21/06034 – Samuel Wakoli

Task 3

Part 1

[Open online compiler]

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

- Primitive data types are already defined in Java while reference data types can be defined by the user.
- Primitive data types specify the size and type of variable values while reference data types specify the reference/address of the variable values.
- Primitive data types store one type of data while reference data types can store different types of data.
- Primitive data types are considered to be storage locations declared in a memory, while reference data types are considered to be objects.

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

A scope is a region of the program. It is where variables can be declared: Inside a function or a block which is called *local variables*, In the definition of function parameters which is called formal parameters. Outside of all functions which are called *global variables*.

3. Why is initialization of variables required?

- To ensure null safety.
- To avoid run-time errors.

4. Differentiate between static, instance and local variables.

Static Variable	**Instance Variable**	**Local Variable**
Is accessible throughout the class	Is accessible throughout the class	Is only accessible in the method/code block where it is declared
Remains in memory as long as program executes	Remains in memory as long as the object is in memory	Remains in memory as long as the method executes
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	Requires to be initialized before it is used

5. Differentiate between widening and narrowing casting in java.

Widening conversions preserve the source value but can change its representation while 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	1	0	-27 to +27-1
Short	2	0	-215 to +215-1
Int	4	0	-231 to +231-1
Long	8	0L	-2^{63} to $2^{63}-1$
Float	4	00.0f	3.4E-38 to 3.4E+38
Double	8	0.0d	-1.8E+308 to +1.8E+308

7. Explain the importance of using Java packages

- To avoid name conflicts.
- To write a better maintainable code.
- To group related classes.

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.
- CheckboxGroup / radio buttons - Used to display options to the user, where the user selects only one option.
- Choice - Is a pop-up menu that allows the user to select an option.

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

A Container is a component that can have other components or other containers while a component cannot have other components or containers in it.

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

```
import java.util.*;
import java.util.stream.*;
public class Main {
    public static void main(String[] args) {
    //creating my array
    Integer[] myArray = {11,20,34,45,59}; //the five items in the array

    //print the array starting from first element
    System.out.println("Original array:");
    for(int i=0;i<myArray.length;i++) {
         System.out.print(myArray[i] + "  ");
    }
    System.out.println();

    //print the array starting from last element
    System.out.println("Printing the array in reverse order:");
         for(int i=myArray.length-1;i>=0;i--) {
         System.out.print(myArray[i] + "  ");}
    }
}
```

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 a change in the state of an object by performing actions triggered by the user when running the program.
Examples:

When a button is clicked (action), close the application (event).
When a key is pressed on the keyboard (action), close a dialog box (event).

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

- Polymorphism and encapsulation - Polymorphism allows program code to have different functions while encapsulation is the process of keeping classes private so they cannot be modified by external codes.
- Method overloading and method overriding - Method overloading happens at compile-time while method overriding happens at runtime.

- Class and interface - A class doesn't support multiple inheritance while interface supports multiple inheritance.
- Inheritance and polymorphism - Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class(Parent 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.

- Method Overloading -

is the process that we 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 Main {
  public static void main(String[] args) {
    Shapes myShape = new Shapes();  // Create a Shapes object

    myShape.area(12.0,6.2);
    myShape.area(6,2);

  }
}

class Shapes {
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);
  }

}
```

- Method Overriding - Is the process when the subclass or a child class has the same method as declared in the parent class.

```
//Creating a parent class
class Fruit{
  void run(){System.out.println("a fruit is the seed-bearing structure in
flowering plants that is formed from the ovary after flowering");}
}

//Creating a child class
class Orange extends Fruit{
  public static void main(String args[]){

  //creating an instance of child class
  Orange thisClassObj = new Orange();

  //calling the method with child class instance
  thisClassObj.run();
  }
}
```