Name:Mohamed ibrahim
Reg No: 21/06819
Java task 3

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

Data types specifies the different sizes and values that can be stored in variable.

Primitive data types: This are set of data types from which all other data types are constructed. The primitive data types include Boolean, char, byte, short, int, long, float and double.
In java language, primitive data types are the building blocks of data manipulation, it's the most basic data types available in java language.

Reference data types: This are data types in java that contains reference/address of dynamically created objects. These are not predefined like primitive data types.

This examples of Reference data types;

Class types − this reference type points to an object of a class.
Array types − this reference type points to an array.
Interface types − this reference type points to an object of a class which implements an interface.

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

A scope is a region of a program where variables can be declared.

<u>Local variables:</u> This are variables that are declared inside a function or block. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own.

<u>Global variables:</u> These are variables that are defined outside of all the functions, usually on top of the program. The global variables will hold their type throughout the program.

This variable can be accessed by any function

3. Why is initialization of variables required?

Every variable in java should be properly initialized, this gives it a correct initial value, failure to do so java will indicate an error has occurred, telling you to initialize a variable.

4. Differentiate between static, instance and local variables.

Static variable: This are variables that can be shared among all instances of a class. Most of the time starts with the keyword Static.

Instance variable:

This are variables declared inside the class but outside the body of the method.

Local variable: This are variables declared inside the body of the method, it can be useful only within that method.

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

Widening type conversion can happen if both types are compatible and the target type is larger than source type whereas **Narrowing typecasting** is conversion of higher data type to lower data type.

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	8 bits	0	-2^{7} to $+2^{7}-1$
Short	32 bits	0	-2^{15} to $+2^{15}-1$
Int	4	-2,-1,0,1,2	-2^{31} to $+2^{31}-1$
Long	64 bits	0L	-9,223,372,036,854,775,808 To 9, 223,372,036,854,775,807
Float	4	00.0f	Up to seven decimal digits
Double	8	0.0	-1.8E+308 to +1.8E+308

7. *Explain the importance of using Java packages*

A java package is used to categorize the classes and the interfaces so that they can easily be maintained, it also access protection and removes naming collision.

8. *Explain three controls used when creating GUI applications in Java language.*
9. *Explain the difference between containers and components as used in Java.* A

Container is a component that can accommodate other components and also other containers, Container provide the support for building complex hierarchical graphical user interface.

A **component** class is found under java.awt package. The container class is the sub class of the component class. The component class defines a number of methods for handling events, changing window bounds, controlling fonts and drawing components and their content.

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

```
// Basic Java program that reverses an array
 public class reverseArray
   {

    // function that reverses array and stores it
           // in another array     static
       void reverse(int a[], int n)
           {         int[] b = new
      int[n];
           int j = n;
           for (int i = 0; i < n; i++) {
             b[j - 1] = a[i];
  j = j - 1;
         }

         // printing the reversed array
         System.out.println("Reversed array is: \n");
 for (int k = 0; k < n; k++) {
           System.out.println(b[k]);
        }
    }
```

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 can be defined as changing the state of an object or behavior by performing actions. Actions can be button click, cursor movement etc.

Example of events are; Mouse events-This event occurs when the mouse pointer moves while it's over control. The handler for this event receives an argument of type MouseEventArgs.

Touch events- Touch events consist of three interfaces (Touch , TouchEvent and TouchList) and the following event types: touchstart - fired when a touch point is placed on the touch surface. touchmove - fired when a touch point is moved along the touch surface. touchend - fired when a touch point is removed from the touch surface.

12. *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 codes *method overloading and method overriding*-- Overriding occurs when the method

signature is the same in the superclass and the child class. Overloading occurs when two or more methods in the same class have the same name but different parameters.

class and interface-- class can be inherited by another class using the keyword 'extends'. An Interface can be inherited by a class using the keyword 'implements' and it can be inherited by another interface using the keyword 'extends'. A class can contain constructors. An Interface cannot contain constructors.

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

Polymorphism in Java can be achieved in two ways i.e., **method overloading and method overriding**. Polymorphism in Java is mainly divided into two types. Compile-time polymorphism can be achieved by method overloading, and Runtime polymorphism can be achieved by method overriding

Example of compile time polymorphism;

```
public class Main {
   public static void main(String args[]) {
      CompileTimePloymorphismExample obj = new CompileTimePloymorphismExample();
obj.display();      obj.display("Polymorphism");
   }
}
class    CompileTimePloymorphismExample   {    void
display() {
      System.out.println("In Display without parameter");
   }
   void display(String value) {
      System.out.println("In Display with parameter" + value);
   }
}
```

Example of runtime polymorphism;

```
public class Main {
   public static void main(String args[]) {
      RunTimePolymorphismParentClassExample obj = new
RunTimePolymorphismSubClassExample();
      obj.display();
   }
}


class    RunTimePolymorphismParentClassExample    {    public void display() {
      System.out.println("Overridden Method");
   }
}


public class RunTimePolymorphismSubClassExample extends
RunTimePolymorphismParentExample {


   public void display() {
    System.out.println("Overriding Method");
```

Part two

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

Mutable classes.

Explain what is meant by mutable class-- A mutable class is one that can change its internal state after it is created. Generally speaking, a class is mutable unless special effort is made to make it immutable.

Write a program that implements the concept of mutable class

```
public class JtpExample {
private String s;
JtpExample(String s) {      this.s
= s;
   }
   public String getName() {
return s;
   }
   public void setName(String coursename) {
this.s = coursename;
   }
   public static void main(String[] args) {
   JtpExample obj = new JtpExample("Kca university");
   System.out.println(obj.getName());
// Here, we can update the name using the setName method.
   obj.setName("Java studies");
   System.out.println(obj.getName());
   }
   }
```

b. Immutable classes.

Explain what is meant by immutable class--
Immutable class in java means that once an object is created, we cannot change its content

Write a program that implements the concept of immutable class

```
public class Immutable {
private final String s;
JtpExample1(final String s) {          this.s
= s;
        }
        public final String getName() {
return s;
        }
        public static void main(String[] args) {
        JtpExample obj = new JavaExample("Java example");
        System.out.println(obj.getName());
        }
        }
```

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

The mutable objects are objects whose value can be changed after initialization. object's values can be changed, such as field and states, after the object is created. For example,

When   change is made in existing mutable objects, no new object will be created; instead, it will alter the value of the existing object. These object's classes provide methods to make changes in it thus making it more preferable than immutable classes.

2. *a)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. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences. StringBuffer may have characters and substrings inserted in the middle or appended to the end. the syntax of creating an object of StringBuffer class **class** StringBufferExample{   **public static void** main(String args[]){   StringBuffer sb=**new** StringBuffer("Hello ");   sb.append("Java");//now original string is changed

```
      System.out.println(sb);//prints Hello Java
      }
      }
```

Explain the methods in the StringBuffer class

It is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc. It is used to insert the specified string with this string at the specified position.

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'));

         }

       }
```

Output

```
$ javac output.java
$ java output  1
14 8 15
```

c. Explain your answer in (2b) above. indexof('c') and lastIndexof('c') are pre defined function which are used to get the index of first and last occurrence of the character pointed by c in the given array.

d. With 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);

}

}

Output

```
$ javac output.java
$ java output
Jambo kenya
```

append() method of class StringBuffer is used to concatenate the string representation to the end of invoking string.

e. With 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);

}

}

Output

```
$ javac output.java
$ java output obmaJ
```

reverse() method reverses all characters. It returns the reversed object on which it was called. f.

. With 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++;
                 }
              }
```

Output

```
$ javac output.java
$ java output
is a digit a is a
lower case        etter
```

Character.isDigit(c[i]), Character.isUpperCase(c[i]), Character.isWhitespace(c[i]) are the function of library java.lang they are used to find whether the given character is of specified type or not. They return true or false i:e Boolean variable.