Task3

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

- Primitive data types is predefined by the language and is named by a reserved keyword eg byte, char, short, int, long.
  Reference data types are those which contain address of dynamically created objects.

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?

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

4. Differentiate between static, instance and local variables.

LOCAL VARIABLE
Defined within a method or a block of code
It is accessible in the method where it is declared
Does not require any special keyword

INSTANCE VARIABLE
Defined outside a method at the class level
It is accessible throughout the class
Does not require any special keyword

STATIC VARIABLE
Defined outside a method at the class level
It is accessible throughout the class
Requires the static keyword to be specified

5. Differentiate between widening and narrowing casting in java.

Widening conversation changes a value to a data type that can allow for any possible value of the original data. Widening conversations preserve the source value but can change its representation.
A narrowing conversation 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.
- To avoid name conflicts.

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 refers to the ability of a class to provide different implementations of a method depending on the type of object that is passed to the method.
Encapsulation is the process by which data (variables) and the code that acts upon them (methods) are integrated as single unit.

Method overloading and method overriding

Method overloading is a concept of java in which we can create multiple methods of the same name in the same class and all methods work in different ways.

Method overriding occurs when a subclass has the same method as the parent class. It occurs when a subclass provides a particular implementation of a method declared by one of its parent classes.

Class and interface

A java class file is a file containing java byte code that can be executed on the java virtual machine. A java class file is usually produced by a java compiler from java programming language source files containing java classes.
An interface is a reference type in java. It is similar to class. It is a collection of abstract methods.

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.

1. Method overloading - is the process that can create multiple methods of the same name in the same class and all methods work in different ways.
Example

```
Class shapes {
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.peintln("Triangle area= "+0.5*b*h);
}
public void area(int l, int b)
{
System.out.println("Rectangle area="+1*b);
}
}
Class Main{
public static void
main(string []args){
Shapes my shape = new
Shapes (); // create a shape object
myshape.area()
myshape.area(5);
myshape.area(6.0,1.2);
myshape.area(6.2);
  }
}
```

Method overriding-is the process when the subclasses or a child class has the same method as declared in the parent class.
Example

```
Class vehicle {
```

```
//defining a method
void run
{System.out.priintln("vehicle is moving");}
}
//creating child class
class car 2 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[]){
car 2 obj = new
Car2()://creating object
obj.run()://calling method
}
}
```