Mike manthi 21/07002

<u>Task3</u>

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.

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

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

3. Why is initialization of variables required?

- So that they can be used in a program
- To avoid run-time errors.
- It makes a program non-deterministic.

4. Differentiate between static, instance and local variables.

Static variables remain in memory as long as the program executes, instance variables remain in memory as long as the object is in memory while local variables remain in memory as long as method executes.
Static variables can be defined outside a method at the class level, instance variables can be defined outside a method at the class level and local variables can be defined within a method or a code block

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	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
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 group related classes.
- Removes naming collision .
- Provides access protection.

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.

Containers can have other containers and components in it while components cannot have other components in them.

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

```
import java.util.*;
import java.util.stream.*;
public class PrintArrays
{
   public static void main(String[] args) {
   //creating the array with 5 items
   Integer[] myArray = { 1, 2, 3, 4, 5};
   //print the array starting from last element
       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.

Event - Is the changing of the state of an object or behavior by performing actions. These actions can be a button click, cursor movement, keypress through keyboard or page scrolling, etc.

- Example: when the user clicks a button, the program can display a dialog box.
- When the user moves the cursor in a container, the cursor can change its shape.

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

- **Polymorphism and encapsulation**

Polymorphism ensures that the proper method will be executed based on the calling object's type. Encapsulation allows you to control access to your object's state, while making it easier to maintain or change your **implementation at a later date.**

- **Method overloading and method overriding**

Method overloading is used when we want multiple methods providing a similar implementation. However, method overriding is used when we want to add some additional functionality on top of base class implementation.

- **Class and interface**

An object of a class can be created while an object of an interface cannot be created.
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 as well as run-time.

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

Method Overloading:
is the process that we can generate multiple methods of the same identifier 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.

```
//my first class
class Planets {
  public void outEarth() {
    System.out.println("Earth supports life.");
  }
public void outMercury() {
    System.out.println("Mercury is closer to the sun");
  }
}

//second class containing main method
class Main {
  public static void main(String[] args) {
    Planets myPlanet = new Planets();

    myPlanet.outEarth();
    myPlanet.outMercury();
  }
}
```

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

```
//Java Program to demonstrate why we need method overriding
//Creating a parent class
class Person{
  void run(){System.out.println("A person is a human being regarded as an
individual");}
}
```

```
//Creating a child class
class Teacher extends Person{
  public static void main(String args[]){

  //creating an instance of child class
  Teacher obj = new Teacher();
  //calling the method with child class instance
  obj.run();
  }
}
```