**Name: Rodney Mutwiri**

**Reg:21/04957**

1. **Explain the differences between primitive and reference data types.**

*a primitive variable's information is stored as the value of that variable, whereas a reference variable holds a reference to information related to that variable*.

1. **Define the scope of a variable (hint: local and global variable)**

*Scope of a variable is the part of the program where the variable is accessible.*

1. **Why is initialization of variables require.**

*To initialize a variable is to give it a correct initial value.*

1. **Differentiate between static, instance and local variables.**

| Local Variable | Instance Variable | Static Variable |
| --- | --- | --- |
| *Defined within a method or a code block* | *Defined outside a method at the class level* | *Defined outside a method at the class level* |
| *Is only accessible in the method/code block where it is declared* | *Is accessible throughout the class* | *Is accessible throughout the class* |
| *Remains in memory as long as the method executes* | *Remains in memory as long as the object is in memory* | *Remains in memory as long as program executes* |
| *Does not require any special keyword* | *Does not require any special keyword but any access specifier (private, protected or public) can be specified. Typically, private or protected is used* | *Requires the static keyword to be specified. In addition, any access specifier (private, protected or public) can be specified. Typically, public is used* |
| *Requires to be initialized before it is used* | *Is given default value based on its data type, so does not require to be initialized before it is used* | *Is given default value based on its data type, so does not require to be initialized before it is used.* |

**5. Differentiate between widening and narrowing casting in java.**

*A widening conversion changes a value to a data type that can allow for any possible value of the original data while A narrowing conversion changes a value to a data type that might not be able to hold some of the possible values.*

**6. the following table shows data type, its size, default value and the range. Filling in the missing values.**

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE** | **SIZE (IN BYTES)** | **DEFAULT** | **RANGE** |
| boolean | 1 bit | false | true, false |
| Char | 2 | \u0000’ | ‘\0000’ to ‘\ffff’ |
| Byte | 1byte | 0 | -27 to +27-1 |
| Short | 2bytes | 0 | -215 to +215-1 |
| Int | 4 | 0 | -231 to +231-1 |
| Long | 8bytes | 0L | -9.223…. to 9223... |
| Float | 4 | 00.0f |  |
| Double | 8 | 0.0d | -1.8E+308 to +1.8E+308 |

1. **Explain the importance of using Java packages**

*Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier*.

1. **Explain three controls used when creating GUI applications in Java language.**

*In the New File dialog box, choose the Swing GUI Forms category and the JFrame Form file type. Click Next.*

*Enter NumberAdditionUI as the class name.*

*Enter my. numberaddition as the package.*

*Click Finish.*

1. **Explain the difference between containers and components as used in Java.**

*The class Component is the abstract base class for the non-menu user-interface controls of AWT. A component represents an object with graphical representation. The class Container is the superclass for the containers of AWT. The container object can contain other AWT components.Jul 30, 2019*

1. **Write a Java program to reverse an array having five items of type int.**

package ARRAYREVERSAL;

public class Reverse

{

public static void main(String[] args)

{

int [] ray= {35, 180, 520, 650, 1000};

reverse(ray, ray.length);

}

static void reverse(int x[], int y)

{

int[] B = new int[y];

int C = y;

for (int i = 0; i < y; i++) {

B[C - 1] = x[i];

C = C - 1;

}

// printing the reversed array

System.out.println("So Your Reversed Array is: \n");

for (int A = 0; A< y; A++)

{

System.out.println(B[A]);

}

}

}

**Output**

*So Your Reversed Array is:*

*1000*

*650*

*520*

*180*

35

1. **Programs written for a graphical user interface have to deal with “events.”**

* **Explain what is meant by the term event.**

*An event is anything that can occur asynchronously, not under the control of the program, to which the program might want to respond.*

* **Give at least two different examples of events, and discuss how a program might respond to those events.**

*Click on a Button -*

*Dragging the Mouse -*

**Explain the difference between the following terms as used in Java programming.**

* **Polymorphism and encapsulation**

*Polymorphism allows program code to have different meaning or functions while encapsulation is the process of keeping classes private so they cannot be modified by external*

* **method overloading and method overriding**

**class and interface**

*Method overloading is used to increase the readability of the program. Method overriding is used to provide the specific implementation of the method that is already provided by its super class. Method overloading is performed within class.*

**inheritance and polymorphism**

*Inheritance is a property pertaining to just classes whereas, polymorphism extends itself into any method and/or function. Inheritance allows the derived class to use all the functions and variables declared in the base class without explicitly defining them*

1. **using examples, explain the two possible ways of implementing polymorphism. Show your code in java**.

***Method overloading*** *is the process that can create multiple methods of the same name in the same class, and all the methods work in different ways. Method overloading occurs when there is more than one method of the same name in the class.*

#### **Example of Method Overloading in Java**

package MethodsOverload;

class Shapes {

public void area()

{

System.out.println("Find area ");

}

public void area(int radius)

{

System.out.println("Circle area = "+3.14\*radius\*radius);

}

//TRIANGLE

public void area(double base, double height)

{

System.out.println("Triangle area="+0.5\*base\*height);

}

//RECTANGLE

public void area(int leng, int widt)

{

System.out.println("Rectangle area="+leng\*widt);

}

}

class Main {

public static void main(String[] args) {

Shapes myShape = new Shapes(); // Create a Shapes object

myShape.area();

myShape.area(100);

myShape.area(32.5,72.2);

myShape.area(20,10);

}

}

Output:

*Find area*

*Circle area = 31400.0*

*Triangle area=1173.25*

*Rectangle area=200*

#### **Example of Method Overriding in Java**

***Method overriding*** *is the process when the subclass or a child class has the same method as declared in the parent class*.

### **Example of Method Overriding in Java**

package MethodsOverloadOverride;

class Beast

{

//defining a method

void run()

{

System.out.println("Vehicle is moving");

}

}

//Creating a child class

class C2 extends Beast

{

//defining the same method as in the parent class

void run(){System.out.println("Vehicle is Moving safely");}

public static void main(String args[]){

C2 obj = new C2();

//creating object

obj.run();

//calling method

}

}

Output:

*Vehicle is Moving safely*

1**. With relevant examples, explain the following concepts as used in Java programming.**

**a. Mutable classes.**

**Explain what is meant by mutable class**

*Polymorphism in Java can be defined as the ability of an object to take many forms. This helps us perform the same action in different ways.*

**Write a program that implements the concept of mutable class**

package mutAnimmut;

public class mutable

{

private String s;

mutable (String s)

{

this.s=s;

}

public String getName()

{

return s;

}

public void setName(String coursename)

{

this.s=coursename;

}

public static void main(String[]args)

{

mutable obj= new mutable("I Love Java");

System.out.println(obj.getName());

//Here,we can update the name using the setName method.

obj.setName("Java Training 30Days Challeng");

System.out.println(obj.getName());

}

}

**Output**

***I*** *Love Java*

*Java Training 30Days Challeng*

**b. Immutable classes.**

**Explain what is meant by immutable class**

The immutable objects are objects whose value can not be changed after initialization. We can not change anything once the object is created. For example, primitive objects such as [int](https://www.javatpoint.com/int-keyword-in-java), [long](https://www.javatpoint.com/long-keyword-in-java), [float](https://www.javatpoint.com/float-keyword-in-java), [double](https://www.javatpoint.com/double-keyword-in-java), all [legacy classes](https://www.javatpoint.com/legacy-class-in-java), [Wrapper class](https://www.javatpoint.com/wrapper-class-in-java), [String class](https://www.javatpoint.com/methods-of-string-class), etc.

**Write a program that implements the concept of immutable class**

package mutAnimmut;

public class Immutable

{

private final String s;

Immutable (final String s)

{

this.s=s;

}

public final String getName()

{

return s;

}

public static void main(String[]args)

{

Immutable obj= new Immutable("I Love Java");

System.out.println(obj.getName());

}

}

c. E**xplain the situations where mutable classes are more preferable than immutable classes when writing a Java program.**

when dealing with multi-thread environments.

2. **Explain what a String buffer class is as used in Java**

StringBuffer is a peer class of String that provides much of the functionality of strings.

the syntax of creating an object of StringBuffer class

Syntax: StringBuffer ob = new StringBuffer(String); StringBuilder(char sequence)

**Explain the methods in the StringBuffer class**

|  |  |
| --- | --- |
| Modifier and Type | Method |
| public synchronized StringBuffer | append(String s) |
| public synchronized StringBuffer | insert(int offset, String s) |
| public synchronized StringBuffer | replace(int startIndex, int endIndex, String str) |
| public synchronized StringBuffer | delete(int startIndex, int endIndex) |

**Write the output of the following program.**

class Myoutput

1. {

2. public static void main(String args[])

3. {

4. String ast = "hello i love java";

5. System.out.println(ast.indexOf('e')+" "+ast.indexOf('ast')+" "+ast.lastIndexOf('l')+" "+ast.lastIndexOf('v'));

6. }

7. }

**Ans:**

*1 -1 8 15*

b. **Explain your answer in (2b) above.**

**Ans:**

*ast.indexOf('e') will output index number for the first ‘e’ in the sentence. Count starts from 0*

*ast.indexOf('ast') will output index for the variable of type string which has stored the sentence, meaning it will count backwards from -1.*

*ast.lastIndexOf('l') will output the index number for the last ‘l’ in the sentence.*

*ast.lastIndexOf('v') will output the index number for the last ‘v’ in the sentence*

d. **With explanation, write the output of the following program.**

class Myoutput

1. {

2. public static void main(String args[])

3. {

4. StringBuffer bfobj = new StringBuffer("Jambo");

5. StringBuffer bfobj1 = new StringBuffer(" Kenya");

6. c.append(bfobj1);

7. System.out.println(bfobj);

8. }

9. }

**Ans:**

*The result is an error, changing variable c to bfobj will get a result of “Jambo Kenya”. Append function will combine the two statements into one*

**e. With explanation, write the output of the following program.**

class Myoutput

1. {

2. public static void main(String args[])

3. {

4. StringBuffer str1 = new StringBuffer("Jambo");

5. StringBuffer str2 = str1.reverse();

6. System.out.println(str2);

7. }

8. }

**Ans:**

*The output is obmaJ, the reverse() function reverses the string Jambo to strat from o.*

**f. With explanation, write the output of the following program.**

class Myoutput

1. {

2. class output

3. {

4. public static void main(String args[])

5. {

6. char c[]={'A', '1', 'b' ,' ' ,'a' , '0'};

7. for (int i = 0; i < 5; ++i)

8. {

9. i++;

10. if(Character.isDigit(c[i]))

11. System.out.println(c[i]+" is a digit");

12. if(Character.isWhitespace(c[i]))

13. System.out.println(c[i]+" is a Whitespace character");

14. if(Character.isUpperCase(c[i]))

15. System.out.println(c[i]+" is an Upper case Letter");

16. if(Character.isLowerCase(c[i]))

17. System.out.println(c[i]+" is a lower case Letter");

18. i++;

19. }

20. }

21. }

**Ans:**

*1 is a digit*

*a is a lower case Letter*