Academic Tast No. 1

Course code: CAP615

Course Title: programming in Java

Section: D2112

Roll No.: RD2112A03

Reg. No. 12111670

Name: Jayshri Lal Pandit

Set 2

(1.) Why Java is known as object oriented programming. Define the features of object-oriented programming in detail.

Ans:→ Java is known as object oriented programming language because Java is a kind of programming language that uses object in each of its programs.

In each java program you have to create classes and in the main function of java you have to create objects of the classes.

You can write a c++ program without creating a class but you have to create class and objects in java program that why java is called "purely" object oriented programming language.

All the concepts like inheritance, modularity, polymorphism, and encapsulation in OOPS are supported by Java.

Features, of Java

The fundamental idea behind OOPS is to combine both data and functions into single unit. Such unit is called an object. Class is a template, blueprint or contract that defines what an objects data fields and methods will be. A Java class uses variables to define data fields and methods to define actions. An object represents an entity in the real world that can be distinctly identified. General concepts :-

(1) Objects
(2) Classes
(3) Data Abstractions
(4) Data Encapsulation
(5) Inheritance
(6) Polymorphism
(7) Dynamic Binding
(8) Message Passing

(1.) Objects :-
Objects are basic run time entitles in an object oriented system. They may represent a person, a place, a bank account that a program must handle. They may also be a user

defined functions such as vectors, date and time. program object should be chosen such that they match closely the real world Object example. Elements of Computer user environment are windows, menus and graphics objects. The physical objects are automobile in graphics stimulation, electronic component in circuit designing. The match between programming objects and a real world Objects is a good resulting. Objects offering the ~~resolv~~ revolution in programming.

(2) Classes :-

In oops, objects are the members of classes almost all computer languages have built in data types like integer, float etc. In similar way you can define many objects of same class and the entire code of an object can be made a user define data type with the help of a class. A class is thus a collection of objects of similar type. For eg. Staff and Students are the members of the college.

A class is thus a collection of objects of similar type. for e.g Staff and students are the members of the college. It is significantly useful as it is used to combine the data and operations into a single entity.

③ Data Encapsulation :-

The wrapping up to data and function into a single unit is called as "Encapsulation". The data is not accessible to the outside world and only those functions which are members of class can access the data. These functions provide interface between objects and programs. This insulation of data from direct access by program is called as data hiding.

④ Data Abstraction :-

Data Abstraction refers to the representation of essential features without including background details or explainations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, cost, etc. It avoids undesired side effects of the member data when it is defined out of the class and also protects the international ~~a~~ misuse of important data. Classes efficiently manage the complexity of large programs through encapsulation.

⑤ Inheritance :-

It is a process by which the object of ~~the~~ one class acquires the properties of objects of another class. It supports the concept of ~~in hei the~~ hierarchical classification. For e.g sparrow is apart of class flying birds which again is a part of class bird. The principle behind this is that each desired class shares the common characteristics with the class from which it is desired. In oops, the concept of inheritance provides the idea of reusability if we can add additional features of an existing class without modifying it. This is possible by deriving a new class from existing one. The new class will have the combined features of both the classes.

(6.) Polymorphism :-

Polymorphism means the ability to take more than one from. for e.g. an operation may exhibit different behaviour in different instants. The behavior depends upon the types of data used in the operation. for example consider operation addition for two numbers, the operation will generate sum. If operands are strings the operation would perform a third string by concatenation. To handle function overloading a single function name can be used to handle different numbers and different types of arguments.

(7) ~~Day~~ Dynamic binding :-

Binding refers to a linking procedure called to the code which is to executed in response to call. Dynamic binding means that the code associated with a given procedure called is not known until the time of call at the run time. consider a procedure 'draw' shown in a fig by inheritance every object will have this procedure. This 'draw' produce will be redefined in each class that defines the object. At the run time, the code matching the object under current reference will be called.

⑧ <u>Message communication</u> :-

An Object-oriented program consists of set of objects that communicate with each other. The process of programming in an object oriented language involves following basic steps :-

- Creating classes that defines object and their behavior.
- Creating object from class definition.
- Establishing communication among object.

Objects may communicate with one another by sending and receiving information through functions. A message for an object is a request for execution of a procedure. ~~Theirrefore~~ Therefore it involves a function in receiving object that generates ~~dir~~ desired results.

(2.) Suppose you have string "hello". How many ways you can create this string object ? Analyze with code. Also define Auto-boxing and Unboxing featur in Java.

Ans:- In Java, a string is a sequence of characters. for example suppode we have string "hello" is containing a sequence of characters 'h', 'e', 'l', 'l' and 'o'. we use double goutes to represent in Java.

There is two way to create String in Java :-

(1) By using String literal
(2) By using new Keyword

(1) By using String Literal :-

Demo program to illustrate it :-

```
Class Test
{
    public static void main(String[] args)
    {
        String s = " hello";

    System.out.println(s);
    }
}
```

Output hello

In the above example, we have created one string named $\boxed{s}$. Here we are directly creating strings like primitive types.

Note :- Strings in Java are not primitive types like $\boxed{\text{int}}$, $\boxed{\text{char}}$, etc. Instead, all strings are objects of a predefined class named $\boxed{\text{String}}$. And, all string variables are instances of the $\boxed{\text{String}}$ class.

(2) By using new keyword :-

Since strings in Java are objects, we can create strings using the $\boxed{\text{new}}$ keyword.

// Demo program illustrate it.

```
Class Test
{
    public static void main(String[] args)
    {
        String name = new String("hello");

        System.out.println(name);

    }
}
```

Output hello

In the above example, we have created a string $\boxed{\text{name}}$ using the $\boxed{\text{new}}$ keyword. Here, when we create a string object, the $\boxed{\text{String()}}$ contructor is invoked.
~~To learn more about const~~

Note :- The $\boxed{\text{String}}$ class provides various constructors to create strings.

-: Autoboxing and Unboxing in Java :-

In Java, primitive data types are treated differently so do there comes the introduction of wrapper classes where two components play a role namely Autoboxing and Unboxing. Autoboxing refers to the conversion of a primitive value into an object of the corresponding wrapper class is called autoboxing. For example, converting ~~into~~ int to Integer class. The Java compiler applies autoboxing when a primitive value is :-

(i) passed as a parameter to a method that expects an object of the corresponding wrapper class.
(ii) Assigned to a variable of the corresponding wrapper class.

Unboxing on the other hand refers to converting an object of a wrapper type to its corresponding primitive value. For example conversion of Integer to int. The Java compiler applies to unbox when an object of a wrapper class is:-

(I) passed as a parameter to a method that expects a value of the corresponding primitive type.

(II) Assigned to a variable of the corresponding primitive type.

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

(3.) An 'if' condition' can be used in many ways. Explain all of these ways with the help of suitable example code.

Ans:- There are four types of 'if condition can be used in Java which is given below:—

(i) if statement
(ii) nested if statement
(iii) if-else Statement
(iv) if-else-if statement

(i) If Statement :—

If Statement consists a condition, followed by ~~Syanx~~ Syntax:—

```
if(condition)
{
    Statements;
}
```

// Demo program Illustrate it.

```
class IfStatementExample
{
  public static void main(String args[])
  {
        int num = 70;
        if (num < 100)
        {
            System.out.println("number is less
                                than 100");
        }

      }
}
```

<u>Output</u>

```
number is less than 100
```

(ii) <u>Nested if statement in Java</u>

When there is an if statement inside another if statement then it is called the nested if Statement.

<u>Syntax :-</u>

```
if (condition_1)
{
        Statement1 ;

        if (condition_2)
        {
            Statement2 ;
        }
}
```

Statement1 would execute if the condition_1 is true. Statement2 would be only execute if both the conditions (condition_1 and condition_2) are true.

// Demo program illustrate it.

```
Public class NestedIfExample
{
  public static void main(String [ ] args)
  {
        int num = 70;

        if(num <100)
        {
            System.out.println("number is less
                                   than 100");
          if(num> 50)
          {
              System.out.printth ("number is
                                   greater than
                                        50");
          }
        }
  }
```

output

```
number is less than 100
number is greater than 50
```

(iii) If else statement :-

Syntax :-

```
if (condition)
{
     Statements;
 }
 else
 {
     Statements;
  }
```

The statements inside "if" would execute if the condition is true, and the statements inside "else" would execute if the condition is false.

// Demo program illustrate it.

~~Public~~ ~~Stat~~

```
public class IfElseExample
{
  public static void main (String args[])
  {  int num = 120;
     if (num <50)
   {
        System.out.println(" num is less than
                                     50");
        }
        else
        {
          System.out.println("num~~ber~~ is greater
                          than or equal 50");
          }
     }
   }
```

<u>output</u>

```
num is greater than or equal 50
```

(iv) if-else-if statement

if-else-if statement is used when we ~~used~~ need to check multiple conditions. In this statement we have only one "if" and one "else", however we can have multiple "else if". It is also known as if else if ladder.

<u>Syntax</u> :-

```
if (condition_1)
{
   Statements;
}
else if (condition_2)
```

```
{
   Statements;
 }
else if (condition_3)
{
    Statements;
 }
- - - - -
- - - - - .

 else
 {
     Statements;
  }
```

Note :- The most important point ~~is~~ to note here is that in if-else-if statement, as soon as the condition is met, the corresponding set of statements get executed, rest gets ignored. If none of the condition is met then the statements inside "else" gets executed.

//Demo program illustrate it.

```
Public Class IfElseIfExample
{
  public static void main(String args[])
  {
        int  num = 1234;
        if (num < 100 && num >= 10)
        {
          System.out.println(" Its a two digit number");
        }
        else if (num < 1000  && num >= 100)
        {
          system.out.println(" Its a tree digit
                                        number");
        }
        else if (num <10000 && num >= 1000)
        {
          System.out.println(" Its a four digit
                                        number");
        }
        else if (num <100000 && num >= 10000)
        {
          system.out.println(" Its a five digit
                                        number");
        }
        else
        {
          System.out.println(" number is not
                                 betwen 1 & 99999");
        }
  }
}
```

<u>Output</u>

```
Its four digit number
```