OBJECT ORIENTED PROGRAMMING

OO PARADIGM

OO paradigm considers a program to be a collection of interacting independent objects.

DESIGN ISSUES

Need to reuse the software components as much as possible

Need to modify program behavior with the minimal changes to existing code

Need to maintain independence of different components

THE WAY SOFTWARE COMPONENTS CAN BE REUSED

Extension

Restriction

Redefinition

Abstraction

polymorphism

EXTENSION OF THE DATA OR OPERATIONS

Functionality of queue can be extended to form a double ended queue.

Functionality of window can be extended to form text window.

RESTRICTION

Some authors call them **encapsulation mechanisms**, while others refer to them as **information-hiding mechanisms**

Ex: Functionality of double ended queue can be restricted to that of a queue.

Functionality of rectangle class can be restricted to that of a square.

REDEFINITION

Even if the operations on a new type of data remain essentially the same as those on existing types, it may be necessary to redefine some of them to accommodate new behavior.

Ex: The calculation of area can be redefined for a square class - if a square is obtained from a rectangle, an area or perimeter function may need to be redefined to take into account the reduced data needed in the computation.

ABSTRACTION

The collection of similar operations from two different components into a new component.

Ex: Square, rectangle and circle class can be generalized to the figure class.

POLYMORPHISM

The extension of the type of data that operations can apply to.

Two kinds of polymorphism: overloading and parameterized types.

Ex: Only one print class must be used to be print anything.

- good example is a toString function, which should be applicable to any object as long as it has a textual representation

CLASS, OBJECTS AND METHODS

Class is a user defined data type

Object is an instance of the class

Methods are the functions that manipulate the data contained in the class.

class classname [extends superclassname]

{

    [variable declaration;]

    [method declration;]

}

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
class A
{
    void p()
    {
        System.out.println("A.p");
    }
    void q()
    {
        System.out.println("A.q");
    }
```

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
    void  f()
    {
            p();
            q();
    }
}
class B extends A
{
```

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
    void p()
    {
        System.out.println("B.p");
    }
    void q()
    {
        System.out.println("B.q");
        super.q();
    }
}
```

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
public class virtualExample
{
    public static void main(String[] args)
    {
        A a=new A();
        a.f();
        a=new  B();
        a.f();
    }
}
```

Output

WRITE A PROGRAM

Write a program in java called complex with the following:

a default, parametrized constructor. Methods to get the real and imaginary parts, display and add two complex numbers.

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
class  A
{
      public void p()
      {
            System.out.println(“A.p”);
      }
      public void q()
      {
            System.out.println(“A.q”);
      }
```

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
        public void r()
        {
                p();
                q();
        }
}
class B extends A
{
        public void p()
        {
                System.out.println("B.p");
        }
}
```

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
Class  C extends B
{
        public void q()
        {
                System.out.println("C.q");
        }
        public void r()
        {
                q();
                p();
        }
}
```

WHAT IS THE OUTPUT OF FOLLOWING CODE

```
public class VirtualDemo
    {
        public static void main(String[] args)
        {
            A a=new B();
            a.r();
            a=new  C();
            a.r();
        }
    }
```

OUTPUT

B.p

A.q

C.q

B.p

WRITE A METHOD IN JAVA TO CHECK WHETHER GIVEN TREE IS BINARY SEARCH TREE

SOLUTION

```
class Tree
{
        int data;
        int num;
        Tree rightTree;
        Tree leftTree;
        public static int validate(Tree root)
        {
                if((countChildren(root)>2)
                return 0;
                if(root.rightTree==NULL && root.leftTree==NULL)
                return 1;
                if(root.rightTree==NULL|| root.leftTree==NULL || root.rightTree.data<root.data ||
                root.leftTree.data>root.data)
        return -1;
        Validate(root.rightTree);
        Validate(root.leftTree);
}
}
```

WHAT IS THE OUTPUT

```
public class Foo
{
    public static void main(String[] args)
    {
        try
        {
            return;
        }
        finally
        {
            System.out.println( "Finally" );
        }
    }
}
```

Finally

WHAT IS THE OUTPUT

```
try
{
    int x = 0;
    int y = 5 / x;
}
catch (Exception e)
{
    System.out.println("Exception");
}
catch (ArithmeticException ae)
{
    System.out.println(" Arithmetic Exception");
}
System.out.println("finished");
```

Compilation Error because general exception is followed by specific exception

WHAT IS THE OUTPUT

```
public class X
{
   public static void main(String [] args)
   {
      try
      {
        badMethod();
        System.out.print("A");
      }
      catch (Exception ex)
      {
        System.out.print("B");
      }
      finally
      {
         System.out.print("C");
      }
         System.out.print("D");
}
    public static void
     badMethod()
    {
        throw new Error(); /*
     Line 22 */
    }
}
```

C is printed before exiting with an error message. Error is in higher hierarchy than Exception

SELECT ONE OF THE ANSWERS

```
public class Outer {
    public void
    someOuterMethod() {
    //Line 5 }

    public class Inner { }
    public static void
    main(String[] argv) {
    Outer ot = new
    Outer(); //Line 10

    }
}
```

Which of the following code fragments inserted, will allow to compile?

A. new Inner(); //At line 5

B. new Inner(); //At line 10

C. new ot.Inner(); //At line 10

D. new Outer.Inner(); //At line 10

ANSWER

Option A compiles without problem.

Option B gives error - non-static variable cannot be referenced from a static context.

Option C package ot does not exist.

Option D gives error - non-static variable cannot be referenced from a static context.

WHAT IS THE OUTPUT

```
class  Q{
public static void main(String
      args[ ])
{
  Holder h=new Holder();
   h.held=100;
      h.bump(h);
   System.out.println(h.held);
}
}
```

```
class Holder{
   public int held;
    public void bump(Holder
     theHolder)
{
   theHolder.held++;
}
}
```

WHAT IS THE OUTPUT

```
class  Q{
public static void
    main(String args[ ])
{
  double d=12.3;
Decrementer dec= new
    Decrementer();
Dec.decrement(d);
System.out.println(d);
}
}
```

```
class Decrementer{
Public void
    decrement(double
    decr)
{
    decr- =1.0;
}
}
```

12.3

WHAT IS THE OUTPUT

```
public class Short{
public static void main(String args[ ]){
StringBuffer s=new StringBuffer("HELLO");
if(s.length()>5) &&
    (s.append("there").equals("False")));
System.out.println(s);
```

HELLO

WHAT IS THE OUTPUT

```
public class Ternary{
 public static void main(String args[ ]);
{
    int x=4;    System.out.println(((x>4) ?
    99.99:9));
```

9.0

```
public class ForCont {
public static void main(String args[])
{
outer: for(int i=0;i<2;i++)
{
for(int j=0;j<3;j++)
{
if(i==j)
{
continue outer;
}
System.out.println("i="+i+"j="+j);
}
}
}
}
```

What is the output

i=1 j=0

WHAT IS THE OUTPUT

```
try{
URL u=new URL(s);
System.out.println("SUCCESS");
}
 catch(MalformedURLException e)
{
    System.out.println("And Failure");
}
finally{ System.out.println("Are Part Of");}
System.out.println("Life");
```

And Failure
Are Part Of
Life

REFERENCES

Text book

Kenneth C. Louden and Kenneth Lambert “Programming Languages Principles and Practice” Third edition Cengage Learning Publication.

Reference Books:

Terrence W. Pratt, Masvin V. Zelkowitz “Programming Languages design and Implementation” Fourth Edition Pearson Education.

Allen Tucker, Robert Noonan “Programming Languages Principles and Paradigms second edition Tata MC Graw – Hill Publication.