BALASORE SCHOOL OF ENGINEERING, BALASORE

STUDY MATERIAL

BRANCH – COMPUTER SCIENCE & ENGG.
SUBJECT :- OBJECT ORIENTED METHODOLOGY
SUBJECT CODE- TH-4
SEMESTER :- 3RD
PREPARED BY: - PRADEEP KUMAR GIRI

Q.1.

(a) Define Data Abstraction?

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.

Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviours of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

(b) What do you by Polymorphism?

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

(c) Define copy constructor.

The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
Java supports for copy constructors but unlike C++ language, Java does not provide an explicit copy constructor you need to define it yourself.
In a copy constructor accept an object of the current class and initialize the values of instance variables with the values in the obtained object.

(d) List down the access specifier in Java.

Four types .

I. default,
II. public
III. protected
IV. private.

(e) What do you mean by JAVA API?

An application programming interface (API), in the context of Java, is a collection of prewritten packages, classes, and interfaces with their respective methods, fields and constructors. Similar to a user interface, which facilitates interaction between humans and computers, an API serves as a software program interface facilitating interaction.

(f) Difference between function overloading and function overriding.

(g) Classify between final and finally statement.

- The **final** keyword can be used with class method and variable. A final class cannot be instantiated, a final method cannot be overridden and a final variable cannot be reassigned.
- The **finally** keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred. Using a finally block allows you to run any cleanup-type statements that you just wish to execute, despite what happens within the protected code.

(h) What is the role of super statement in constructor?

The super keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

(i) Write the syntax of creating an object of a class in java.

className object = new className();

(j) Define virtual function.

Virtual function in Java is expected to be defined in the derived class. We can call the virtual function by referring to the object of the derived class using the reference or pointer of the base class.

Q. 2.

(a) What do you mean by stream? Describe i/o stream of java.

Stream

A stream is a sequence of data. In Java, a stream is composed of bytes. It's called a stream because it is like a stream of water that continues to flow.

A stream can be defined as a sequence of data. There are two kinds of Streams −

- **InPutStream − The InputStream is used to read data from a source.**
- **OutPutStream − The OutputStream is used for writing data to a destination.**

Depending on the type of operations, streams can be divided into two primary classes:

Input Stream: These streams are used to read data that must be taken as an input from a source array or file or any peripheral device. For eg., FileInputStream, BufferedInputStream, ByteArrayInputStream etc.

InputStream class is an abstract class. It is the superclass of all classes representing an input stream of bytes.

Useful methods of InputStream

Method	Description
1) public abstract int read()throws IOException	reads the next byte of data from the input stream. It returns -1 at the end of the file.
2) public int available()throws IOException	returns an estimate of the number of bytes that can be read from the current input stream.
3) public void close()throws IOException	is used to close the current input stream.

Output Stream: These streams are used to write data as outputs into an array or file or any output peripheral device. For eg., FileOutputStream, BufferedOutputStream, ByteArrayOutputStream etc.

OutputStream class is an abstract class. It is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

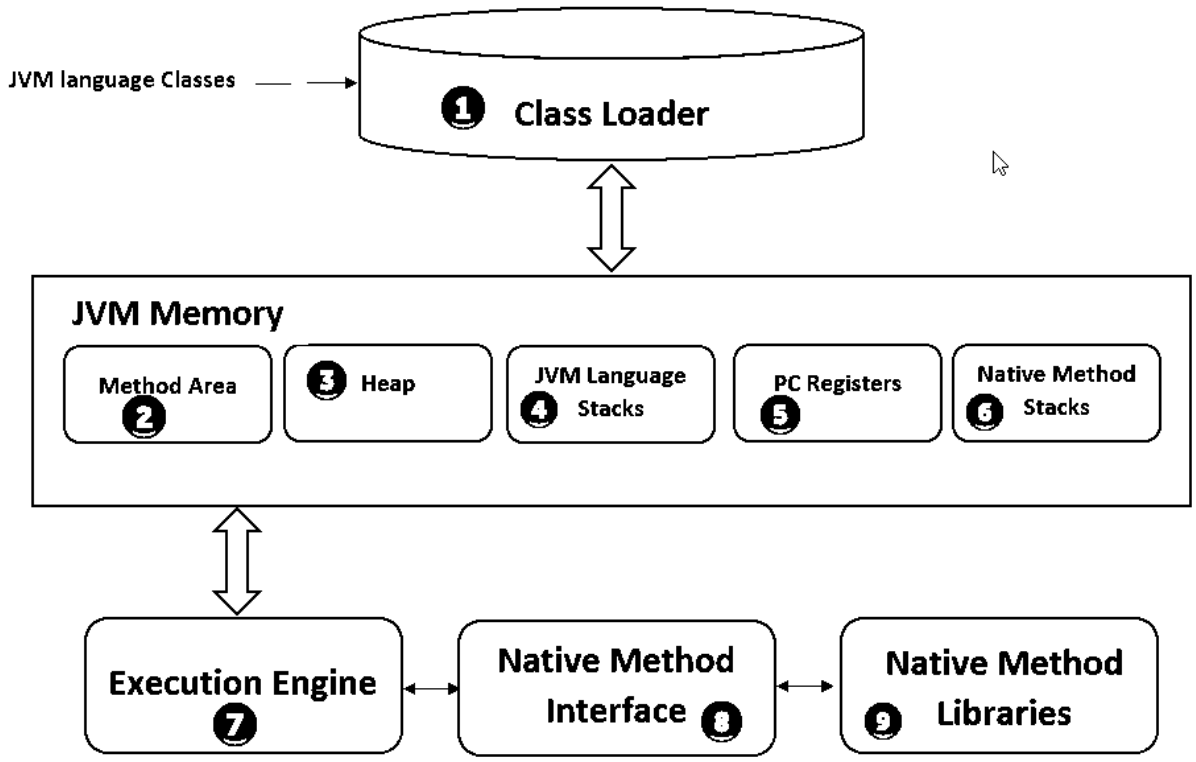
Useful methods of OutputStream

Method	Description
1) public void write(int)throws IOException	**is used to write a byte to the current output stream.**
2) public void write(byte[])throws IOException	**is used to write an array of byte to the current output stream.**
3) public void flush()throws IOException	**flushes the current output stream.**
4) public void close()throws IOException	**is used to close the current output stream.**

(b) Describe the working principle of JVM.

Java Virtual Machine (JVM) is a engine that provides runtime environment to drive the Java Code or applications. It converts Java bytecode into machines language. JVM is a part of Java Run Environment (JRE). In other programming languages, the compiler produces machine code for a particular system. However, Java compiler produces code for a Virtual Machine known as Java Virtual Machine.

- **First, Java code is complied into bytecode. This bytecode gets interpreted on different machines**
- **Between host system and Java source, Bytecode is an intermediary language.**
- **JVM is responsible for allocating memory space.**



1) ClassLoader

The class loader is a subsystem used for loading class files. It performs three major functions viz. Loading, Linking, and Initialization.

2) Method Area

JVM Method Area stores class structures like metadata, the constant runtime pool, and the code for methods.

3) Heap

All the Objects, their related instance variables, and arrays are stored in the heap. This memory is common and shared across multiple threads.

4) JVM language Stacks

Java language Stacks store local variables, and it's partial results. Each thread has its own JVM stack, created simultaneously as the thread is created. A new frame is created whenever a method is invoked, and it is deleted when method invocation process is complete.

5) PC Registers

PC register store the address of the Java virtual machine instruction which is currently executing. In Java, each thread has its separate PC register.

6) Native Method Stacks

Native method stacks hold the instruction of native code depends on the native library. It is written in another language instead of Java.

7) Execution Engine

It is a type of software used to test hardware, software, or complete systems. The test execution engine never carries any information about the tested product.

8) Native Method interface

The Native Method Interface is a programming framework. It allows Java code which is running in a JVM to call by libraries and native applications.

9) Native Method Libraries

Native Libraries is a collection of the Native Libraries(C, C++) which are needed by the Execution Engine.

(c) Explain the concept of overriding with example.

If the same method is defined in both the superclass class and the subclass class, then the method of the subclass class overrides the method of the superclass. This is known as method overriding.

Java Overriding Rules

- **Both the superclass and the subclass must have the same method name, the same return type and the same parameter list.**
- **We cannot override the method declared as final and static.**
- **We should always override abstract methods of the superclass (will be discussed in later tutorials).**

EXAMPLE:-

```
class Human{
  //Overridden method
  public void eat()
  {
    System.out.println("Human is eating");
  }
}
class Boy extends Human{
  //Overriding method
  public void eat(){
    System.out.println("Boy is eating");
  }
  public static void main( String args[]) {
    Boy obj = new Boy();
    //This will call the child class version of eat()
    obj.eat();
  }
}
```

Output:

Boy is eating

(d) Explain the file handling classes and methods.

File handling in Java means that how to read from and write to file in Java. Java provides the basic I/O package for reading and writing streams. java.io package allows to do all Input and Output tasks in Java .

File Handling Methods

Some of the methods are given below for performing different operations in java:

- **createNewFile(): createNewFile method used to create an empty file. It returns the response as boolean.**
- **getName(): This method is used to get the file name. It returns the string i.e. name of the file in response.**
- **getAbsolutePath(): It returns the absolute path of the file. The return type of this method is a string.**
- **canRead(): This method used to check whether the file is readable or not. It returns a boolean value.**
- **canWrite(): This method used to check whether the file is writable or not. It returns a boolean value.**
- **delete(): This method used in deleting a file. It returns a boolean value.**
- **exists(): This method used to check whether a file exists or not. It returns a boolean value.**
- **length(): This method returns the size of the file in bytes. The return type of this method is long.**
- **list(): This method returns an array of the files available in the directory. It returns an array of string values.**
- **mkdir(): This method is used to create a directory. It returns a boolean value.**

(e) Define constructor. Describe the types of constructor.

Constructors in Java

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

Rules for creating Java constructor

1. **Constructor name must be the same as its class name**
2. **A Constructor must have no explicit return type**
3. **A Java constructor cannot be abstract, static, final, and synchronized**

Types of Java constructors

There are two types of constructors in Java:

1. **Default constructor (no-arg constructor)**
2. **Parameterized constructor**

Default Constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

```
class Bike1
{
Bike1()
{
    System.out.println("Bike is created");}

     public static void main(String args[]){
    Bike1 b=new Bike1();

}
}
```

o/p-

Bike is created

Parameterized Constructor

A constructor which has a specific number of parameters is called a parameterized constructor.

```
class Student{
   int id;
   String name;

   Student(int i,String n){
   id = i;
   name = n;
   }
   void display(){System.out.println(id+" "+name);}

   public static void main(String args[]){
   Student s1 = new Student4(111,"Karan");
   Student s2 = new Student4(222,"Aryan");
   s1.display();
   s2.display();
  }
}
```

o/p-

111 Karan

222 Aryan

(f) How we can pass object as parameter?

When we pass a primitive type to a method, it is passed by value. But when we pass an object to a method, the situation changes dramatically, because objects are passed by what is effectively call-by-reference. Java does this interesting thing that's sort of a hybrid between pass-by-value and pass-by-reference. Basically, a parameter cannot be changed by the function, but the function can ask the parameter to change itself via calling some method within it.

- **While creating a variable of a class type, we only create a reference to an object. Thus, when we pass this reference to a method, the parameter that receives it will refer to the same object as that referred to by the argument.**
- **This effectively means that objects act as if they are passed to methods by use of call-by-reference.**
- **Changes to the object inside the method do reflect in the object used as an argument.**

```
import java.util.Scanner;
public class Student {
   private String name;
   private int age;
   public Student(){
   }
   public Student(String name, int age){
      this.name = name;
      this.age = age;
   }
   public Student copyObject(Student std){
      this.name = std.name;
      this.age = std.age;
      return std;
   }
   public void displayData(){
      System.out.println("Name : "+this.name);
      System.out.println("Age : "+this.age);
   }
   public static void main(String[] args) {
      Scanner sc =new Scanner(System.in);
      System.out.println("Enter your name ");
      String name = sc.next();
      System.out.println("Enter your age ");
      int age = sc.nextInt();
```

```
        Student std = new Student(name, age);
        System.out.println("Contents of the original object");
        std.displayData();
        System.out.println("Contents of the copied object");
        Student copyOfStd = new Student().copyObject(std);
        copyOfStd.displayData();
    }
}
```

Output

```
Enter your name
Krishna
Enter your age
20
Contents of the original object
Name : Krishna
Age : 20
Contents of the copied object
Name : Krishna
Age : 20
```

(g) What do you mean by member of a class? How member function is defined in a class?

A class has two types of member.

- **(i) Data Member**
- **(ii) Member function**

A method is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the reusability of code. We write a method once and use it many times. We do not require to write code again and again. It also provides the easy modification and readability of code, just by adding or removing a chunk of code. The method is executed only when we call or invoke it.

```
public class MyClass {
  static void myMethod() {
    // code to be executed
  }
}
```

Example:-

```
public class MyClass {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

  public static void main(String[] args) {
    myMethod();
  }
}
```

Q. 3. Briefly explain concepts of OOP.

Object

Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

Class

Collection of objects **is called class. It is a logical entity.**

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

When one object acquires all the properties and behaviors of a parent object, **it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.**

Polymorphism

If ***one task is performed in different ways***, **it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.**

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Abstraction

Hiding internal details and showing functionality **is known as abstraction. For example phone call, we don't know the internal processing.**

In Java, we use abstract class and interface to achieve abstraction.

Encapsulation

***Binding (or wrapping) code and data together into a single unit are known as encapsulation*. For example, a capsule, it is wrapped with different medicines.**

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

Q. 4. Describe the structure of java program.

```
Package list;
public Class <class name>
{
     Data members;
     Constructor functions;
     User-defined methods;

     public static void main (String arguments[])
     {
          Block of statements;
     }
}
```

Example:-

```
//Name of this file will be "Hello.java"

public class Hello
{
    /*
    Writes the words "Hello Java" on the screen */
    public static void main(String[] args)
    {
        System.out.println("Hello Java");
    }
}
```

Q. 5. Explain the types of inheritance in OOPs.

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

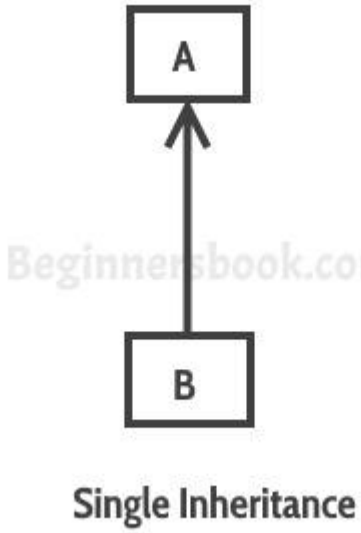
- **subclass (child) - the class that inherits from another class**
- **superclass (parent) - the class being inherited from**

To inherit from a class, use the extends **keyword.**

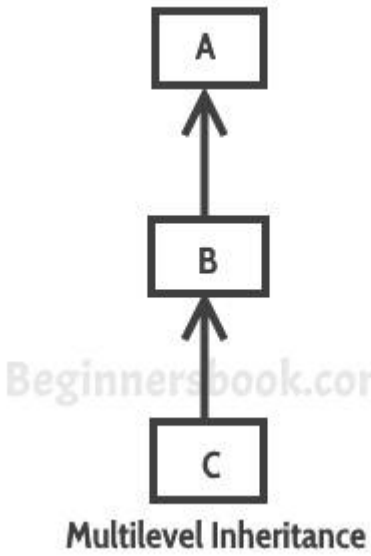
Types of inheritance

Types of Inheritance in Java.

Single Inheritance: refers to a child and parent class relationship where a class extends the another class.

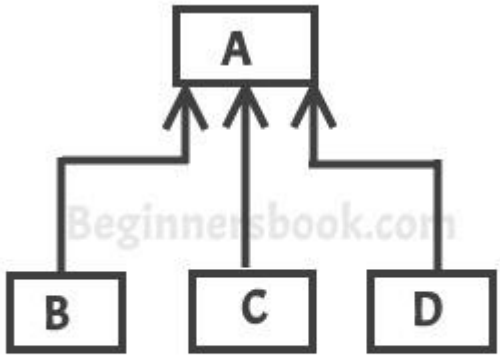


Multilevel inheritance: refers to a child and parent class relationship where a class extends the child class. For example class C extends class B and class B extends class A.



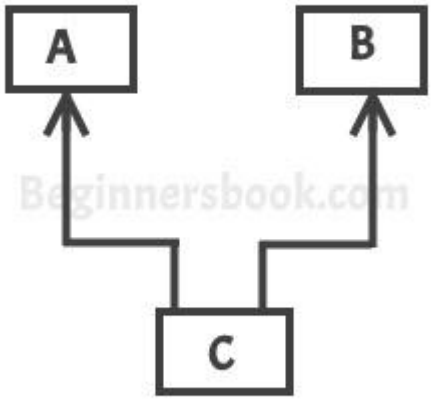
Hierarchical inheritance: refers to a child and parent class relationship where more than one classes extends the same class. For example, classes B, C & D extends the same

class A.



Hierarchical Inheritance

Multiple Inheritance: refers to the concept of one class extending more than one classes, which means a child class has two parent classes. For example class C extends both classes A and B. Java doesn't support multiple inheritance, read more about it here.



Multiple Inheritance

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Hybrid inheritance: Combination of more than one types of inheritance in a single program. For example class A & B extends class C and another class D extends class A then

this is a hybrid inheritance example because it is a combination of single and hierarchical inheritance.

Q. 6. Illustrate the exception handling mechanism in java.

EXCEPTION HANDLING IN JAVA

The **Exception Handling in Java** is one of the powerful ***mechanism to handle the runtime errors*** so that normal flow of the application can be maintained.

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Advantage of Exception Handling

The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application that is why we use exception handling

Java Exception Keywords

There are 5 keywords which are used in handling exceptions in Java.

Keyword	Description
try	The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

```
public class JavaExceptionExample{
  public static void main(String args[]){
   try{
```

```
    int data=100/0;
  }catch(ArithmeticException e){System.out.println(e);}
  System.out.println("rest of the code...");
 }
}
```

o/p-

java.lang.ArithmeticException: / by zero
rest of the code...

Q.7. Define package. Write a simple program that describes package.

JAVA PACKAGE

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.

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.

Example:-

```
import java.util.Scanner; // import the Scanner class


class MyClass {
public static void main(String[] args) {
Scanner myObj = new Scanner(System.in);
String userName;
```

```
// Enter username and press Enter
System.out.println("Enter username");
userName = myObj.nextLine();

System.out.println("Username is: " + userName);
}
}
```