GROUP NAME: BE FEARLESS

NAME:	**REG NO.**
Cleophas Mugo	**SCM211-0341/2021**
Jacob Kut	**SCM211-0361/2021**
Samuel Ben	**SCM211-0356/2021**
Elvis Ayillo	**SCM211-0355/2021**
Benjamin Rutto	**SCM211-0318/2021**
Cleophas Oichoe	**SCM211-0351/2021**
Andrew Mutungan	**SCM211-1333/2021**
Toel Jim	**SCM211-1258/2021**

OOP II

ASSIGNMENT 1.

Abstract Classes

An abstract class is a class that is declared abstract, it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class.

It is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

An abstract class must be declared with an abstract keyword.

It can have constructors and static methods also.

It can have final methods which will force the subclass not to change the body of the method.

Its syntax is:

//Syntax:

<Access_Modifier> abstract class <Class_Name> {

//Data_Members;

//Statements;

//Methods;

}

e.g.

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Inner Classes

An inner class in Java is defined as a class that is declared inside another class. Inner classes are often used to create helper classes, such as views or adapters that are used by the outer class. Inner classes can also be used to create nested data structures, such as a linked list.

Inner classes are a security mechanism in Java. We know a class cannot be associated with the access modifier private, but if we have the class as a member of other class, then the inner class can be made private. And this is also used to access the private members of a class.

Its syntax is:

OuterClass outerObject = new OuterClass();

OuterClass.InnerClass innerObject = outerObject.new InnerClass();

e.g.

```
class Outer_Demo {
  int num;

  // inner class
  private class Inner_Demo {
    public void print() {
      System.out.println("This is an inner class");
    }
```

```
  }

  // Accessing he inner class from the method within
  void display_Inner() {
    Inner_Demo inner = new Inner_Demo();
    inner.print();
  }
}

public class My_class {

  public static void main(String args[]) {
    // Instantiating the outer class
    Outer_Demo outer = new Outer_Demo();

    // Accessing the display_Inner() method.
    outer.display_Inner();
  }
}
```

Anonymous Classes

In Java, a class can contain another class known as nested class. It's possible to create a nested class without giving any name.

A nested class that doesn't have any name is known as an anonymous class.An anonymous class must be defined inside another class. Hence, it is also known as an anonymous inner class. Its syntax is:

```
class outerClass {
   // defining anonymous class
   object1 = new Type(parameterList) {
       // body of the anonymous class
```

```
    };
}
```

e.g.

```
class Polygon {
  public void display() {
    System.out.println("Inside the Polygon class");
  }
}

class AnonymousDemo {
  public void createClass() {

    // creation of anonymous class extending class Polygon
    Polygon p1 = new Polygon() {
      public void display() {
        System.out.println("Inside an anonymous class.");
      }
    };
    p1.display();
  }
}

class Main {
  public static void main(String[] args) {
     AnonymousDemo an = new AnonymousDemo();
     an.createClass();
  }
}
```