*// 150364  
// Ndungi Tiffany Waithera  
  
import* java.sql.Connection;  
*import* java.sql.DriverManager;  
*import* java.sql.SQLException;  
  
*public class* DatabaseConnector {  
 *// Required variables:  
 private static* DatabaseConnector instance = *null*;  
 *private static final* String USERNAME = "root";  
 *private static final* String PASSWORD = "";  
 *private static final* String DB\_URL = "jdbc:mysql://localhost:3306/db\_books";  
 *private static final* String DB\_DRIVER = "com.mysql.cj.jdbc.Driver";  
 *private* Connection conn;  
  
 *private* DatabaseConnector(){  
 *// private constructor - prevents direct instantiation* }  
  
 *// Function to create only one instance  
 public static* DatabaseConnector getInstance(){  
 *if*(instance == *null*){  
 instance = *new* DatabaseConnector();  
 }  
 *return* instance;  
 }  
  
 *// DatabaseConnector methods:  
  
 // Method for establishing a connection  
 public void* establishConnection() {  
 *try* {  
 Class.forName(DB\_DRIVER);  
 conn = DriverManager.getConnection(DB\_URL, USERNAME, PASSWORD);  
 System.out.println("Connected successfully!");  
 } *catch* (SQLException | NullPointerException | ClassNotFoundException e) {  
 System.out.println("Connection failed: " + e.getMessage());  
 }  
 }  
  
 *// Method for closing a connection  
 public void* closeConnection(){  
 *try*{  
 conn.close();  
 System.out.println("Connection closed successfully!");  
  
 }*catch*(NullPointerException | SQLException e){  
 System.out.println("Failed to close connection: "+e.getMessage());  
 }  
 }  
}

*// Main class to test database connection  
public class* Main {  
 *public static void* main(String[] args) *throws* InterruptedException {  
 DatabaseConnector db = DatabaseConnector.getInstance();*//creating only one object of the DatabaseConnector class* db.establishConnection();*//establish the database connection* Thread.sleep(3000);*//wait 3 seconds* db.closeConnection();*//close the database connection* }  
}

*// Factory Method  
// 150364  
  
public abstract class* Shape {  
 *// Abstract methods to be implemented by subclasses  
 public abstract* String getName();  
 *public abstract int* getSidesCount();  
  
 *// Function to create objects of type Shape by using the subclasses  
  
 public static* Shape generateShape(String shape){  
 *switch*(shape){  
 *case* "triangle" ->{  
 *return new* Triangle();  
 }  
 *case* "circle"->{  
 *return new* Circle();  
 }  
 *case* "quadrilateral"->{  
 *return new* Quadrilateral();  
 }  
 *case* "hexagon"->{  
 *return new* Hexagon();  
 }  
 *default* ->{  
 *throw new* IllegalArgumentException("Invalid option. Choose between triangle, circle, hexagon," +  
 " and quadrilateral.");  
 }  
 }  
 }  
}

*public class* Triangle *extends* Shape {  
 *@Override  
 public* String getName() {  
 *return* "Triangle";  
 }  
  
 *@Override  
 public int* getSidesCount() {  
 *return* 3;  
 }  
}

*public class* Circle *extends* Shape {  
 *@Override  
 public* String getName() {  
 *return* "Circle";  
 }  
  
 *@Override  
 public int* getSidesCount() {  
 *return* 0;  
 }  
}

*public class* Hexagon *extends* Shape {  
 *@Override  
 public* String getName() {  
 *return* "Hexagon";  
 }  
  
 *@Override  
 public int* getSidesCount() {  
 *return* 6;  
 }  
}

*public class* Quadrilateral *extends* Shape {  
 *@Override  
 public* String getName() {  
 *return* "Quadrilateral";  
 }  
  
 *@Override  
 public int* getSidesCount() {  
 *return* 4;  
 }  
}

*public class* Main {  
 *public static void* main(String[] args) {  
 *// Generating the shape:* Shape hexagon = Shape.generateShape("hexagon");  
 Shape quadrilateral = Shape.generateShape("quadrilateral");  
 Shape circle = Shape.generateShape("circle");  
 Shape triangle = Shape.generateShape("triangle");  
  
 *// Printing the name and sides:* System.out.println("Shapes and no. of sides:\n" +  
 hexagon.getName()+": "+hexagon.getSidesCount()+"\n" +  
 quadrilateral.getName()+": "+quadrilateral.getSidesCount()+"\n" +  
 circle.getName()+": "+circle.getSidesCount()+"\n" +  
 triangle.getName()+": "+triangle.getSidesCount()+"\n");  
 }  
}

*// OBSERVER DESIGN  
// Subject - Traffic Light Signal  
public interface* TrafficLightSubject {  
 *void* addObserver(TrafficLightObserver observer);  
 *void* removeObserver(TrafficLightObserver observer);  
 *void* notifyObservers();  
}

*// Creating an enum to store the traffic light constants for easy switching  
// This isn't necessary, String variables can be used instead:  
public enum* TrafficLightColours {  
  
 RED,  
 AMBER,  
 GREEN  
}

*// Concrete subject - concrete implementation of the subject interface   
import* java.util.ArrayList;  
*import* java.util.List;  
  
*public class* TrafficLightSignal *implements* TrafficLightSubject{  
 *private* TrafficLightColours light;  
 *private final* List<TrafficLightObserver> observers = *new* ArrayList<>();  
  
 *public void* addObserver(TrafficLightObserver observer){  
 observers.add(observer);  
 }  
  
 *public void* removeObserver(TrafficLightObserver observer){  
 observers.remove(observer);  
 }  
  
 *public void* setLight(TrafficLightColours light){  
 *this*.light = light;  
 notifyObservers();  
 }  
  
 *@Override  
 public void* notifyObservers(){  
 *for*(TrafficLightObserver observer: observers){  
 observer.update(light);  
 }  
 }  
}

*// Concrete observer(s)  
public class* TrafficLight *implements* TrafficLightObserver{  
 TrafficLightColours light;  
 *@Override  
 public void* update(TrafficLightColours light) {  
 *this*.light = light;  
 displayLight();  
 }  
  
 *private void* displayLight(){  
 *switch*(*this*.light.ordinal()){  
 *case* 0->{  
 System.out.println("RED");  
 }  
 *case* 1 -> {  
 System.out.println("AMBER");  
 }  
 *case* 2 ->{  
 System.out.println("GREEN");  
 }  
 }  
 }  
}

*// Main class to implement Observer Design  
  
import* java.util.concurrent.TimeUnit; *// This is just a library for a delay function  
  
public class* Main {  
 *public static void* main(String[] args) *throws* InterruptedException {  
 TrafficLightSignal trafficlightsignal = *new* TrafficLightSignal();  
 trafficlightsignal.addObserver(*new* TrafficLight());  
  
 *// Start the Traffic Light signal:  
 //It runs 3 times before stopping* System.out.println("Traffic Light is up. Will cycle 3 times.");  
 *int* trafficLightCounter = 1;  
 *do* {  
 trafficlightsignal.setLight(TrafficLightColours.RED);*// print 'RED'* TimeUnit.SECONDS.sleep(1);*// wait 1 second* trafficlightsignal.setLight(TrafficLightColours.AMBER);*//print 'AMBER'* TimeUnit.SECONDS.sleep(1);*// wait 1 second* trafficlightsignal.setLight(TrafficLightColours.GREEN);*//print 'GREEN'* TimeUnit.SECONDS.sleep(1);*// wait 1 second* trafficlightsignal.setLight(TrafficLightColours.AMBER);*//print 'AMBER'* TimeUnit.SECONDS.sleep(1);*// wait 1 second* trafficLightCounter++;  
  
 } *while* (trafficLightCounter < 3);  
 System.out.println("Traffic light stopped...");  
  
 }  
}

*//DECORATORS  
// Area interface with the abstract function to calculate area  
public interface* Area {  
 *double* calculateArea(*float* length, *float* width);  
}

*// Concrete implementation of the interface Area  
public class* RectangleArea *implements* Area{  
  
 *@Override  
 public double* calculateArea(*float* length, *float* width) {  
 *return* length\*width;  
 }  
}

*// A decorator superclass - to be inherited by subclasses to convert units and change area format  
public abstract class* AreaDecorator *implements* Area{  
 *protected* Area decoratedArea;  
  
 *public* AreaDecorator(Area decoratedArea){  
 *this*.decoratedArea=decoratedArea;  
  
 }  
  
 *@Override  
 public double* calculateArea(*float* length, *float* width) {  
 *return* decoratedArea.calculateArea(length, width);  
 }  
}

*// A subclass decorator to convert units  
public class* UnitConverter *extends* AreaDecorator {  
 *private final* String unit;*//holds the specific unit chosen by the user  
 public* UnitConverter(Area decoratedArea, String unit) {  
 *super*(decoratedArea);*// call the constructor of AreaDecorator  
 this*.unit = unit;  
 }  
  
 *@Override  
 public double* calculateArea(*float* length, *float* width) {  
 *double* area = *super*.calculateArea(length, width);  
 *// Converting centimetres to inches and vice versa.  
 /\* Conversion details:  
 If 1 inch = 2.54 centimetres, then:  
 1 inch\*\*2 = (2.45)\*\*2, approximately 6.452, and  
 1 cm\*\*2 = 1/6.452, approximately 0.155  
 \*/  
 switch*(*this*.unit){  
 *case* "in"->{  
 System.out.println("From cm^2 to in^2: ");  
 *return* area \* 0.155;  
 }*case* "cm"->{  
 System.out.println("From in^2 to cm^2: ");  
 *return* area \* 6.452;  
 }*default*->{  
 *return* area; *// Return the area without conversion if none is among the options* }  
 }  
 }  
}

*// Formatting the area in decimal or scientific  
public class* AreaFormat *extends* AreaDecorator{  
 *private* String format;  
  
 *public* AreaFormat(Area decoratedArea, String format) {  
 *super*(decoratedArea);  
 *this*.format = format;  
 }  
  
 *public double* calculateArea(*float* length, *float* width){  
 *double* area = *super*.calculateArea(length, width);  
 System.out.print("Area:");  
 *if* (format.equals("scientific")) {  
 */\*System.out.format() is a function that prints out the value assigned to it in a specified  
 format\*/  
 /\*  
 %e-> scientific  
 %f-> float - For a specific no of decimal places: %.2f-> two decimal places  
 %n-> new line  
 \*/* System.out.format("%e%n", area); *// e = \* 10^n;* } *else if* (format.equals("decimal")) {  
 System.out.format("%.2f%n", area); *//return in two decimal places* } *else* {  
 System.out.println(area);  
 }  
 *return* area;  
 }  
}

*import* java.util.Scanner;  
*public class* Main {  
 *public static void* main(String[] args) {  
 *// Defining the variables  
 float* length, width;  
 String format, unit;  
  
 *// Creating scanner object and rectangleArea object of type Area:* Scanner input = *new* Scanner(System.in);  
  
 *// Creating an object of type Area using class RectangleArea that has concrete implementation of it* Area rectangleArea = *new* RectangleArea();  
  
 *// Prompting the user with details:* System.out.println("Welcome to Rectangle Area calculator!");  
 System.out.println("Please enter the length: ");  
 length = input.nextFloat();  
 System.out.println("Please enter the width: ");  
 width = input.nextFloat();  
 System.out.println("Great! Now enter the units to convert to(in or cm): ");  
 unit = input.next();  
 System.out.println("Finally, what format do u want it to be in? (scientific or decimal): ");  
 format = input.next();  
  
 *// Creating decorator objects:* rectangleArea = *new* UnitConverter(rectangleArea, unit);  
 rectangleArea = *new* AreaFormat(rectangleArea, format);  
  
 *// Function that computes the area* rectangleArea.calculateArea(length, width);  
  
 }  
}

*// Adapter Design  
// Client interface - the TemperatureConverter  
public interface* TemperatureConverter {  
  
 *// Abstract method to convert the temperatures  
 double* convertFahrenheitToCelsius(*double* fahrenheit);  
}

*// Kelvin interface  
public interface* KelvinInterface {  
 *double* convertFahrenheitToKelvin(*double* fahrenheit);

*double* convertKelvinToCelsius(*double* kelvin);  
}

*// Concrete implementation of the KelvinInterface  
public class* KelvinConverter *implements* KelvinInterface{  
 *public double* convertFahrenheitToKelvin(*double* fahrenheit) {  
 *// K = ((F - 32) \*5/9)+273.15  
  
 return* ((fahrenheit - 32) \* 5 / 9)+273.15;  
 }  
  
 *public double* convertKelvinToCelsius(*double* kelvin) {  
 *return* kelvin - 273.15;  
 }  
}

*// Adapter class  
class* TemperatureConverterAdapter *implements* TemperatureConverter {  
 *private* KelvinInterface kelvinConverter;  
  
 *public* TemperatureConverterAdapter(KelvinInterface kelvinTemperatureConverter) {  
 *this*.kelvinConverter = kelvinTemperatureConverter;  
 }  
  
 *@Override  
 public double* convertFahrenheitToCelsius(*double* fahrenheit) {  
 *double* kelvin = kelvinConverter.convertFahrenheitToKelvin(fahrenheit);  
 *return* kelvinConverter.convertKelvinToCelsius(kelvin);  
 }  
}

*import* java.util.Scanner;  
  
*// Main class to execute client code  
public class* Main {  
 *public static void* main(String[] args) {  
 Scanner input = *new* Scanner(System.in);  
  
 System.out.println("Enter a temperature in Fahrenheit: ");  
 *double* fahrenheit = input.nextDouble();  
  
 KelvinInterface kelvinConverter = *new* KelvinConverter();  
 TemperatureConverter temperatureConverter = *new* TemperatureConverterAdapter(kelvinConverter);  
  
 *double* celsius = temperatureConverter.convertFahrenheitToCelsius(fahrenheit);  
 System.out.println(fahrenheit + "°F is equal to " + celsius + "°C.");  
 }  
}