COURSE: DIPLOMA IN IT [STAGE FOUR]

UNIT CODE: DIT 409

UNIT NAME: JAVA PROGRAMMING

TASK THREE

REGISTRATION NO: 21/06157

SUBMITTED TO: MR. JOHN KIMOTHO

TASK 3 ASSIGNMENT

QUESTION 1: Explain the differences between primitive and reference data types.

- Primitive data types are already defined in java e.g.; Boolean, char etc while reference data types while reference data types are created or defined by the users in java e.g. arrays , class etc.

QUESTION 2: Define the scope of a variable.

- The scope of a variable is a location or region of the program where the variable is visible to a program and can be accessible.
- The scope of variables can be defined with their declaration, in which they are declared in two ways.

Global variable: can be accessed outside all the functions/or globally in the entire program and **Local variable:** can be accessed within the function or block in which they are defined.

QUESTION 3: Why is initialization required?

It is required because without initialization, a variable would have an unknown value, which can lead to unpredictable outputs when used in computation or other operations.

QUESTION 4: Differentiate between static, instance and local variables.

A *Static variable* is defined outside a method at class level, an *instance variable* is defined outside a method at class level and *local variable* is defined within a method or block.

A *static variable* remains in memory as long as the program executes, instance *variable* remains in memory as long as the objects is in the memory and the *local variable* remains in the memory as long as the method executes.

QUESTION 5: Differentiate between widening and narrowing casting in java.

Widening casting involves the conversion of a smaller data type to a larger data type.

Narrowing casting involves converting a larger data type to a smaller data type

QUESTION 6: The following table shows data type, size default value and the range. Fill in the missing values.

TYPE	**SIZE(IN BYTES)**	**DEFAULT**	**RANGE**
Boolean	1bit	*false*	True, false
Char	2	*'\u0000'*	'\u0000' 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	*-9,223,372,036,854,775,808to 9,223,7372,036,854,775,807*
Float	4	00.0f	*3.4-38 to 3.4E+38*
Double	8	*0d*	-1.8E +308 to +1.8E+308

QUESTION 7: Explain the importance of using Java Packages.

1. To group the related classes
2. Since the Java package creates a new name space there won't be any name conflicts within the names of the packages.

3. Programmers can define their own packages to bundle a group of classes/interfaces.
4. Using packages it is easier to provide access control.

QUESTION 8: Explain three controls used when creating GUI applications in java language.

1. **Label;** It is used for inputting text/defining a descriptive text.
2. **Button:** It is used to execute codes in a program when clicked by the user.
3. **Textfield;** Used to get the input from the user in the form of text

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

Containers are components in the AWT that contains other components while **component** is the abstract class for the non-menu user interface controls of AWT it represents an object with graphical components.

QUESTION 10: Write a program to reverse an array having five items of type int.

```
Public class ReverseArray{

   Public static void main (String[] args){

      //initialize array

     Int[]arr = new int[] {1,2,3,4,5};

     System.out.println{“Original array; ”);

     for (int i = 0; i<arr.length;i++){

     System.out.print(arr[i]+””);

}

     System.out.println();

      System.out.println{“Array in reverse order; ”);

     //Loop through the array in reverse order

     for (int i=arr.length-1; i=>=0;i--){

     System.out.print(arr[i]+””);
```

```
    }
  }
}
```

QUESTION 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.

Event: Is changing the state of an object or the behavior by performing actions, these actions can be cursor movement, a button click , key press through keyboard or page scrolling.

Examples:

- When a user clicks a button the application generates an Onclick event to which you can respond with an event handler.
- When a scrollbar is being scrolled the onscroll event occurs.

QUESTION 12: Explain the difference between the following terms as used in java.

Polymorphism and encapsulation

- **Polymorphism** is the ability of a class to provide different implementations of a method depending on the type of object that is passed to the method while **Encapsulation** is the process of keeping classes private so they cannot be modified by external codes.

Method overloading and Method overriding

- **Method overloading** is a concept in java which can create multiple methods of the same class and all the methods work in different ways while **method overriding** occurs when a subclass provides a particular implementation of a method declared by one of its parent classes.

Class and interface

- A **class** can inherit another class while an **interface** cannot inherit a class.
- A **class** can be inherited by another class using keyword 'extends' while an **interface** can be inherited by a class using the keyword '*implements*' and it can be inherited by another interface using the keyword '*extends*'.

Inheritance and polymorphism

- **Inheritance** supports the concept of reusability and reduces code length in object oriented programming while **polymorphism** allows the object to decide which form of the function to implement at compile-time (overloading) as well as run- time (overriding).

QUESTION 13: Using examples, explain the two possible ways of implementing polymorphism. Show your code in java.

1. **Polymorphism using method overriding.**

- **If the same method is present in both subclass and superclass then the method in the subclass overrides the same method hence called method overriding. This method performs one operation in the super class and another operation in subclass.**
- **The method that is called is determined during the execution of the program hence method overriding is a run-time polymorphism.**

- **Example;**

```
class Language
  public void displayInfo(){
    System.out.println("Common English Language");
  }
}
class Java extends Language{
@Override
Public void displayInfo(){
System.out.println("Java programming Language")
  }
}

class Main{
 public static void main (String[] args){

//create an object of java class
Java j1 = new Java();
j1.displayInfo();

//create an object of Language class
Language l1 = new Language();
```

```
l1.displayInfo();
   }
}
```

Output:

Java Programming Language
Common English Language

2. **Polymorphism using method overloading**

- **In this method it will perform different operations based on the parameter.**
- **The method that is called is determined by the compiler hence it is also known as the compile time polymorphism.**

Example:

```
class Pattern{
//method without parameter
public void dispay(){
    for(int i = 0; i <10; i ++){
      System.out.print("@");
}}
//method with single parameter
public void display (char symbol){
 for (int i = 0; i <10; i ++){
      System.out.print(symbol);
    }
  }
}
class Main{
  public static void main (String [] args){
    Pattern d1 = new Pattern();

//call method without argument
d1.display();
System.out.println("\n");

//call method with a single argument
d1.display('#');
   }
```

```
}
```

Output:

```
@@@@@@@@@@
##########
```

In this example we created a class named Pattern and the class contains a method named display() that is overloaded .The main function display() is to print the pattern .However based on the arguments passed, the method is performing different operations.

- ✓ **Prints a pattern of @, if no argument is passed**
- ✓ **Prints pattern of the parameter. if a single char type argument is passed**

PART B

QUESTION 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/mutate its internal state after it is created.

Write a program that implements the concept of mutable class.
(Solution):

```
public class JpExample{
   private String s;
     JpExample(String s){
     this.s= s;
     }
     public String getName(){
     return s;
     }
    public void setName (String coursename){
    this.s = coursename;
    }
```

```
public static void main (String[] args){
JpExample obj = new JpExample(" Diploma in Information Technology");
System.out.println(obj.getName());
 //here we can update the name using the setName
obj.setName("Application Programming")
System.out.println(obj.getName());
}}
```

b. Immutable class

Explain what is meant by immutable class.

- An Immutable class is one that cannot change the internal content once the object is created.

Write a program that Implements the concept of immutable class.

(Solution);

```
public class JpExample1{
private final String s;
JpExample1(final String s){
this.s = s;
}
public final String getName(){
return s;
}
Public static void main(String [] args){
JpExampleobj= new JpExample("Advanced Application Programming");
System.out.println(obj.getName());
}}
```

c. Explain the situations in which mutable classes are more preferable than immutable classes when writing a java program.

- When transforming a class from state A to state Z would produce a lot of intermediate objects which would rather not spend time creating.
- Passing mutable objects into methods lets you collect multiple results without jumping through syntactic hoops.
- One can make changes to the program,without allocating a new object.
- Functional programming methods are pretty effective with mutable classes unlike immutable classes which may waste memory and garbage collector time as changing of data will generate wasted instances.

QUESTION 2:

a) Explain what a String Buffer class is as used in java, the syntax of creating an object of string Buffer class and explain the methods in the String buffer class.

➢ A String Buffer class is used to create mutable (modifiable) string objects.

- The syntax of creating an object of StringBuffer class : **StringBuffer str= new StringBuffer(String str);**
- Methods in the String Buffer class;

1. capacity() method– used to return the current capacity.
2. append () method – it is used for appending the new sequence to the existing sequence of the String Buffer class.
3. reverse() method- it is used to reverse the current string.
4. Length() method- is used to return the length of the string i.e the total number of characters.
5. deleteCharAt() method- it is used for deleting the particular character at the given index from the StringBuffer sequence.
6. Insert()method-inserts the given string with the string at the given 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:

The program has no output

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

In this code we have ast.indexOf ('ast')and indexOf() does not take a String argument hence resulting to an error.

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.  }
```

Explanation;
The program does not run because of an error in line 6. "c.append(bfobj1);".
The variable "c" was not created.

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.  }
```

Output: **obmaJ**

Explanation;
This is because the original str1 having "Jambo" has been reversed by the reverse() method and transferred to the str2 variable that is later printed.

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.  }
```

Output:

1 is a digit

a is a lower case Letter

Explanation;

At the first loop, we check if the second value is a digit, a whitespace character, an uppercase or lowercase letter. Since it's "1", then it is a digit, and we print to the console. We then skip the third value, and check the forth value and if it is a digit, a whitespace character, an uppercase or lowercase. Since the forth value is "a", then it is a lowercase letter, and we print to the console.

"i" is incremented two times in the loop.