**SCM211-0363/2020 HARMONY NAMAI**

**SCM211-0282/2021 PAUL CALEB NGUMBAU**

**SCM211-0236/2018 JOSEPH MUKULA JACOB**

**SCM211-1343/2021 MWENDWA JOHN MUTISO**

**Abstract Methods and Classes**

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

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.

An *abstract method* is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

abstract void moveTo(double deltaX, double deltaY);

If a class includes abstract methods, then the class itself *must* be declared abstract, as in:

public abstract class GraphicObject {

// declare fields

// declare nonabstract methods

abstract void draw();

}

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract

**Java Inner Class**

Java inner class is defined inside the body of another class. Java inner class can be declared private, public, protected, or with default access whereas an outer class can have only public or default access. Java Nested classes are divided into two types.

**static nested class**

If the nested class is static, then it’s called a static nested class. Static nested classes can access only static members of the outer class. A static nested class is the same as any other top-level class and is nested for only packaging convenience. A static class object can be created with the following statement.

Any non-static nested class is known as inner class in java. Java inner class is associated with the object of the class and they can access all the variables and methods of the outer class. Since inner classes are associated with the instance, we can’t have any static variables in them. The object of java inner class are part of the outer class object and to create an instance of the inner class, we first need to create an instance of outer class. Java inner class can be instantiated like this

class OuterClass {

int x = 10;

class InnerClass {

int y = 5;

}

}

public class Main {

public static void main(String[] args) {

OuterClass myOuter = new OuterClass();

OuterClass.InnerClass myInner = myOuter.new InnerClass();

System.out.println(myInner.y + myOuter.x);

}

}

**Java Anonymous Class**

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:

**Note:** Anonymous classes are defined inside an expression. So, the semicolon is used at the end of anonymous classes to indicate the end of the expression.

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();

}

}

In anonymous classes, objects are created whenever they are required. That is, objects are created to perform some specific tasks. For example