JAVA PROGRAMMING TASK THREE

NAME: Marion Kariuki
REG NO: 21/07779

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

PRIMITIVE DATA TYPE	REFERENCE DATA TYPE
1. They are the data types provided by java	They are data types created by a programmer.
2. They always have a value.	They can be null
3. The size depends on the data type.	They are all the same size
4. It starts with a lowercase letter.	It starts with an uppercase letter.

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

It is a block of code in the entire program where the variable is declared, used and can be modified.

3. **Why is initialization of variables required**

Without initialization, a variable would have an unknown value, which can lead to unpredictable outputs when used in computations or other operations.

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

   A local variable is defined within a method or a code block while an instance variable is defined outside a method at a class level and a static variable is a variable which belongs to the class and is initialized only once at the start of the execution.

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

Widening casting is the conversion of a smaller data type to a larger type size while Narrowing casting involves converting a larger data type to a smaller size type.

6. **The following table shows the 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	**1**	0	-2^7 to +2^7-1
Short	**2**	0	-2^15 to +2^15-1
Int	4	**0**	-2^31 to +2^31 - 1
Long	**8**	0L	**-2147483648 to 2147483647**
Float	4	00.0f	**3.4E-38 to 3.4E+38**
Double	8	**0.0d**	-18E +308 + 1.8E + 308

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

- Package creates a new namespace so there won't be any name conflicts with names in other packages.
- They Make it easy to search or locate classes and interfaces.
- By using packages, it is easier to provide access control
- Programmers can define their own packages to bundle a group of classes/interfaces, etc.

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

- Label - Is used to provide a descriptive text string that cannot be changed directly by the user.
- TextField - Used to get text input from the user into the program for processing.
- Button - Used to execute blocks of code in a program when clicked by the user.
- Checkbox - Used to display options to the user, where the user can select more than one option.

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

A component is an object, like a button or a scroll bar. Examples of components are JLabel, JTextField, and JButton while containers are window-like components that can contain other components. Examples of containers are JPanel, JFrame, JApplet

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

```
// Reversing an array using Java collections

import java.util;

public class reversingArray {

     // function reverses the elements of the array
     static void reverse(Integer a[])
     {
          Collections.reverse(Arrays.asList(a));
          System.out.println(Arrays.asList(a));
     }

     public static void main(String[] args)
     {
          Integer [] arr = {1, 2, 3, 4, 5};
```

```
            reverse(arr);
        }
}
```

11. **Programs written for a graphical user interface have to deal with "events." Explain what is meant by the term event. Give at least two different examples of events, and discuss how a program might respond to those events.**

An event is anything that can occur asynchronously, not under the control of the program, to which the program might want to respond. GUI programs are said to be "event-driven" because, for the most part, such programs simply wait for events and respond to them when they occur. In many (but not all) cases, an event is the result of a user action, such as when the user clicks the mouse button, types a character, or clicks a button. The program might respond to a mouse click on a canvas by drawing a shape, to a typed character by adding the character to an input box, or to a click on a button by clearing a drawing. More generally, a programmer can set up any desired response to an event by writing an event-handling routine for that event.

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

    - ***Polymorphism and encapsulation*** - Encapsulation is the mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit while polymorphism is the ability to create a variable, a function, or an object that has more than one form.

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

    - ***class and interface*** - A class is a user-defined blueprint or prototype from which objects are created while an interface can have methods and variables like a class, but the methods declared in the interface are by default abstract (only method signature, no body).

    - ***Inheritance and polymorphism*** - Inheritance is one in which a new class is created (derived class) that inherits the features from the already

existing class(Base class) Whereas polymorphism is that which can be defined in multiple forms.

13. Explain the two possible ways of implementing polymorphism. Show your code in java.

- **Method overloading**

It 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;

```
class overloading {
 public void area() {
System.out.println("Find area ");
}
 public void area(int r) {
System.out.println("Circle area = "+3.14*r*r);
}

 public void area(double b, double h) {
System.out.println("Triangle area="+0.5*b*h);
}
 public void area(int l, int b) {
System.out.println("Rectangle area="+l*b);
}


}

class Main {
public static void main(String[] args) {
overloading example = new overloading();

example.area();
example.area(5);
example.area(6.0,1.2);
example.area(6,2);

}
```

```
}
```

- **Method overriding**

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

Example;

```
class Vehicle{
//defining a method
void run(){System.out.println("Vehicle is moving");}
}
//Creating a child class
class Car2 extends Vehicle{
//defining the same method as in the parent class
void run(){System.out.println("car is running safely");}

public static void main(String args[]){
Car2 obj = new Car2();//creating object
obj.run();//calling method
}
}
```

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

- **Mutable classes**

A mutable class is one that can change its internal state after it is created.

Write a program that implements the concept of mutable class

```
public class example {
private String coursename;
example(String coursename) {
```

```
this.coursename = coursename;
}
public String getName() {
return coursename;
}
public void setName(String coursename) {
this.coursename = coursename;
}
public static void main(String[] args) {
example obj = new example("Information Technology");
System.out.println(obj.getName());

// update the name, this object is mutable
obj.setName("Diploma Information Technology");
System.out.println(obj.getName());

}
}
```

- **Immutable classes.**

An immutable class is one that can not change its internal state after it is created.

Write a program that implements the concept of immutable class

```
public class immutable {
private final String unitname;
immutable(final String unitname) {
this.unitname =unitname;
}
public final String getName() {
return unitname;
}
public static void main(String[] args) {
immutable obj = new immutable("Java programming");
System.out.println(obj.getName());

}
```

```
}
```

- **Explain the situations where mutable classes are more preferable than immutable classes when writing a Java program.**

- Mutable objects can be changed to any value or state without adding a new object
- Mutable objects provide a method to change the content of the object.
- The mutable objects support the setters and getters both.

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

A string buffer class is used to create mutable (modifiable) string.

The syntax of creating an object of StringBuffer class

StringBuffer sb = new StringBuffer();

Explain the methods in the StringBuffer class

1. capacity() - the total allocated capacity can be found by the capacity( ) method.
2. append() - Used to add text at the end of the existing text.
3. length() - The length of a StringBuffer can be found by the length( ) method.
4. charAt() - This method returns the char value in this sequence at the specified index.
5. delete() - Deletes a sequence of characters from the invoking object

(b) Write the output of the following program.

```
class Myoutput
{
    public static void main(String args[])
    {
        String ast = "hello i love java";
        System.out.println(ast.indexOf('e')+" "+ast.indexOf('ast')+"
"+ast.lastIndexOf('l')+" "+ast .lastIndexOf('v'));
    }
}
```

Answer: The program has no output

(C) **Explain your answer**

In the above code we have ast.indexOf('ast'). indexOf() does not take a String argument hence resulting in an error.

(D) **With an explanation, write the output of the following program.**

```
class Myoutput
{
    public static void main(String args[])
    {
        StringBuffer bfobj = new StringBuffer("Jambo");
        StringBuffer bfobj1 = new StringBuffer(" Kenya");
```

```
                c.append(bfobj1);
                System.out.println(bfobj);
            }
        }
```

Answer: The program does not run because of an error in line 6. “c.append(bfobj1);”. The variable “c” was not created.

(E). **With an explanation, write the output of the following program.**

```
class Myoutput
    {
      public static void main(String args[])
      {
        StringBuffer str1 = new StringBuffer("Jambo");
        StringBuffer str2 = str1.reverse();
        System.out.println(str2);
      }
    }
```

Answer:
Output: obmaJ

This is because the original str1 having “Jambo” has been reversed by the reverse() function and transferred to the str2 variable that is later printed.

(F). **With an explanation, write the output of the following program.**

```
class Myoutput
    {
      class output
      {
        public static void main(String args[])
        {
          char c[]={'A', '1', 'b' ,' ' ,'a' , '0'};
```

```
            for (int i = 0; i < 5; ++i)
            {
                i++;
                if(Character.isDigit(c[i]))
                   System.out.println(c[i]+" is a digit");
                if(Character.isWhitespace(c[i]))
                   System.out.println(c[i]+" is a Whitespace character");
                if(Character.isUpperCase(c[i]))
                   System.out.println(c[i]+" is an Upper case Letter");
                if(Character.isLowerCase(c[i]))
                   System.out.println(c[i]+" is a lower case Letter");
                i++;
            }
        }
    }
```

Answer:

Output:

1 is a digit

a is a lower case Letter

At the first loop, we check if the second value is a digit, whitespace, an uppercase or lowercase. Since it is “1”, then it is a digit, and we print it to the console.

We then skip the third value and check the fourth value if it is a digit, whitespace, uppercase or lowercase. Since the fourth value is “a”, then it is lowercase, and we print to the console.

“I” is incremented two times in the loop.