Name:- Vivek Mishra
Roll :- 21
Div : SE15

PAGE No.
DATE / /

Object oriented Programing

Assignment no. 1

Q.1 Write an example explain the working of >> and >>> operation in java.

Ans >> [Right side] : Binary right shift operator the left operand value is moved right by the number of bits specified by the right operand.

Example:

```
public operator example {
    public static void main (String args[]) {
        System.out.println(10>>2); // 10/2^2 = 10/4 = 2
        System.out.println(20>>2); // 20/2^2 = 20/4 = 5
        System.out.println(30>>3); // 30/2^3 = 30/8 = 3
    }
}
```

output

2
5
3

>>> [zero fill right shift] ; shift right zero fill operator the left operand value is moved right by the no. of bits specified by the specified right operand and shifted value are filled up with zero

Example:

```
public class operationExample {
    public static void main (String args[]) {
        // for positive number >> and >>> works same
        System.out.println(20>>2);
        System.out.println(20>>>2);
```

// for negative number >>> changes parity bit (MSB) to 0.

```
System.out.println(-20>>2);
System.out.println(-20>>>2);
}
}
```

output :

```
5
5
-5
1073741819
```

Q.2.) What is constructor ? Explain with example.

Ans: 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 is called. It calls a default constructor. if there is no constructor available in the class. In such case, Java compiler provide a default constructor by default.

There are 2 types of Java constructor.

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

1) Java Default contructor.

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

• Syntax of default constructor

```
class_name(){}
```

PAGE No.
DATE / /

Vivek Mishra 21 SEIJ

Example:

```
class Box {
/* Here, Box uses a constructor to initialize the
   dimensions of a box. */

class Box {
    double width;
    double height;
    double depth;

// This is the constructor for Box
    Box() {
      System.out.println("Constructing Box")
        width = 10;
        height = 10;
        depth = 10;
    }
    // compute and return volume
    double volume() {
        return width * height * depth;
    }
}

class BoxDemo {
    public static void main(String args[]) {
        // declare, allocate and initialize Box objects
        Box mybox1 = new Box();
        Box mybox2 = new Box();

        double vol;
```

PAGE No.
DATE / /

```
vol = mybox1.volume();
System.out.println("Volume is:" +vol);

//get volume of second box.
    Vol = mybox1.volume();
    System.out.println("Volume is " + vol);
  }
}
```

Output:

```
Constructing Box
constructing Box
Volume is 1000.0
 Volume is 1000.0
```

2.1 Parameterized Constructor:-

*) ~~And~~ the second constructor is parameterized constructors. While the Box() constructor in the preceding example does initialize a Box object, it is not very useful - all boxes have the same dimensions. What is needed is a way to constructor. As you can probably guess this make them much more useful.

Example.

```
/* Here Box uses a parameterized constructor to initialize the
   dimensions of a box */
class Box {
    double width;
    double height;
    double depth;
```

VIVEK MISHRA 21 SE15

```
// This is the constructor for Box.
Box (double w, double h, double d) {
    width = w;
    height = h;
    depth = d;
}
// compute and return volume.
    double volume() {
        return width * height * depth;
    }
}

class BoxDemo1 {
    Public static void main(String args[]) {
        //declare allocate and initialize Box objects
        Box mybox1 = new Box(10, 20, 15);
        Box mybox2 = new Box(3, 6, 9);

        double vol
        vol = mybox1.volume();
        System.out.println("Volume box1 is" + vol);

        vol = mybox2.volume();
        System.out.println("Volume box2 is" + vol);
    }
}
```

OUTPUT

Volume box1 is 3000.0

Volume box2 is 162.0

Vivek Mishra 21 SE15

PAGE No.
DATE / /

Q.3)

Ans.) To create package is quite easy: simply include a package command as the first statement in a java source file. Any classes declared within that file will belong to the specified package. The package statement defines a name space in which classes are stored. if you omit the package statement, the class the class names are put into the default package, which has no name. (This is why you haven't had to worry about packages before now.) While the default package is fine for short, sample programs, it is inadequate for real application. Most of the time, you will define a package for your code.

This is the general form of the package statement:

package pkg;

Here pkg is the name of the package. for example the following statement creats a package called MyPackage.

package MyPackage;

Java uses file system directories to store packages. for example the .class files for any classes you declare to be part of MyPackage must be stored in a directory called MyPackage. Remember that case is significant and the directory name must match the package name exactly.

More than one file can include the same package statement. The Package statement simply specifies to which package the classes defined in a file belong. It does not exclude other classes in other files from being part of that same package. Most real world packages are spread across many files.

Vivek Mishra 21 SEIS

Example:

```
// A simple package

Package MyPack;

class Balance {
    String name
    double bal;

    Balance (String n, double b) {
        name = n
        bal = b
    }

    Void show() {
        if (bal < 0)
            System.out.println ("-->");
        System.out.println(name + ": $" + bal);
    }
}

class AccountBalance {
    Public static void main [String args[]) {
        Balance current[] = new Balance[3];
        current[0] = new Balance ("K.J. Fielding", 123.33);
        current[1] = new Balance ("Will Tell", 157.02);
        current[2] = new Balance ("Tom Jackson", -12.33);
        for(int i=0; i<3; i++)
            current[i].show();
    }
}
```

PAGE No.
DATE / /

call this file AccountBalance.java and put it in the directory called MyPack. Next compile the file make sure that the resulting .class file is also in the MyPack.

Java MyPack.AccountBalance.

As explained AccountBalance is now part of the package MyPack. This means that it cannot be executed by itself. That is, you cannot use this command line

Java AccountBalance.

PAGE No.
DATE / /

Vivek Mishra 21 SE15

Q.4)

Ans. Vector is like the dynamic array which can grow shrink it size it is similar to arraylist, lust with two difference.

- vector is by synchronized
- Java vector contains many legacy methods that are not the part of collection framed.

Java vector class declaration
Public class vector <E>
extend class vector <E>
implement list <t>, clone able serializable.

1) add() : it is used to append the specified element in the given vector
2) add all() : it is used to append all the element in the specified collection to the end of this vector
3) addelement() : it is used to append the specified component to the end of this vector it increases the vectors size by one.
4) capacity() : it is used to get the current capacity of this vater
5) clear() : it is used to delet all the element from this vector
6) clone() : it returns a clone of this vector
7) contain() : it return turn if the vector contain the specific element

PAGE No.
DATE / /

Example

```
import java.util.*;
Public class VectorExample {
    // create a vector
    Vector String > vec = new vector <String>();
// adding elements using add() method of List
vec.add("whale");
vec.add("Iguana");
vec.add("Lion");
vec.add("Dog");
vec.add elements("Elephant");
vec.addelement("wolf");
vec.add.element("cat");
System.out.println("Element are:" + vec);
 }
}
```

Output

Element are : [whale, Iguana, Lion, Dog, Elephant, wolf, cat].