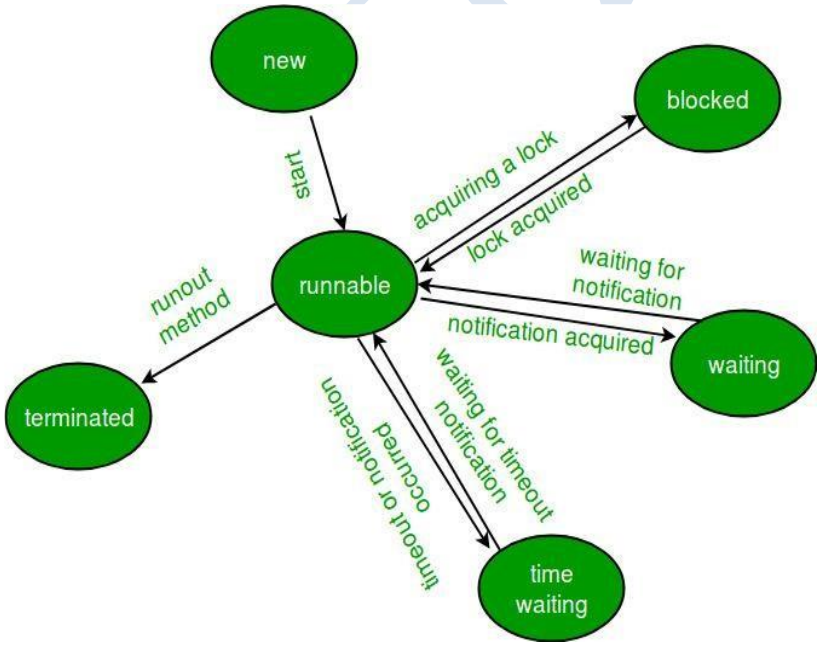
2 Marks and 5 Marks Questions

1. **Explain the life cycle of a Thread.**

Lifecycle and States of a Thread in Java

A thread in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:

1. New
2. Runnable
3. Blocked
4. Waiting
5. Timed Waiting
6. Terminated



Life Cycle of a thread

1. New Thread: When a new thread is created, it is in the new state. The thread has not yet started to run when thread is in this state. When a thread lies in the new state, it's code is yet to be run and hasn't started to execute.

2. Runnable State: A thread that is ready to run is moved to runnable state. In this state, a thread might actually be running or it might be ready run at any instant of time. It is the responsibility of the thread scheduler to give the thread, time to run. A multi-threaded program allocates a fixed amount of time to each individual thread. Each and every thread runs for a short

while and then pauses and relinquishes the CPU to another thread, so that other threads can get a chance to run. When this happens, all such threads that are ready to run, waiting for the CPU and the currently running thread lies in runnable state.

3. Blocked/Waiting state: When a thread is temporarily inactive, then it's in one of the following states:

- Blocked
- Waiting

For example, when a thread is waiting for I/O to complete, it lies in the blocked state. It's the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread. A thread in this state cannot continue its execution any further until it is moved to runnable state. Any thread in these states does not consume any CPU cycle.

1. A thread is in the blocked state when it tries to access a protected section of code that is currently locked by some other thread. When the protected section is unlocked, the schedule picks one of the thread which is blocked for that section and moves it to the runnable state. Whereas, a thread is in the waiting state when it waits for another thread on a condition. When this condition is fulfilled, the scheduler is notified and the waiting thread is moved to runnable state.

If a currently running thread is moved to blocked/waiting state, another thread in the runnable state is scheduled by the thread scheduler to run. It is the responsibility of thread scheduler to determine which thread to run.

2. Timed Waiting: A thread lies in timed waiting state when it calls a method with a time out parameter. A thread lies in this state until the timeout is completed or until a notification is received. For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.

3. Terminated State: A thread terminates because of either of the following reasons:

- Because it exists normally. This happens when the code of thread hasentirely executed by the program.
- Because there occurred some unusual erroneous event, likesegmentation fault or an unhandled exception.

A thread that lies in a terminated state does no longer consumes any cyclesof CPU.

2. How will you create thread in java?There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

Thread class:

Thread class provide constructors and methods to create and perform operations on athread.Thread class extends Object class and implements Runnable interface.

he Runnable interface should be implemented by any class whose instances are

intended to be executed by a thread. Runnable interface have only one methodnamed run().

1. **public void run():** is used to perform action for a thread.

Starting a thread:

start() method of Thread class is used to start a newly created thread. It performsfollowing tasks:

- A new thread starts(with new callstack).
- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

1) Java Thread Example by extending Thread class

```
class Multi extends Thread{
public void run(){ System.out.println("thread is running...");
}
public static void main(String args[]){Multi t1=new Multi();
t1.start();
}
}
```

Output:thread is running...

2) Java Thread Example by implementing Runnable interface

```
class Multi3 implements Runnable{ public void run(){ System.out.println("thread is running...");
}

public static void main(String args[]){Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);t1.start();
```

}

}

3. **What's the difference between the methods sleep() and wait()?**

The code sleep(1000); puts thread aside for exactly one second. The code

wait(1000), causes a wait of up to one second. A thread could stop waiting earlier if it receives the notify() or notifyAll() call. The method wait() is definedin the class Object and the method sleep() is defined in the class Thread.

4. **Define deadlock.**

This occurs when two threads have a circular dependency on a pair of synchronized objects. For example, suppose one thread enters the monitor on object X and another thread enters the monitor on object Y. If the thread in X tries to call any synchronized method on Y, it will block as expected. However, if the thread in Y, in turn, tries to call any synchronized method on X, the thread waits forever, because to access X, it would have to release its own lock on Y so that the first thread could complete.

5. **Define multithreading.**

A thread is a line of execution. It is the smallest unit of code that is dispatchedby the scheduler. Thus, a process can contain multiple threads to execute its different sections. This is called multithread.

6. **List out the advantages of multithreading.**

The advantages are as follows

- Can be created faster
- Requires less ove.rheads
- Inter-process communication is faster
- Context switching is faster
- Maximum use of CPU time

7. **Define the term thread.**

A thread is a line of execution. It is the smallest unit of code that is dispatched by the scheduler. Thus,

a process can contain multiple threads toexecute its different sections. This is called multithread.

8. **List the several states of 'Thread' in Java.**

There are four states associated with a thread namely – new, runnable, dead,blocked

9. **What is synchronization? Briefly explain.**

Two or more threa.ds accessing the same data simultaneously may lead to loss of data integrity. For example, when two people access a savings account, it is possible that one person may overdraw and the cheque

may bounce. The importance of updating of the pass book can be wellunderstood in this case.

10. **Define an exception**

An exception is an abnormal condition, which occurs during the execution of a program Exceptions are erroneous events like division by zero, opening of a file that does not exist, etc. A java execution is an object, which describes the error condition that has materialized in the program.

11. **Explain the usage of try and catch clause**

The try and catch clause is used to handle an exception explicitly. The advantages of using the try and catch clause are that, it fixes the error andprevents the program from terminating abruptly.

12. **What is use of 'throw statement' give an example? (or) state the purposeof the throw statement.**

Whenever a program does not want to handle exception using the try block, it can use the throws clause. The throws clause is responsible to handle the different types of exceptions generated by the program. This clause usually contains a list of the various types of exception that are likely to occur in the program.

13. **List any three common run time errors.**

Exception	Meaning

ArithmeticException	Arithmetic error, such as divide-by-zero
ArrayIndexOutOfBoundsException IllegalThreadStateException	Array index is out-of-bounds Requested operation not compatiblewith current thread state

14. How user defined exception is done?

A. User Defined Exception or custom exception is creating your own exceptionclass and throws that exception using 'throw' keyword. This can be done by extending the class Exception. There is no need to override any of the above methods available in the Exception class, in your derived class.

15. What is error? Compare with exception.

A. Error: An Error "indicates serious problems that a reasonable application should not try to catch." Both Errors and Exceptions are the subclasses of Throwable class. Errors are the conditions which cannot get recovered by anyhandling techniques. Errors mostly occur at runtime that's they belong to an unchecked type.

Exceptions are the problems which can occur at runtime and compile time. Itmainly occurs in the code written by the developers. Exceptions are divided into two categories such as checked exceptions and uncheckedexceptions.

16. What is framework in Java?

A framework is a popular and readymade architecture that contains a set of classes and interfaces.

17. What is the Collection framework in Java?

Collection Framework is a grouping of classes and interfaces that is used to store and manage the objects. It provides various classes like Vector, ArrayList, HashSet, Stack, etc. Java Collection framework can also be used for interfaces like Queue, Set, List, etc.

18. Explain Collections Class

java.util.Collections is a class consists of static methods that operate on collections. It contains polymorphic algorithms to operate on collections, "wrappers". This class contains methods for algorithms, like binary sorting, search, shuffling, etc.

19. List out benefits of generic collection.

- The benefits of using the generic collection are:
- If the programmers are using generic class, they don't require typecasting.
- It is type-safe and can be checked at the time of compilation.
- It provides the stability of the code by detecting bug at the compilation time.

20. Explain equals() with example

Equals() verifies whether the number object is equal to the object, which is passed as an argument or not.

The syntax of the equals() method is:

public boolean equals(Object o)

This method takes two parameters 1) any object, 2) return value. It returns true if the passed argument is not null and is an object of a similar type having the same numeric value.

Example:

```
import java.lang.Integer;

public class Test {

  public static void main(String args[]) {

    Integer p = 5;

    Integer q = 20;

    Integer r =5;

    Short s = 5;

    System.out.println(p.equals(q));

    System.out.println(p.equals(r));

    System.out.println(p.equals(s));

  }

}
```

21. Explain the basic interfaces of the Java collections framework.

Java collection framework is a root of the collection hierarchy. It represents a group of objects as its elements. The Java programming language does not provide a direct implementation of such interface.

Set: Set is a collection having no duplicate elements. It uses hashtable for storing elements.

List: List is an ordered collection that can contain duplicate elements. It enables developers to access any elements from its inbox. The list is like an array having a dynamic length.

MAP: It is an object which maps keys to values. It cannot contain duplicate keys. Each key can be mapped to at least one value.

22. Distinguish between ArrayList and Vector in the Java collection framework.

ArrayList	Vector
ArrayList is cannot be synchronized.	Vector can be is synchronized.
It is not a legacy class.	It is a legacy class.
It can increase its size by 50% of the size of the array.	It can increase its size by doubling the size of the array.
ArrayList is not thread-safe.	Vector is a thread-safe.

23. What is EJB?

A server-side component, which manages the architecture for constricting enterprise applications and managed is called Enterprise JavaBeans(EJB).

24. When was EJB developed?

EJB was developed by IBM in 1997.

25. Who took over EJB?

EJB was taken over by Sun Microsystems in 1999.

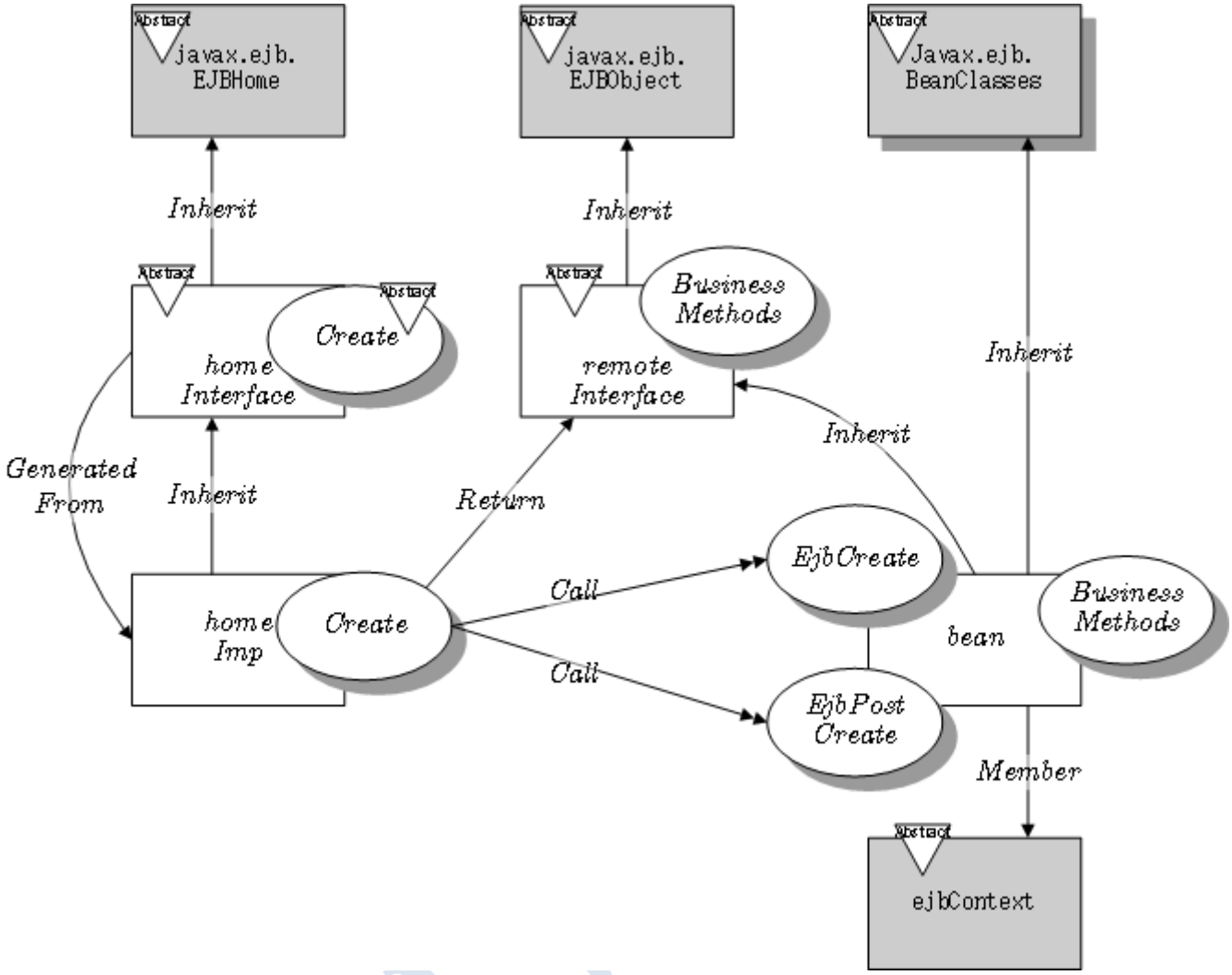
26. Enlist the Enterprise Beans types?

Session Beans: Expanded as “Stateful”,”Stateless” and “Singleton”, A Remote or Local interface is used to access the EJB files.

Message Driven Beans (MDB): Asynchronous execution by means of messaging paradigm is supported.

27. What were Entity Beans?

Entity Beans were presented in the earlier versions of EJB consisting of persistent data in distributed objects.



28. Define network programming.

It refers to writing programs that executes across multiple devices (computers), in which the devices are all connected to each other using a network.

29. What is the Proxy Server?

A proxy server speaks the client side of a protocol to another server.

This is often required when clients have certain restrictions on which servers they can connect to.

And when several users are hitting a popular website, a proxy server can get the contents of the web server's popular pages once, saving expensive internetwork transfers while providing faster access to those pages to the clients.

30. What is the difference between TCP and UDP?

These two protocols differ in the way they carry out the action of communicating.

A TCP protocol establishes a two way connection between a pair of computers, while the UDP protocol is a one way message sender.

The common analogy is that TCP is like making a phone call and carrying on a two way communication while UDP is like mailing a letter.

31. What is a socket?

Sockets provide the communication mechanism between two computers using TCP.

A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

32. What is SMTP?

Simple Mail Transmission Protocol, the TCP/IP Standard for Internet mails.

SMTP exchanges mail between servers; contract this with POP, which transmits mail between a server and a client.

33. What are the two important TCP Socket classes?

Socket and ServerSocket.

ServerSocket is used for normal two way socket communication.

Socket class allows us to read and write through the sockets.

getInputStream() and getOutputStream() are the two methods available in Socket class.

34. What are the seven layers of OSI model?

- Application
- Presentation
- Session
- Transport
- Network
- DataLink
- Physical Layer

35. Explain try catch with an example

try block is used to enclose the code that might throw an exception. It must beused within the method.

If an exception occurs at the particular statement of try block, the rest of the block code will not execute. So, it is recommended not to keeping the code intry block that will not throw an exception.

Syntax of Java try-catch

try{

//code that may throw an exception

}**catch**(Exception_class_Name ref){}catch block

Java catch block is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent classexception (i.e., Exception) or the generated exception type. However, the good approach is to declare the generated type of exception.

The catch block must be used after the try block only. You can use multiplecatch block with a single try block.

public class TryCatchExample2 {

public static void main(String[] args) {

try
{

int data=50/0; //may throw exception

}

//handling the exception

catch(ArithmeticException e)

{

System.out.println(e);

}

System.out.println("rest of the code");

}

}

36. Explain nested try statementWhy use nested try block

Sometimes a situation may arise where a part of a block may cause one errorand the entire block itself may cause another error. In such cases, exception handlers have to be nested.

Syntax:

....

try
{

statement 1;

statement 2;

try
{

statement 1;

statement 2;

}

catch(Exception e)

{

}

}

catch(Exception e)

{

}

Example

class Excep6{

public static void main(String args[]){

try{ **try**{

System.out.println("going to divide");

int b =39/0;

}**catch**(ArithmeticException e){System.out.println(e);}

```
try{
int a[]=new int[5];a[5]=4;
}catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}

System.out.println("other statement);
}catch(Exception e){System.out.println("handeled");}

System.out.println("normal flow..");
}
}
```