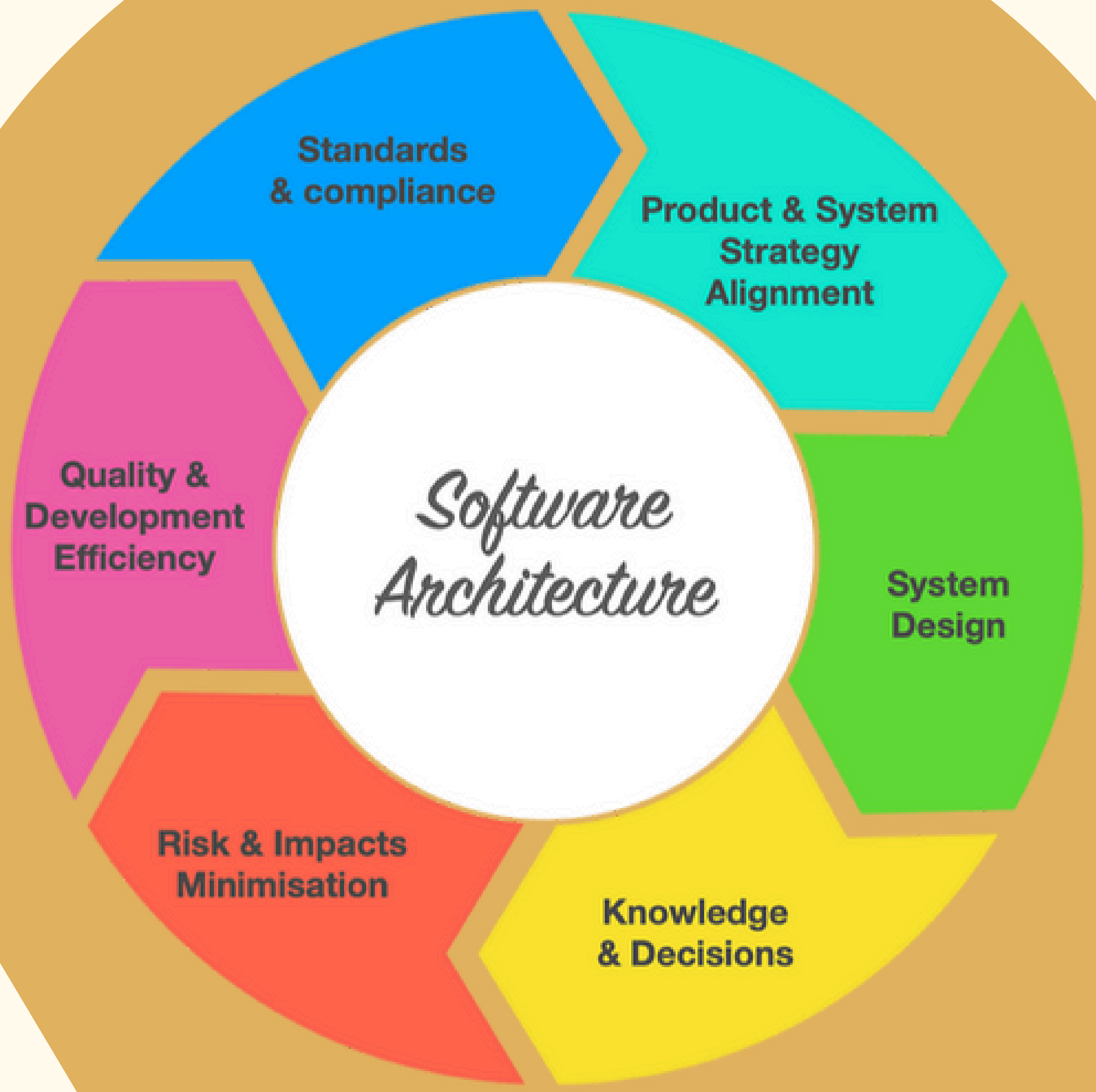
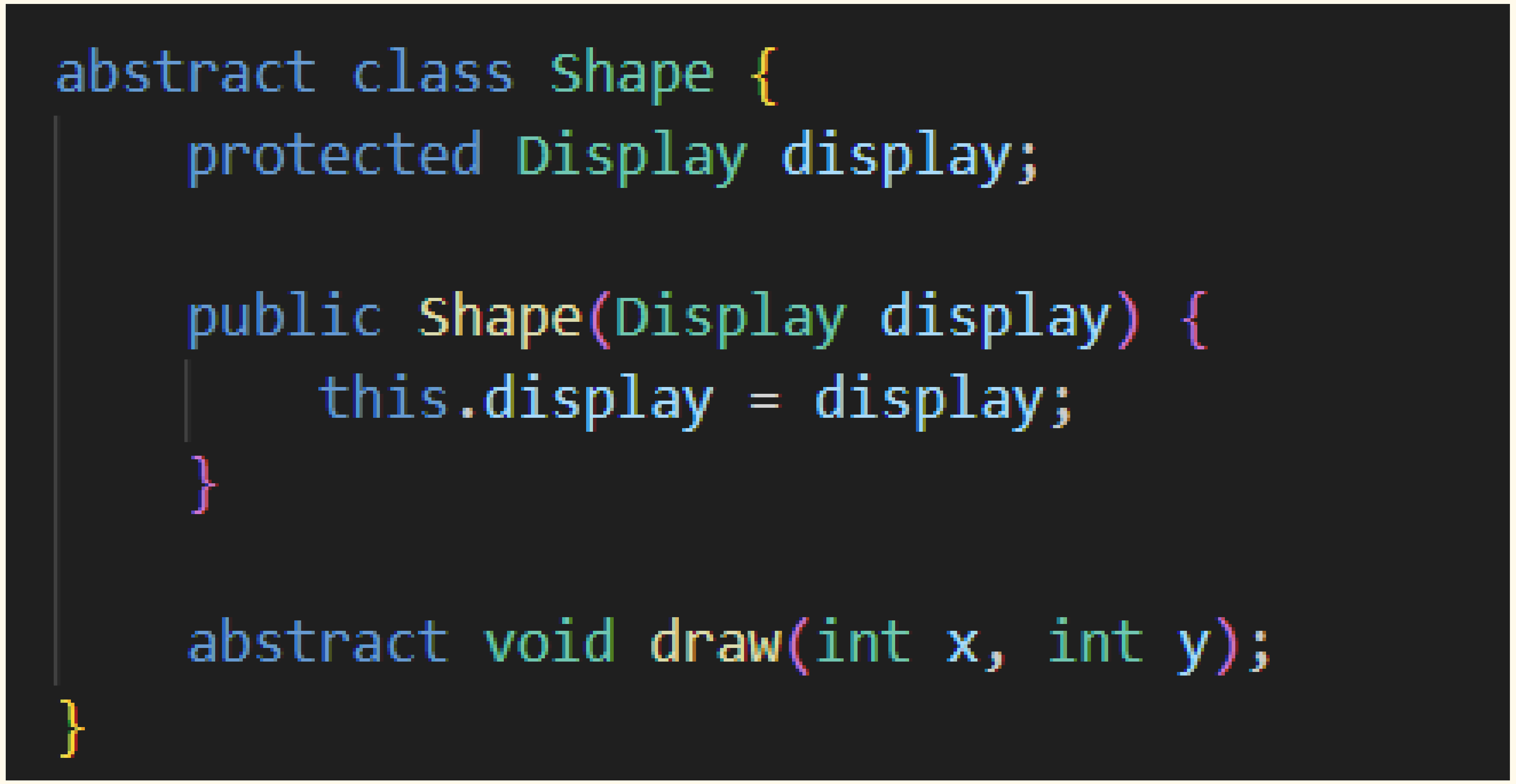
WEEK 9


Standards & compliance
Product & System Strategy Alignment
System Design
Knowledge & Decisions
Risk & Impacts Minimisation
Quality & Development Efficiency
Software Architecture

https://www.canva.com/design/DAGQDqrFOM0/tYcShPJMrfAxQmkaKsYQwg/edit

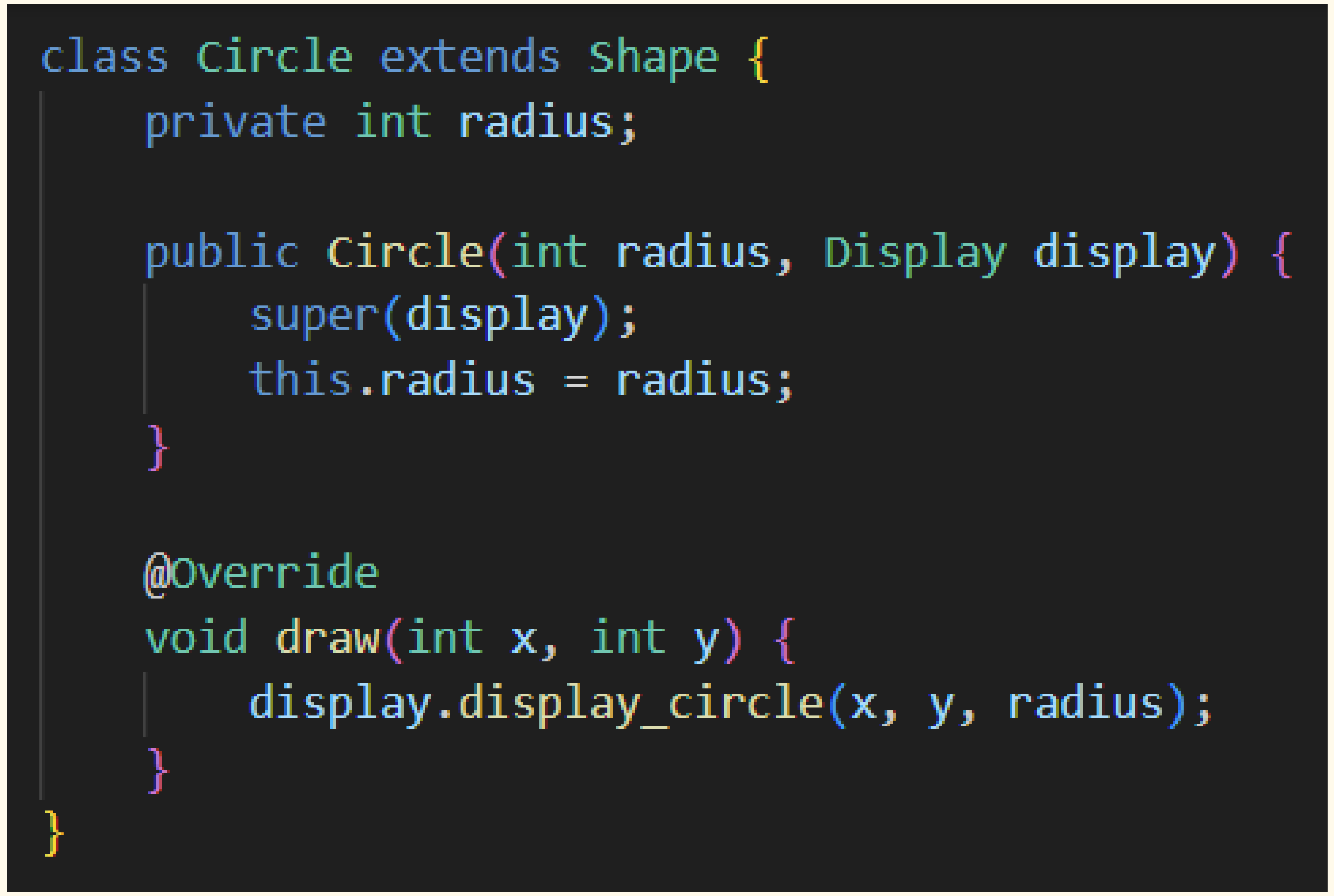
TASK 1

Abstraction



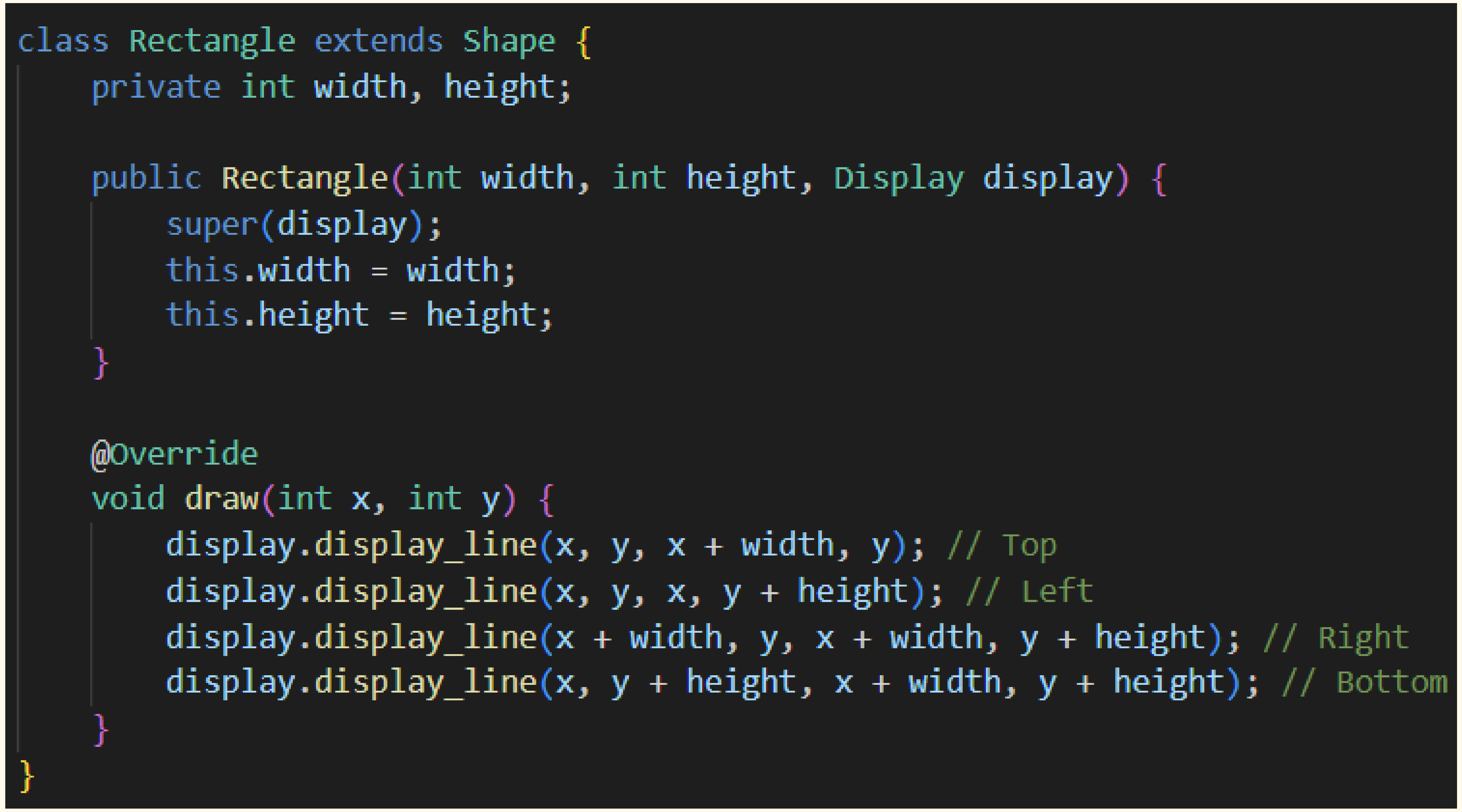
```java
abstract class Shape {
    protected Display display;

    public Shape(Display display) {
        this.display = display;
    }

    abstract void draw(int x, int y);
}
```

Variation



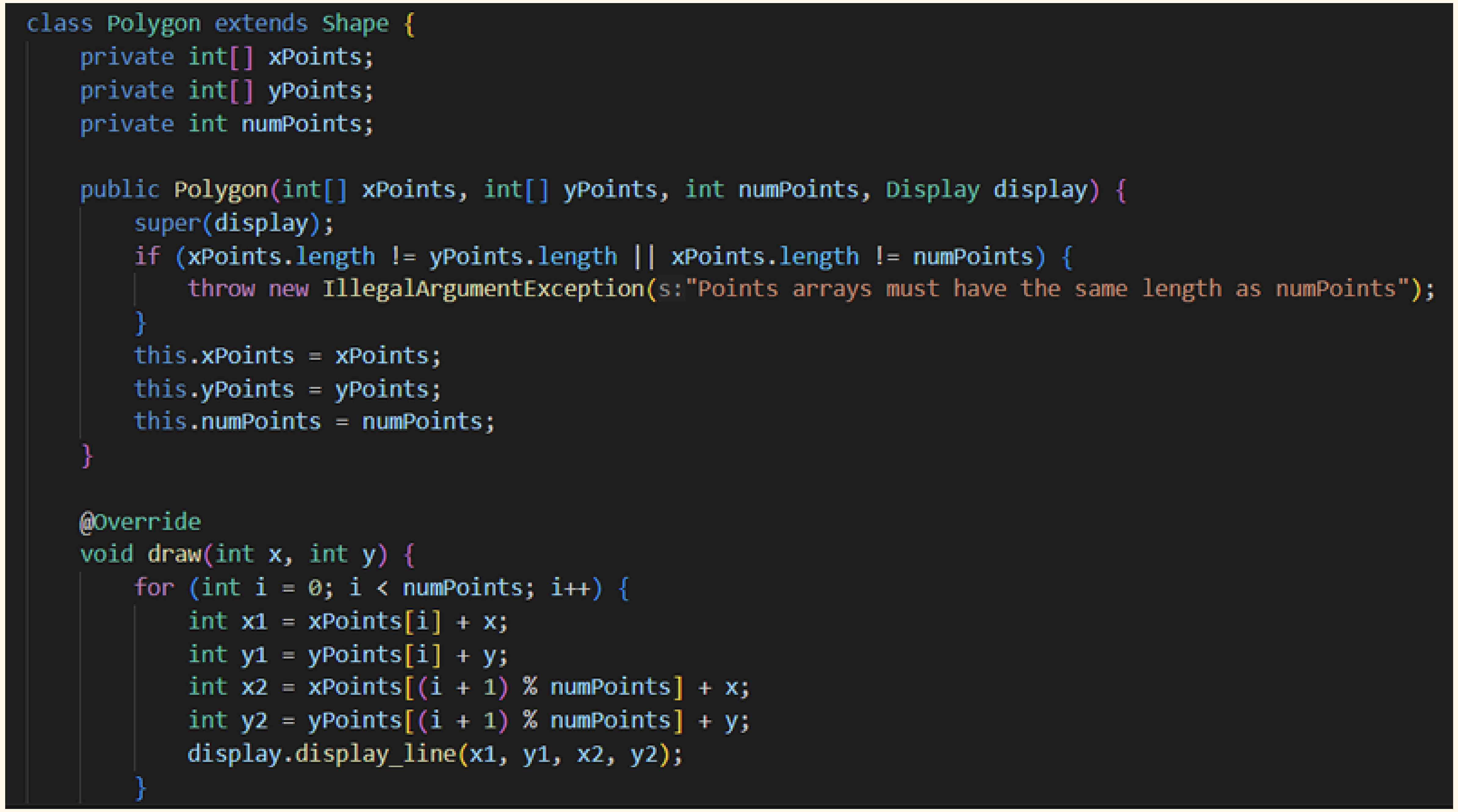
```
class Circle extends Shape {
    private int radius;

    public Circle(int radius, Display display) {
        super(display);
        this.radius = radius;
    }

    @Override
    void draw(int x, int y) {
        display.display_circle(x, y, radius);
    }
}
```

Variation



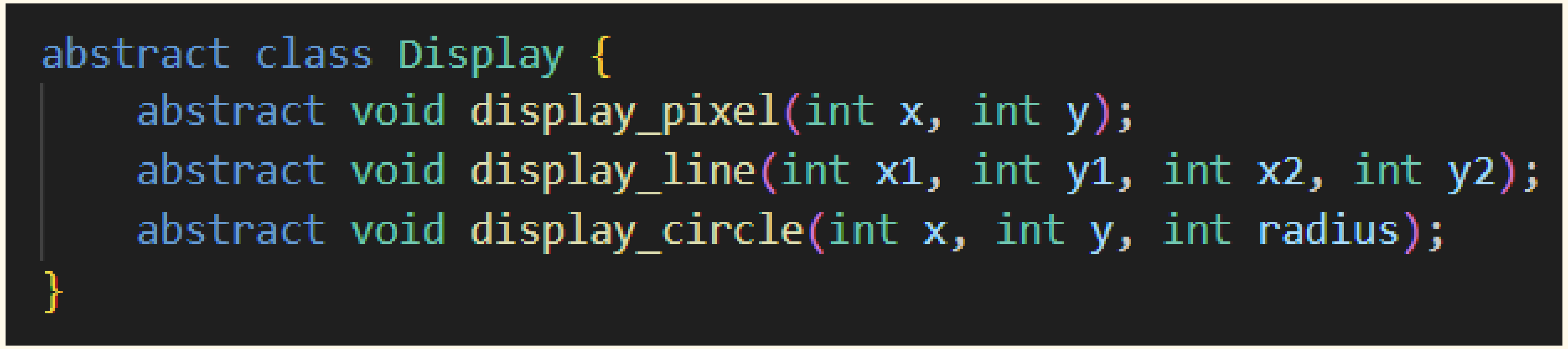
```java
class Rectangle extends Shape {
    private int width, height;

    public Rectangle(int width, int height, Display display) {
        super(display);
        this.width = width;
        this.height = height;
    }

    @Override
    void draw(int x, int y) {
        display.display_line(x, y, x + width, y); // Top
        display.display_line(x, y, x, y + height); // Left
        display.display_line(x + width, y, x + width, y + height); // Right
        display.display_line(x, y + height, x + width, y + height); // Bottom
    }
}
```

Variation



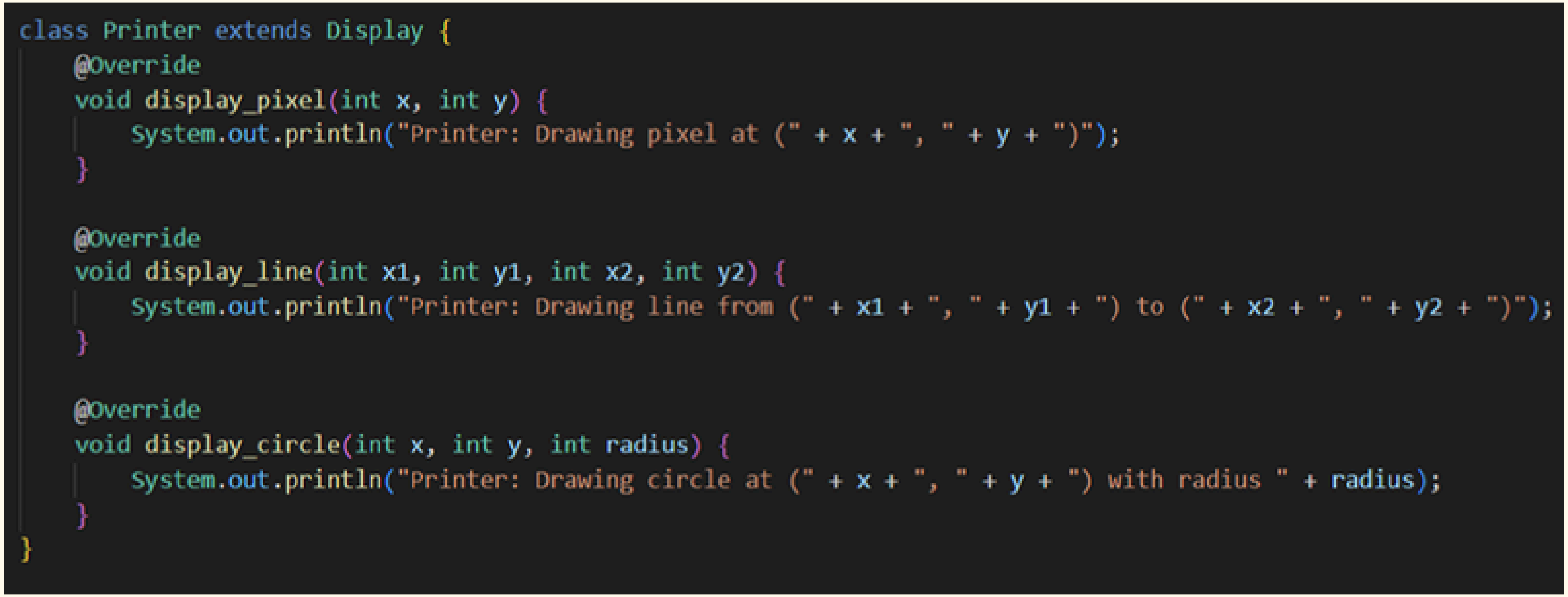
```java
class Polygon extends Shape {
    private int[] xPoints;
    private int[] yPoints;
    private int numPoints;

    public Polygon(int[] xPoints, int[] yPoints, int numPoints, Display display) {
        super(display);
        if (xPoints.length != yPoints.length || xPoints.length != numPoints) {
            throw new IllegalArgumentException(s:"Points arrays must have the same length as numPoints");
        }
        this.xPoints = xPoints;
        this.yPoints = yPoints;
        this.numPoints = numPoints;
    }

    @Override
    void draw(int x, int y) {
        for (int i = 0; i < numPoints; i++) {
            int x1 = xPoints[i] + x;
            int y1 = yPoints[i] + y;
            int x2 = xPoints[(i + 1) % numPoints] + x;
            int y2 = yPoints[(i + 1) % numPoints] + y;
            display.display_line(x1, y1, x2, y2);
        }
```

Implementor



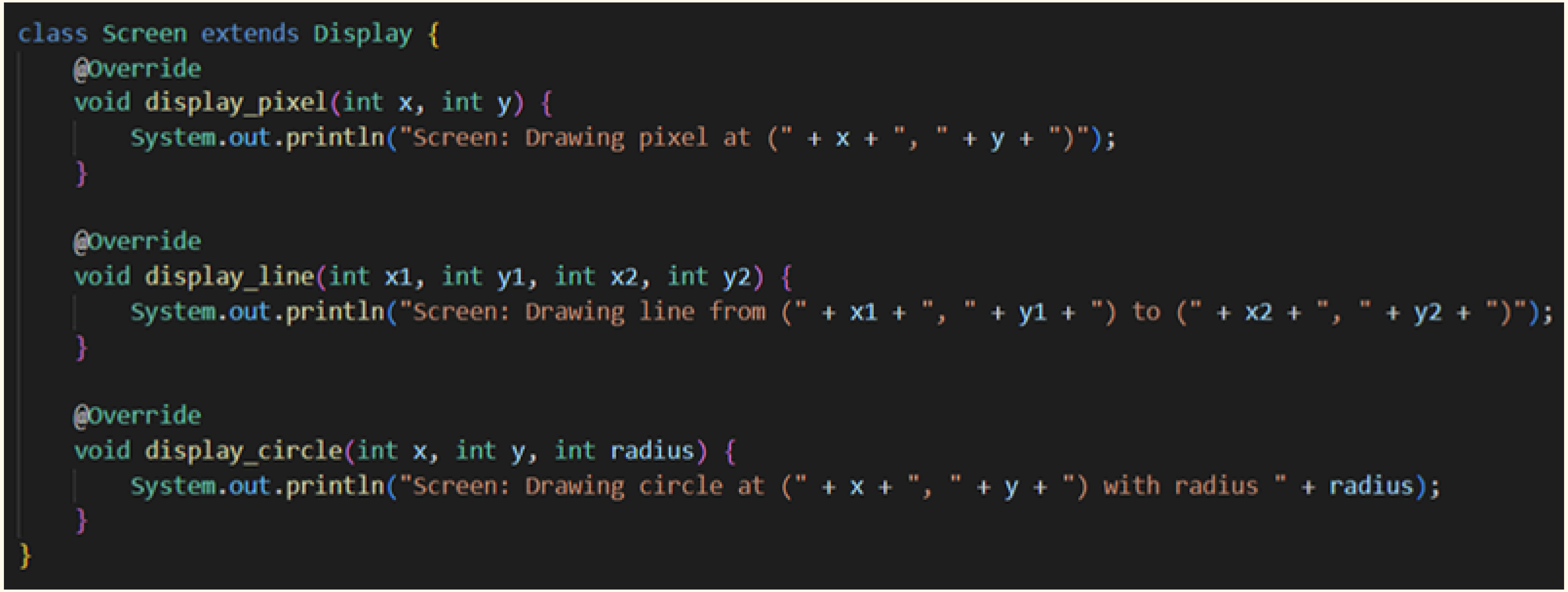
```
abstract class Display {
    abstract void display_pixel(int x, int y);
    abstract void display_line(int x1, int y1, int x2, int y2);
    abstract void display_circle(int x, int y, int radius);
}
```

ConcreteImplementor



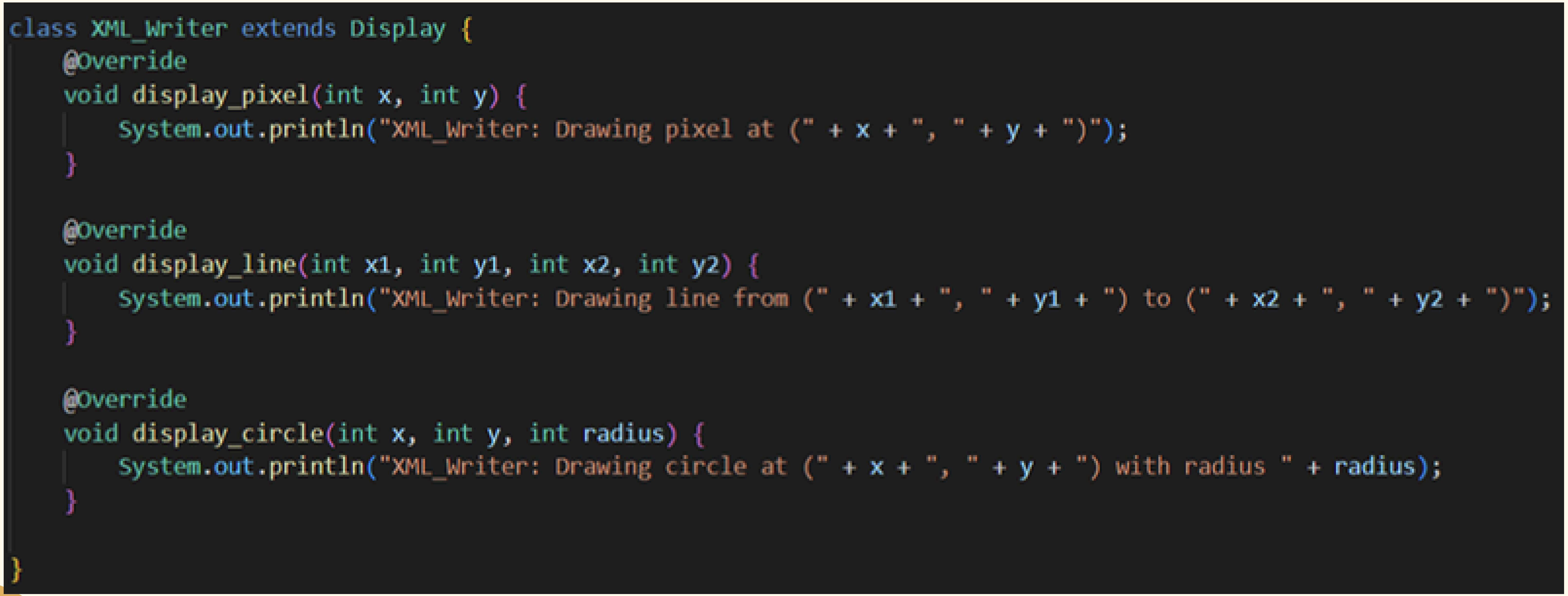
```java
class Printer extends Display {
    @Override
    void display_pixel(int x, int y) {
        System.out.println("Printer: Drawing pixel at (" + x + ", " + y + ")");
    }

    @Override
    void display_line(int x1, int y1, int x2, int y2) {
        System.out.println("Printer: Drawing line from (" + x1 + ", " + y1 + ") to (" + x2 + ", " + y2 + ")");
    }

    @Override
    void display_circle(int x, int y, int radius) {
        System.out.println("Printer: Drawing circle at (" + x + ", " + y + ") with radius " + radius);
    }
}
```

ConcreteImplementor

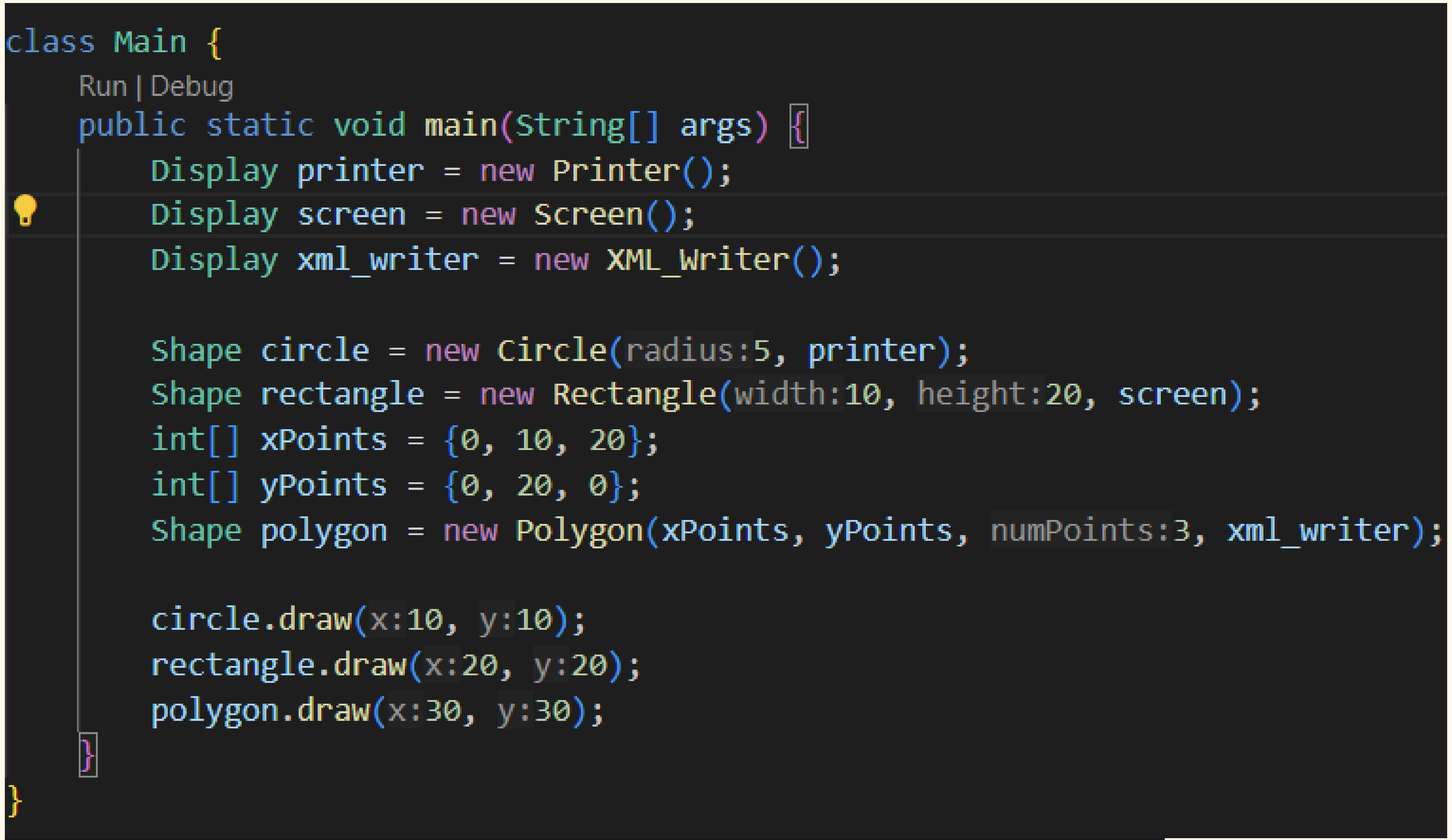


```java
class Screen extends Display {
    @Override
    void display_pixel(int x, int y) {
        System.out.println("Screen: Drawing pixel at (" + x + ", " + y + ")");
    }

    @Override
    void display_line(int x1, int y1, int x2, int y2) {
        System.out.println("Screen: Drawing line from (" + x1 + ", " + y1 + ") to (" + x2 + ", " + y2 + ")");
    }

    @Override
    void display_circle(int x, int y, int radius) {
        System.out.println("Screen: Drawing circle at (" + x + ", " + y + ") with radius " + radius);
    }
}
```

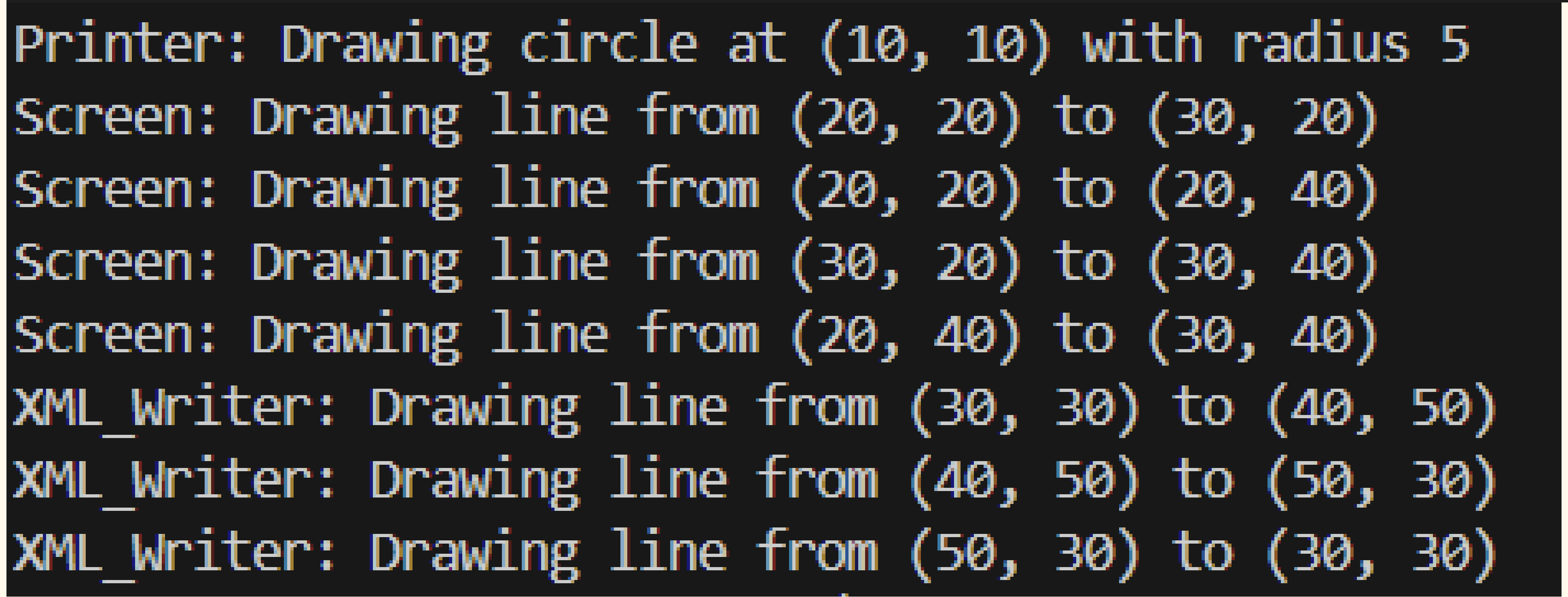
ConcreteImplementor



```java
class XML_Writer extends Display {
    @Override
    void display_pixel(int x, int y) {
        System.out.println("XML_Writer: Drawing pixel at (" + x + ", " + y + ")");
    }

    @Override
    void display_line(int x1, int y1, int x2, int y2) {
        System.out.println("XML_Writer: Drawing line from (" + x1 + ", " + y1 + ") to (" + x2 + ", " + y2 + ")");
    }

    @Override
    void display_circle(int x, int y, int radius) {
        System.out.println("XML_Writer: Drawing circle at (" + x + ", " + y + ") with radius " + radius);
    }

}
```

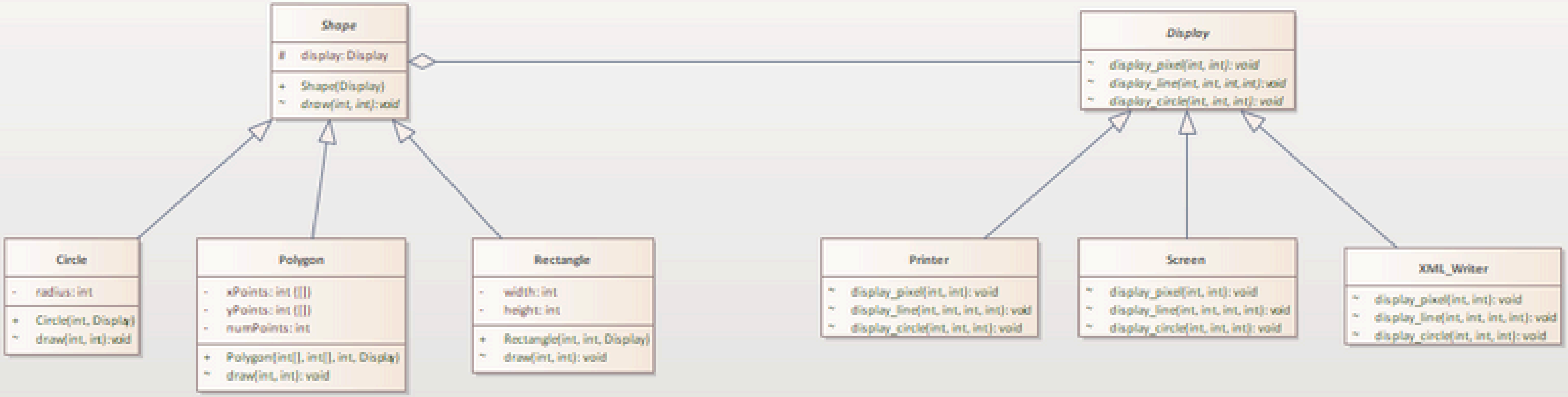


```
class Main {
    Run | Debug
    public static void main(String[] args) {
        Display printer = new Printer();
        Display screen = new Screen();
        Display xml_writer = new XML_Writer();

        Shape circle = new Circle(radius:5, printer);
        Shape rectangle = new Rectangle(width:10, height:20, screen);
        int[] xPoints = {0, 10, 20};
        int[] yPoints = {0, 20, 0};
        Shape polygon = new Polygon(xPoints, yPoints, numPoints:3, xml_writer);

        circle.draw(x:10, y:10);
        rectangle.draw(x:20, y:20);
        polygon.draw(x:30, y:30);
    }
}
```



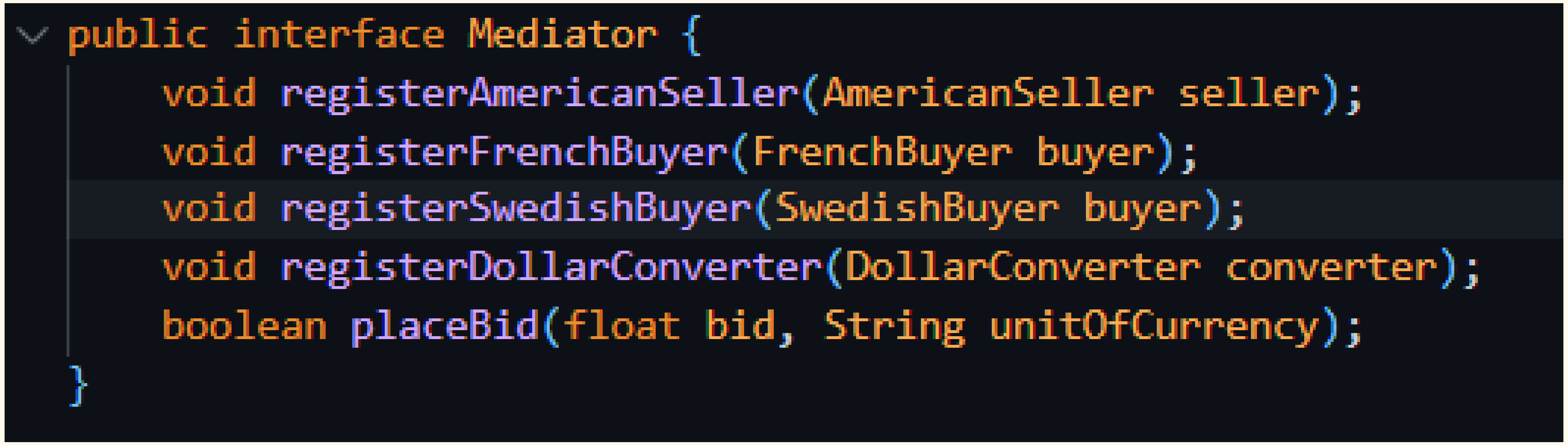
```
Printer: Drawing circle at (10, 10) with radius 5
Screen: Drawing line from (20, 20) to (30, 20)
Screen: Drawing line from (20, 20) to (20, 40)
Screen: Drawing line from (30, 20) to (30, 40)
Screen: Drawing line from (20, 40) to (30, 40)
XML_Writer: Drawing line from (30, 30) to (40, 50)
XML_Writer: Drawing line from (40, 50) to (50, 30)
XML_Writer: Drawing line from (50, 30) to (30, 30)
```

Class diagram


Shape
# display: Display
+ Shape(Display)
~ draw(int, int): void
Display
~ display_pixel(int, int): void
~ display_line(int, int, int, int): void
~ display_circle(int, int, int): void
Circle
- radius: int
+ Circle(int, Display)
~ draw(int, int): void
Polygon
- xPoints: int ([])
- yPoints: int ([])
- numPoints: int
+ Polygon(int[], int[], int, Display)
~ draw(int, int): void
Rectangle
- width: int
- height: int
+ Rectangle(int, int, Display)
~ draw(int, int): void
Printer
~ display_pixel(int, int): void
~ display_line(int, int, int, int): void
~ display_circle(int, int, int): void
Screen
~ display_pixel(int, int): void
~ display_line(int, int, int, int): void
~ display_circle(int, int, int): void
XML_Writer
~ display_pixel(int, int): void
~ display_line(int, int, int, int): void
~ display_circle(int, int, int): void

TASK 2

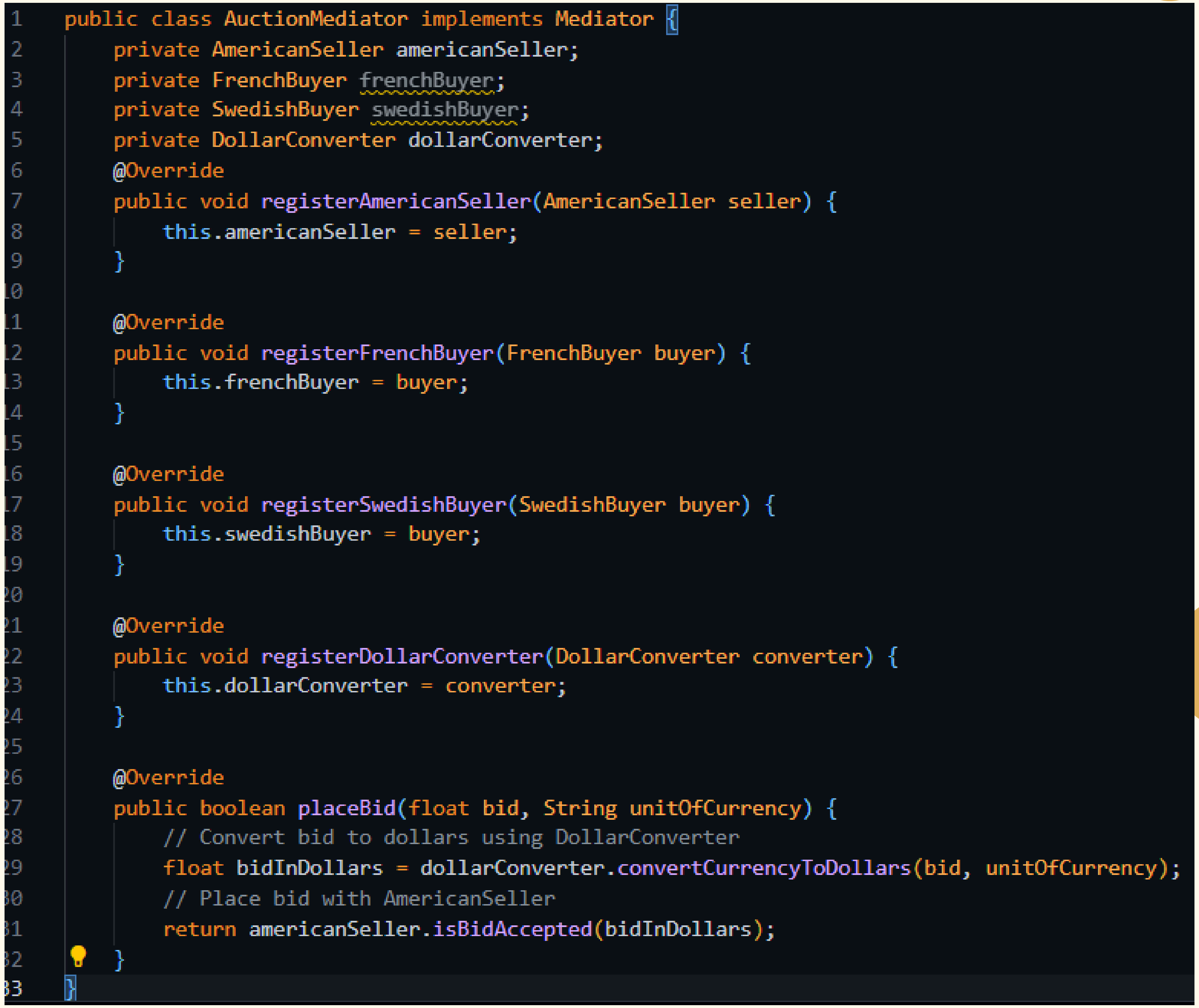
Mediator Interface



```java
public interface Mediator {
    void registerAmericanSeller(AmericanSeller seller);
    void registerFrenchBuyer(FrenchBuyer buyer);
    void registerSwedishBuyer(SwedishBuyer buyer);
    void registerDollarConverter(DollarConverter converter);
    boolean placeBid(float bid, String unitOfCurrency);
}
```

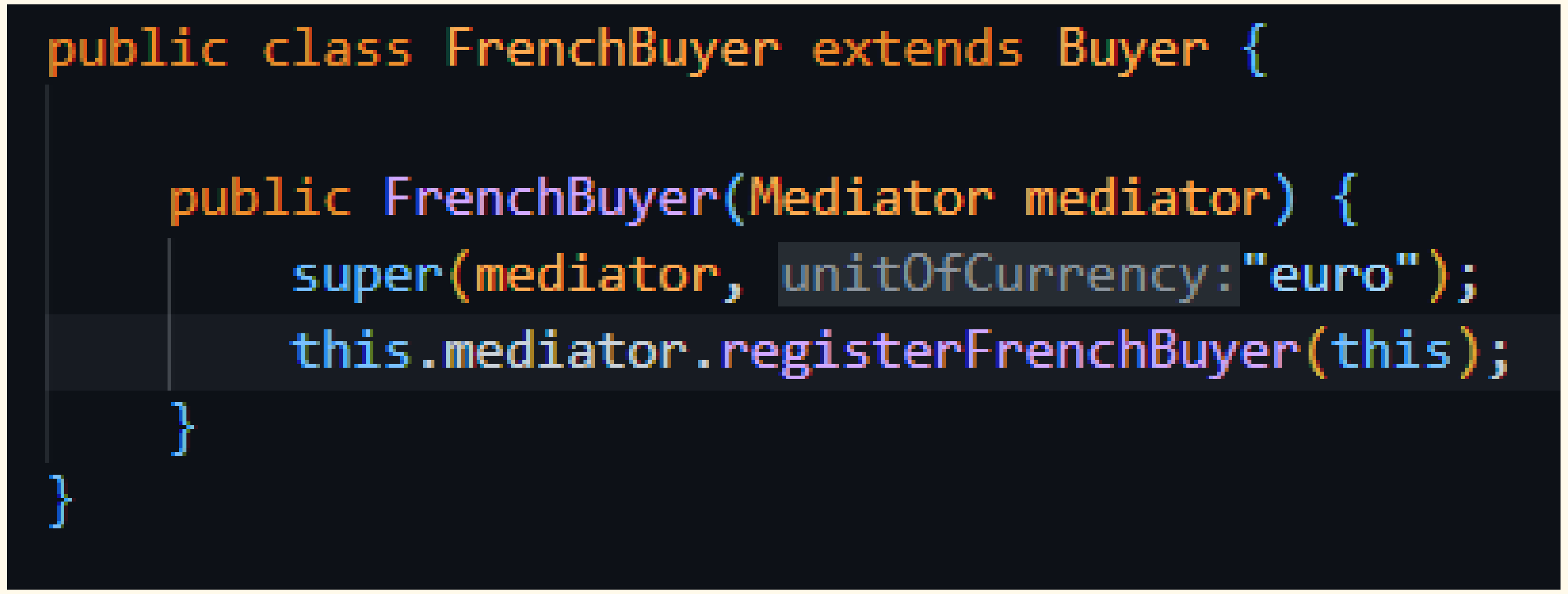
ConcreteMediator

implemented the logic for bid conversion and communication between the seller and buyers.

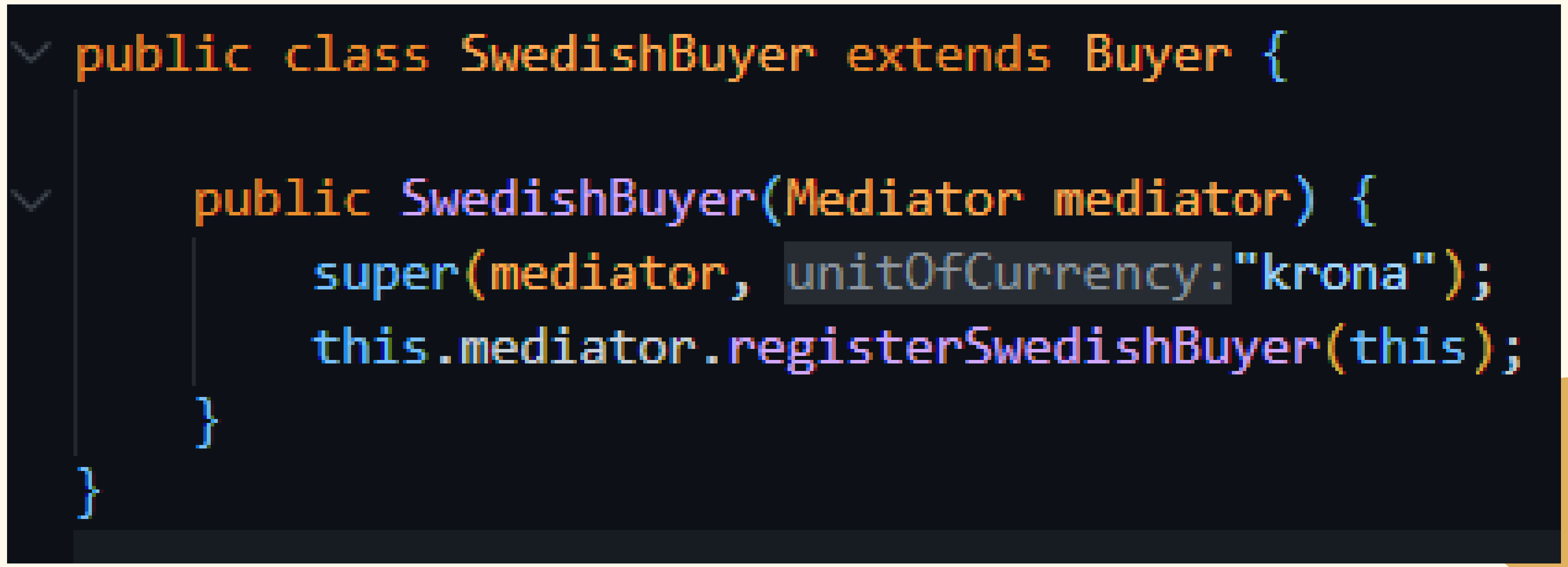


```java
public class AuctionMediator implements Mediator {
    private AmericanSeller americanSeller;
    private FrenchBuyer frenchBuyer;
    private SwedishBuyer swedishBuyer;
    private DollarConverter dollarConverter;
    @Override
    public void registerAmericanSeller(AmericanSeller seller) {
        this.americanSeller = seller;
    }

    @Override
    public void registerFrenchBuyer(FrenchBuyer buyer) {
        this.frenchBuyer = buyer;
    }

    @Override
    public void registerSwedishBuyer(SwedishBuyer buyer) {
        this.swedishBuyer = buyer;
    }

    @Override
    public void registerDollarConverter(DollarConverter converter) {
        this.dollarConverter = converter;
    }

    @Override
    public boolean placeBid(float bid, String unitOfCurrency) {
        // Convert bid to dollars using DollarConverter
        float bidInDollars = dollarConverter.convertCurrencyToDollars(bid, unitOfCurrency);
        // Place bid with AmericanSeller
        return americanSeller.isBidAccepted(bidInDollars);
    }
}
```

component classes

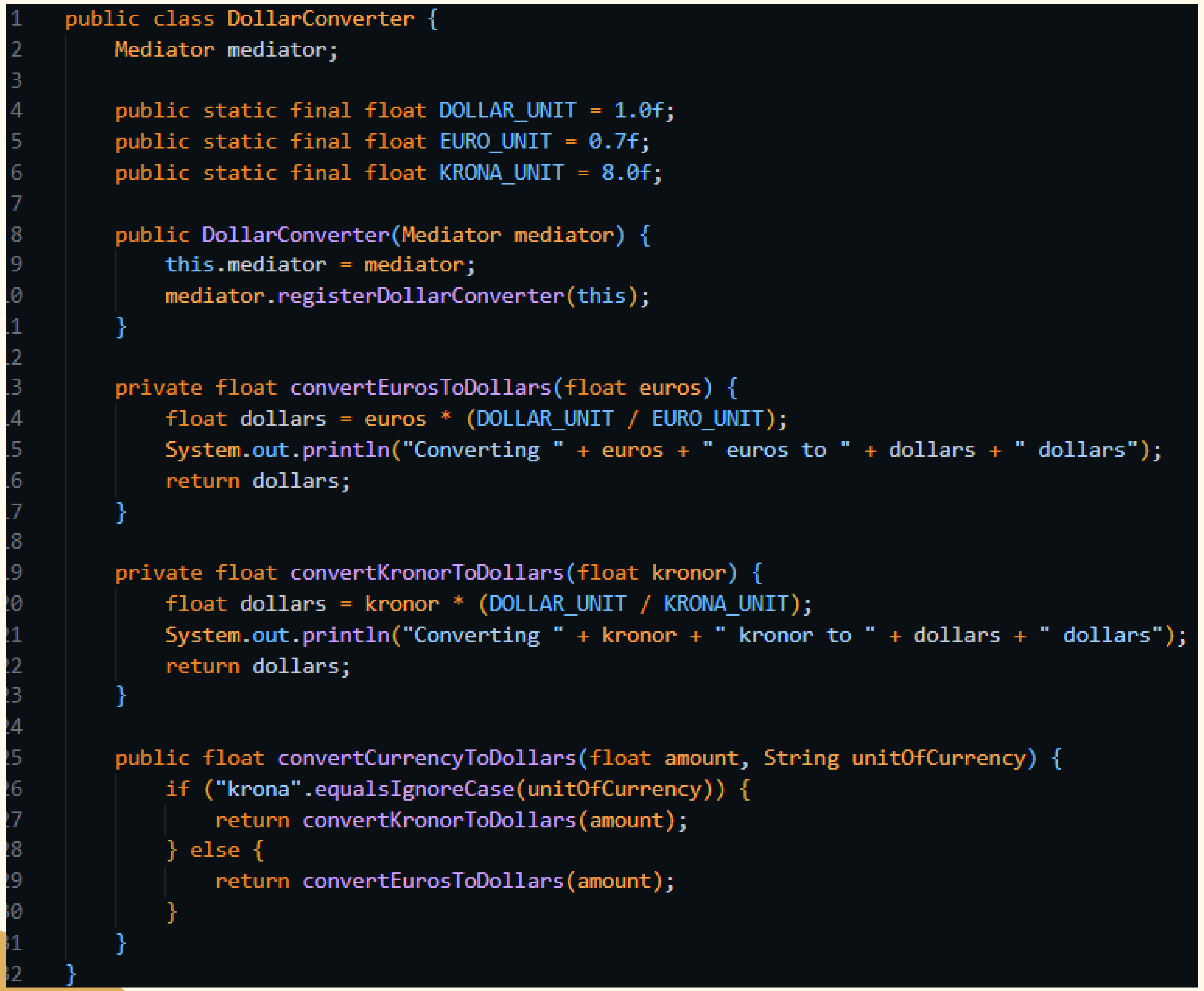


```
public class FrenchBuyer extends Buyer {

    public FrenchBuyer(Mediator mediator) {
        super(mediator, unitOfCurrency:"euro");
        this.mediator.registerFrenchBuyer(this);
    }
}
```

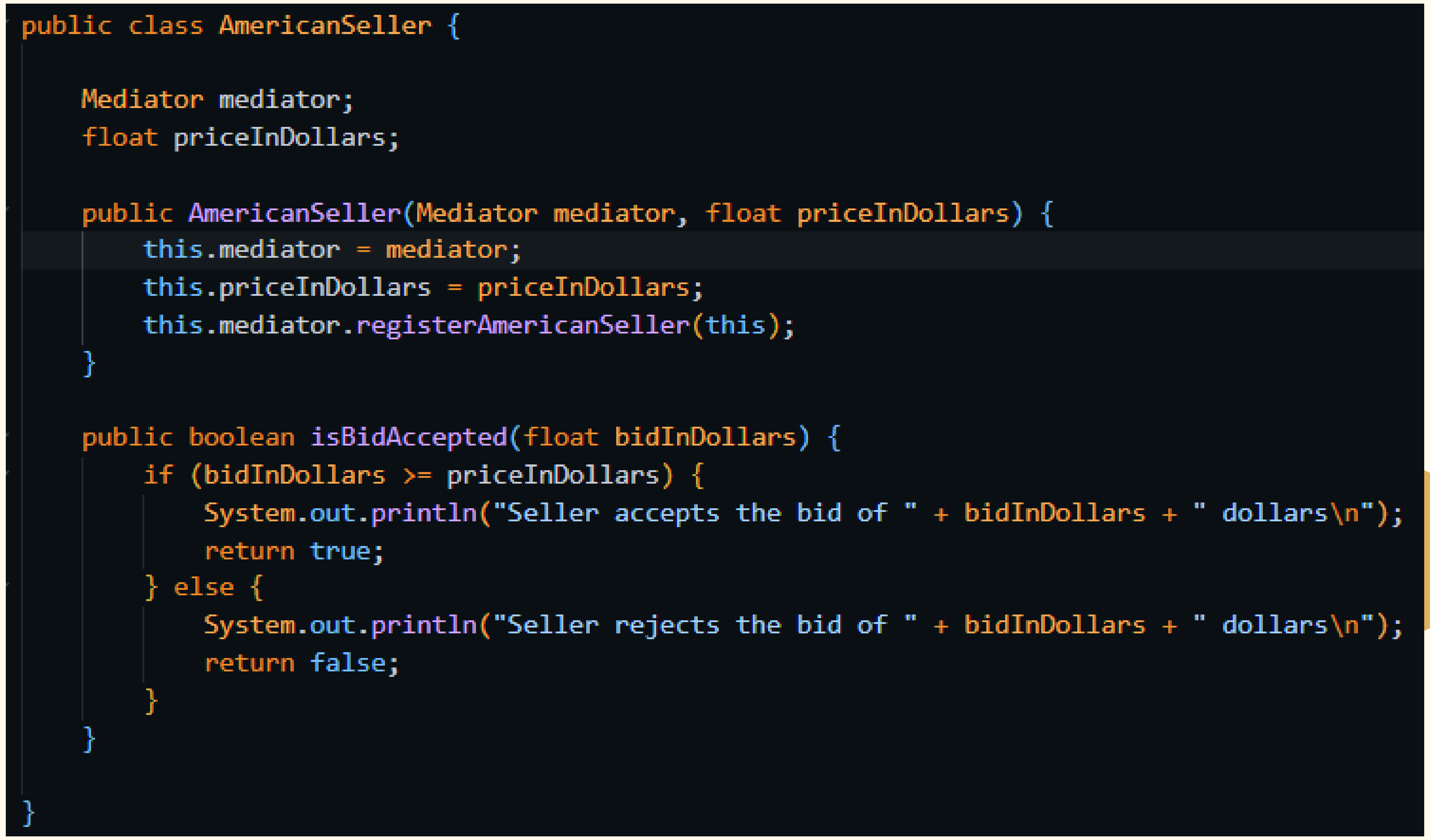


```
public class SwedishBuyer extends Buyer {

    public SwedishBuyer(Mediator mediator) {
        super(mediator, unitOfCurrency:"krona");
        this.mediator.registerSwedishBuyer(this);
    }
}
```

component classes

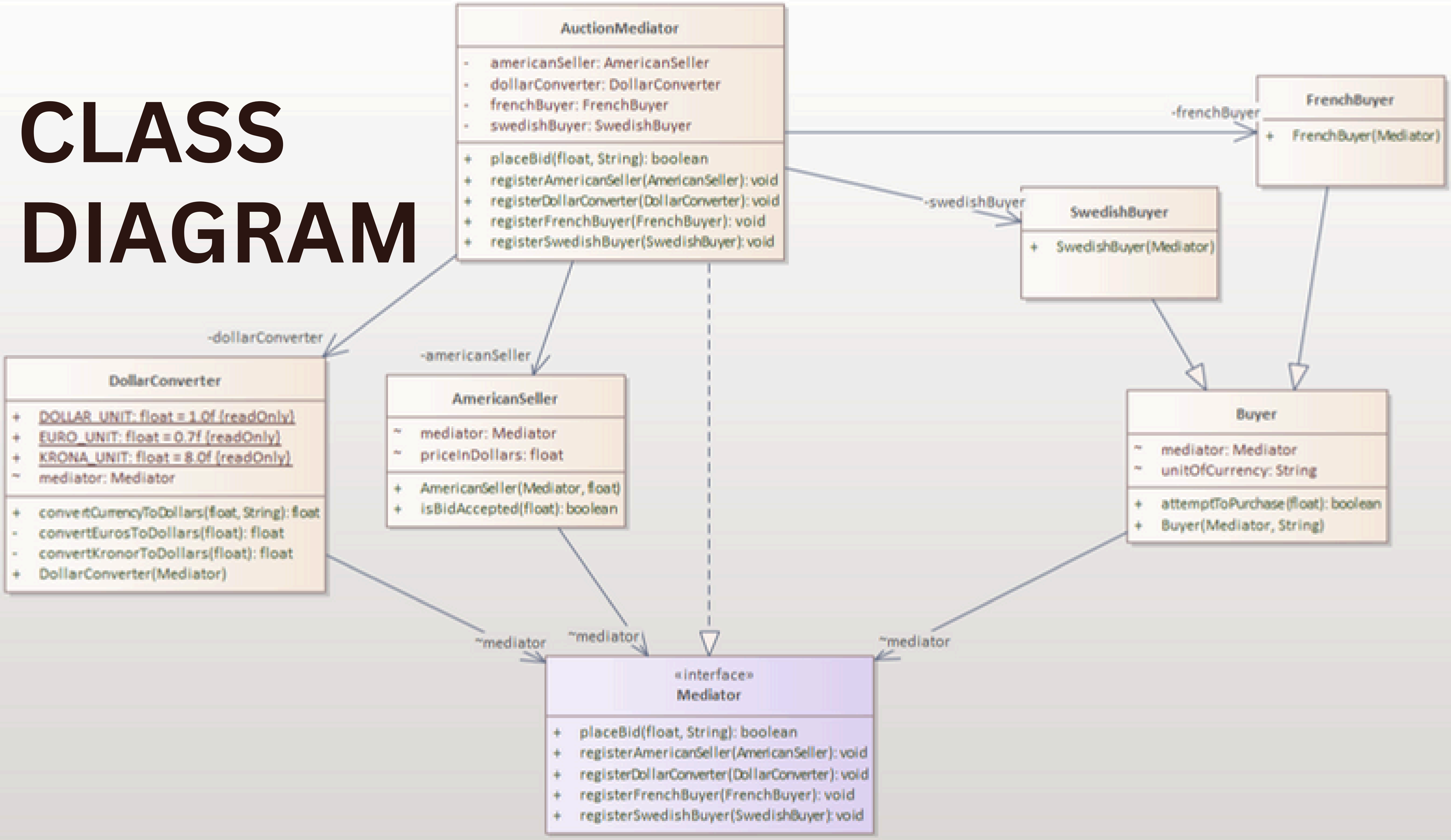


```java
public class DollarConverter {
    Mediator mediator;

    public static final float DOLLAR_UNIT = 1.0f;
    public static final float EURO_UNIT = 0.7f;
    public static final float KRONA_UNIT = 8.0f;

    public DollarConverter(Mediator mediator) {
        this.mediator = mediator;
        mediator.registerDollarConverter(this);
    }

    private float convertEurosToDollars(float euros) {
        float dollars = euros * (DOLLAR_UNIT / EURO_UNIT);
        System.out.println("Converting " + euros + " euros to " + dollars + " dollars");
        return dollars;
    }

    private float convertKronorToDollars(float kronor) {
        float dollars = kronor * (DOLLAR_UNIT / KRONA_UNIT);
        System.out.println("Converting " + kronor + " kronor to " + dollars + " dollars");
        return dollars;
    }

    public float convertCurrencyToDollars(float amount, String unitOfCurrency) {
        if ("krona".equalsIgnoreCase(unitOfCurrency)) {
            return convertKronorToDollars(amount);
        } else {
            return convertEurosToDollars(amount);
        }
    }
}
```

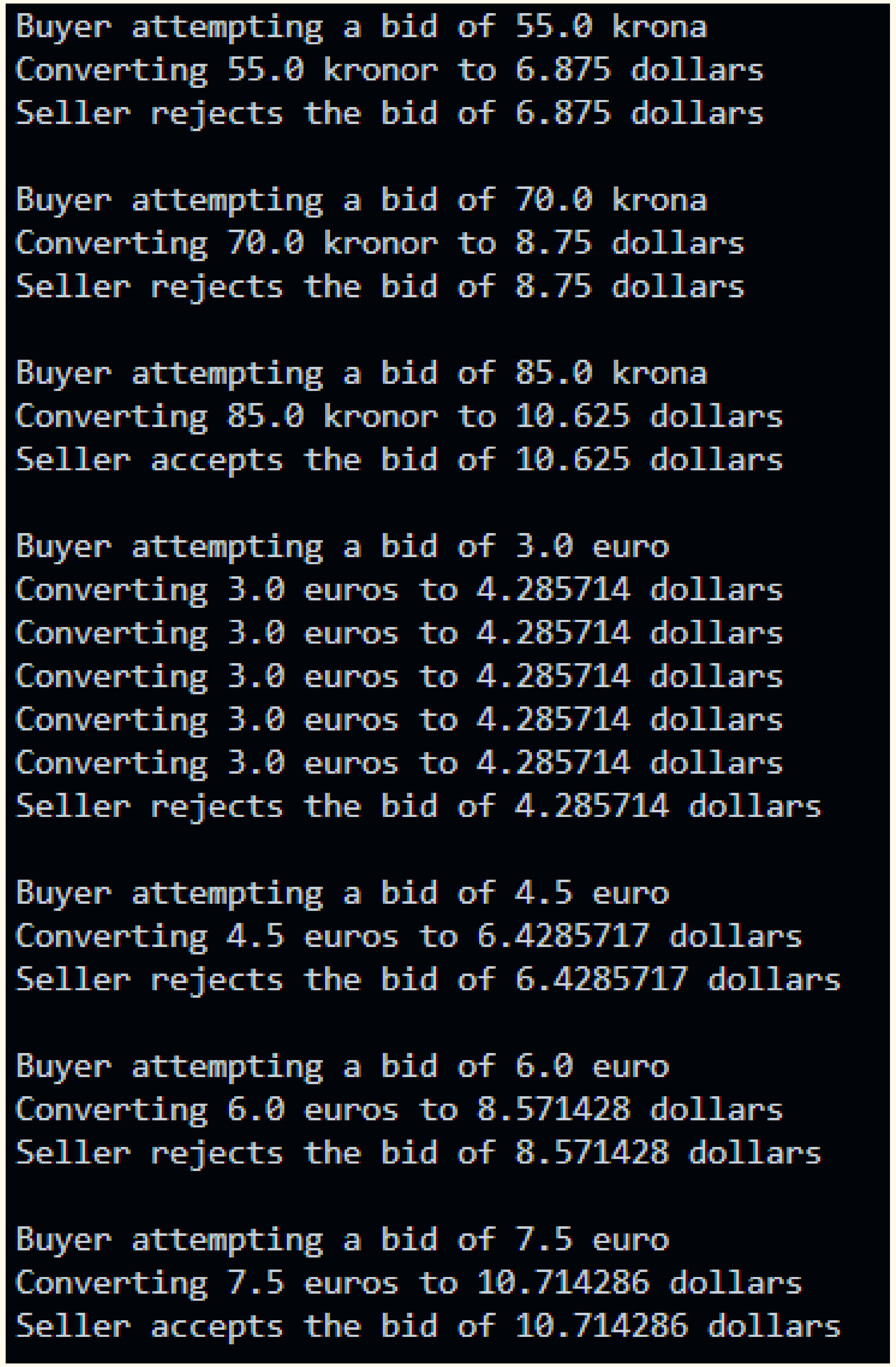


```java
public class AmericanSeller {

    Mediator mediator;
    float priceInDollars;

    public AmericanSeller(Mediator mediator, float priceInDollars) {
        this.mediator = mediator;
        this.priceInDollars = priceInDollars;
        this.mediator.registerAmericanSeller(this);
    }

    public boolean isBidAccepted(float bidInDollars) {
        if (bidInDollars >= priceInDollars) {
            System.out.println("Seller accepts the bid of " + bidInDollars + " dollars\n");
            return true;
        } else {
            System.out.println("Seller rejects the bid of " + bidInDollars + " dollars\n");
            return false;
        }
    }

}
```

CLASS DIAGRAM


AuctionMediator
- americanSeller: AmericanSeller
- dollarConverter: DollarConverter
- frenchBuyer: FrenchBuyer
- swedishBuyer: SwedishBuyer
+ placeBid(float, String): boolean
+ registerAmericanSeller(AmericanSeller): void
+ registerDollarConverter(DollarConverter): void
+ registerFrenchBuyer(FrenchBuyer): void
+ registerSwedishBuyer(SwedishBuyer): void
-frenchBuyer
FrenchBuyer
+ FrenchBuyer(Mediator)
-swedishBuyer
SwedishBuyer
+ SwedishBuyer(Mediator)
-dollarConverter
-americanSeller
DollarConverter
+ DOLLAR_UNIT: float = 1.0f {readOnly}
+ EURO_UNIT: float = 0.7f {readOnly}
+ KRONA_UNIT: float = 8.0f {readOnly}
~ mediator: Mediator
+ convertCurrencyToDollars(float, String): float
- convertEurosToDollars(float): float
- convertKronorToDollars(float): float
+ DollarConverter(Mediator)
AmericanSeller
~ mediator: Mediator
~ priceInDollars: float
+ AmericanSeller(Mediator, float)
+ isBidAccepted(float): boolean
Buyer
~ mediator: Mediator
~ unitOfCurrency: String
+ attemptToPurchase(float): boolean
+ Buyer(Mediator, String)
~mediator
~mediator
~mediator
«interface»
Mediator
+ placeBid(float, String): boolean
+ registerAmericanSeller(AmericanSeller): void
+ registerDollarConverter(DollarConverter): void
+ registerFrenchBuyer(FrenchBuyer): void
+ registerSwedishBuyer(SwedishBuyer): void

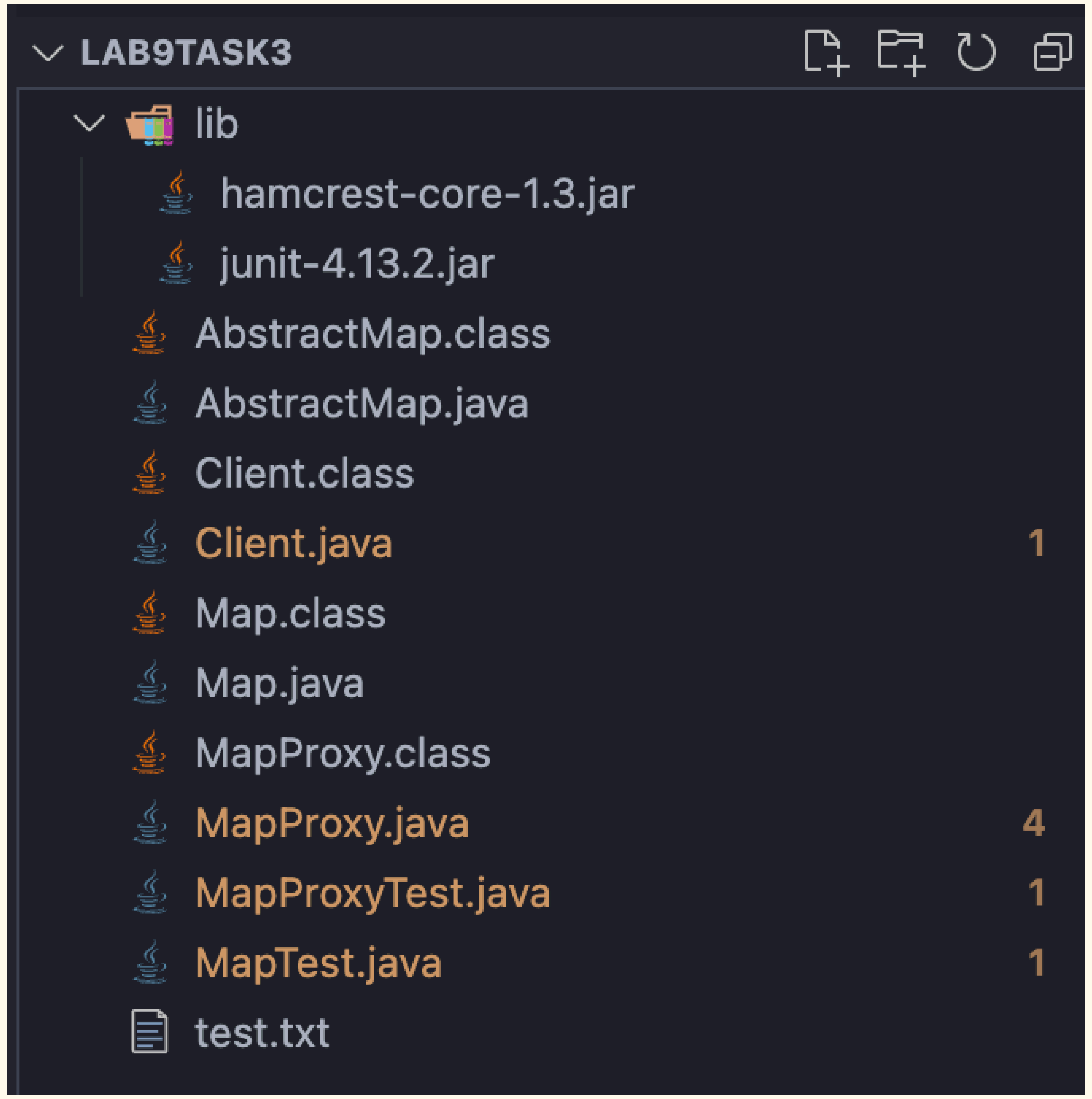
Demo.java OUTPUT



```
Buyer attempting a bid of 55.0 krona
Converting 55.0 kronor to 6.875 dollars
Seller rejects the bid of 6.875 dollars

Buyer attempting a bid of 70.0 krona
Converting 70.0 kronor to 8.75 dollars
Seller rejects the bid of 8.75 dollars

Buyer attempting a bid of 85.0 krona
Converting 85.0 kronor to 10.625 dollars
Seller accepts the bid of 10.625 dollars

Buyer attempting a bid of 3.0 euro
Converting 3.0 euros to 4.285714 dollars
Converting 3.0 euros to 4.285714 dollars
Converting 3.0 euros to 4.285714 dollars
Converting 3.0 euros to 4.285714 dollars
Converting 3.0 euros to 4.285714 dollars
Seller rejects the bid of 4.285714 dollars

Buyer attempting a bid of 4.5 euro
Converting 4.5 euros to 6.4285717 dollars
Seller rejects the bid of 6.4285717 dollars

Buyer attempting a bid of 6.0 euro
Converting 6.0 euros to 8.571428 dollars
Seller rejects the bid of 8.571428 dollars

Buyer attempting a bid of 7.5 euro
Converting 7.5 euros to 10.714286 dollars
Seller accepts the bid of 10.714286 dollars
```

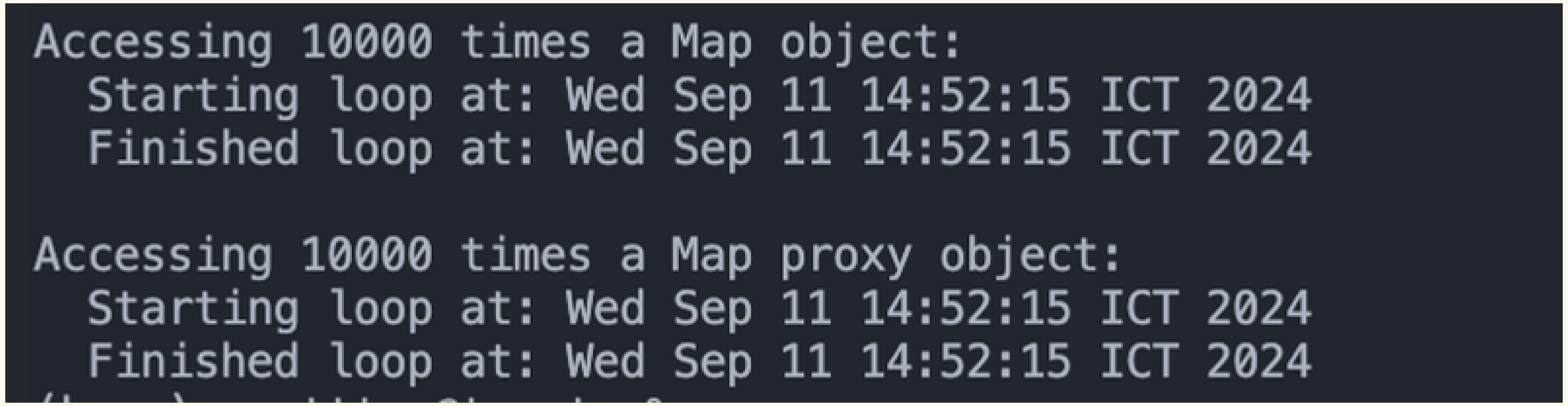
TASK 3

Structure&Library



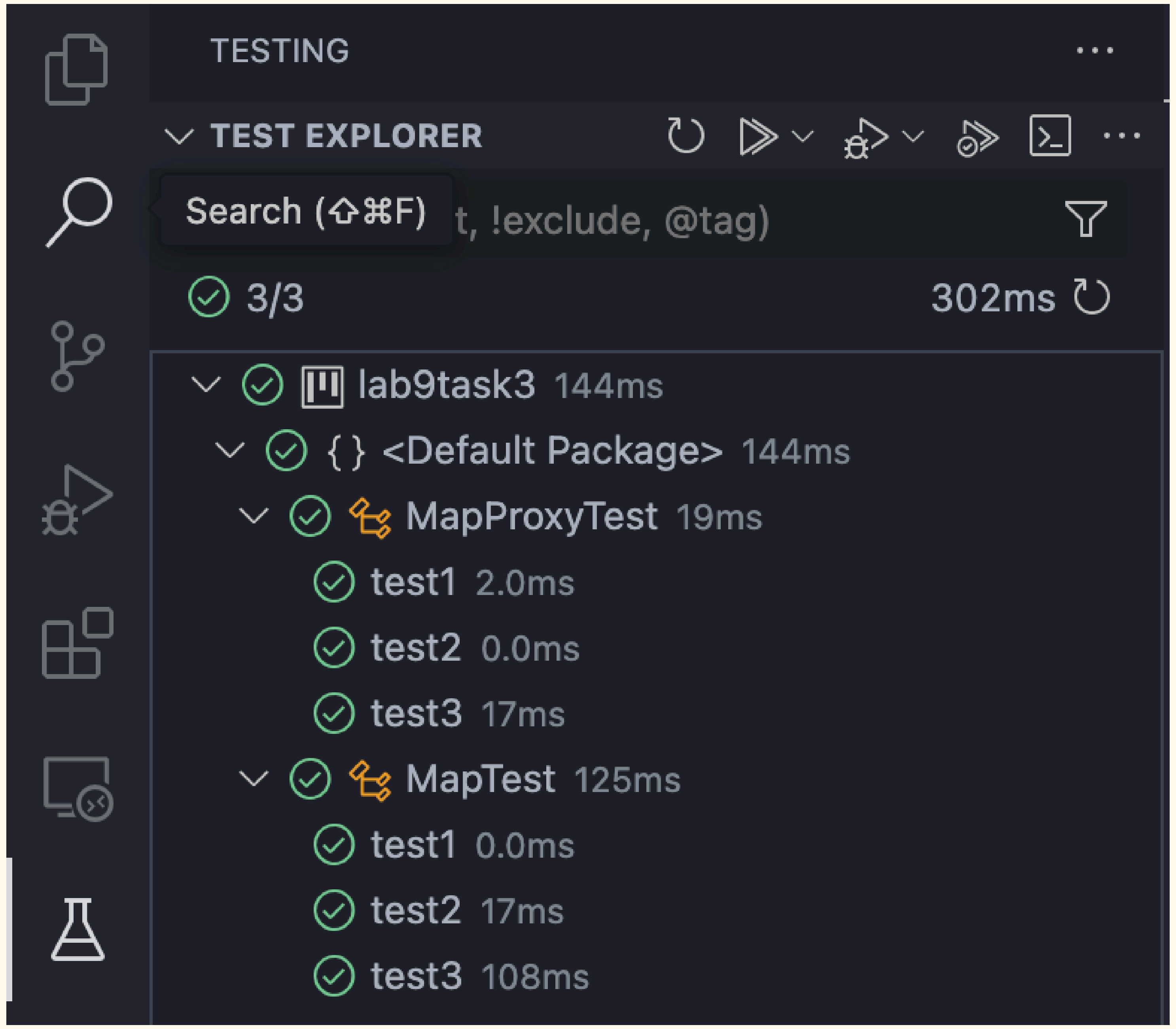
```
LAB9TASK3
  lib
    hamcrest-core-1.3.jar
    junit-4.13.2.jar
  AbstractMap.class
  AbstractMap.java
  Client.class
  Client.java                1
  Map.class
  Map.java
  MapProxy.class
  MapProxy.java              4
  MapProxyTest.java          1
  MapTest.java               1
  test.txt
```

Client.java OUTPUT



```
Accessing 10000 times a Map object:
  Starting loop at: Wed Sep 11 14:52:15 ICT 2024
  Finished loop at: Wed Sep 11 14:52:15 ICT 2024

Accessing 10000 times a Map proxy object:
  Starting loop at: Wed Sep 11 14:52:15 ICT 2024
  Finished loop at: Wed Sep 11 14:52:15 ICT 2024
```

junit Test OUTPUT

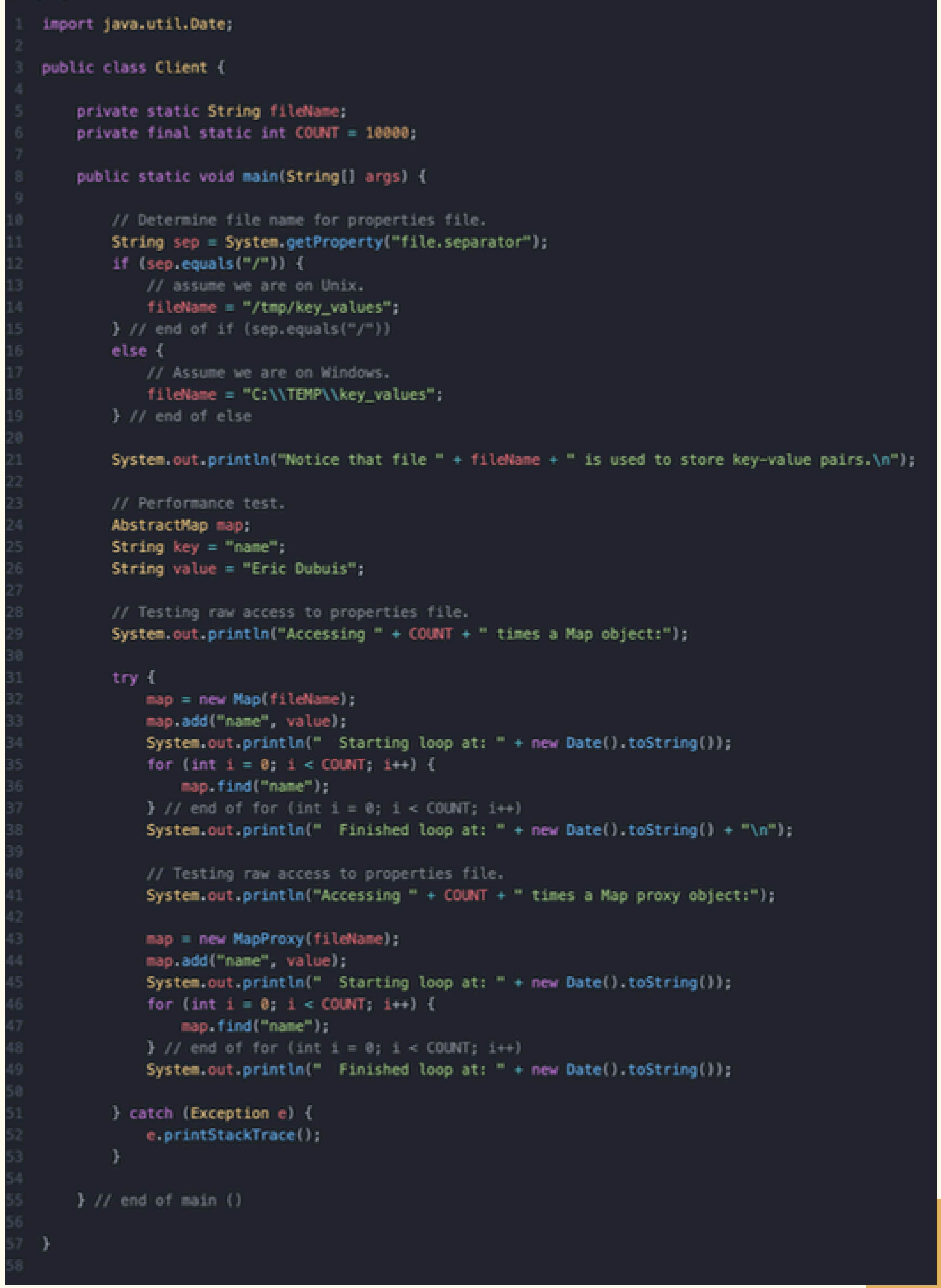

TESTING

TEST EXPLORER

Search (⇧⌘F) t, !exclude, @tag)

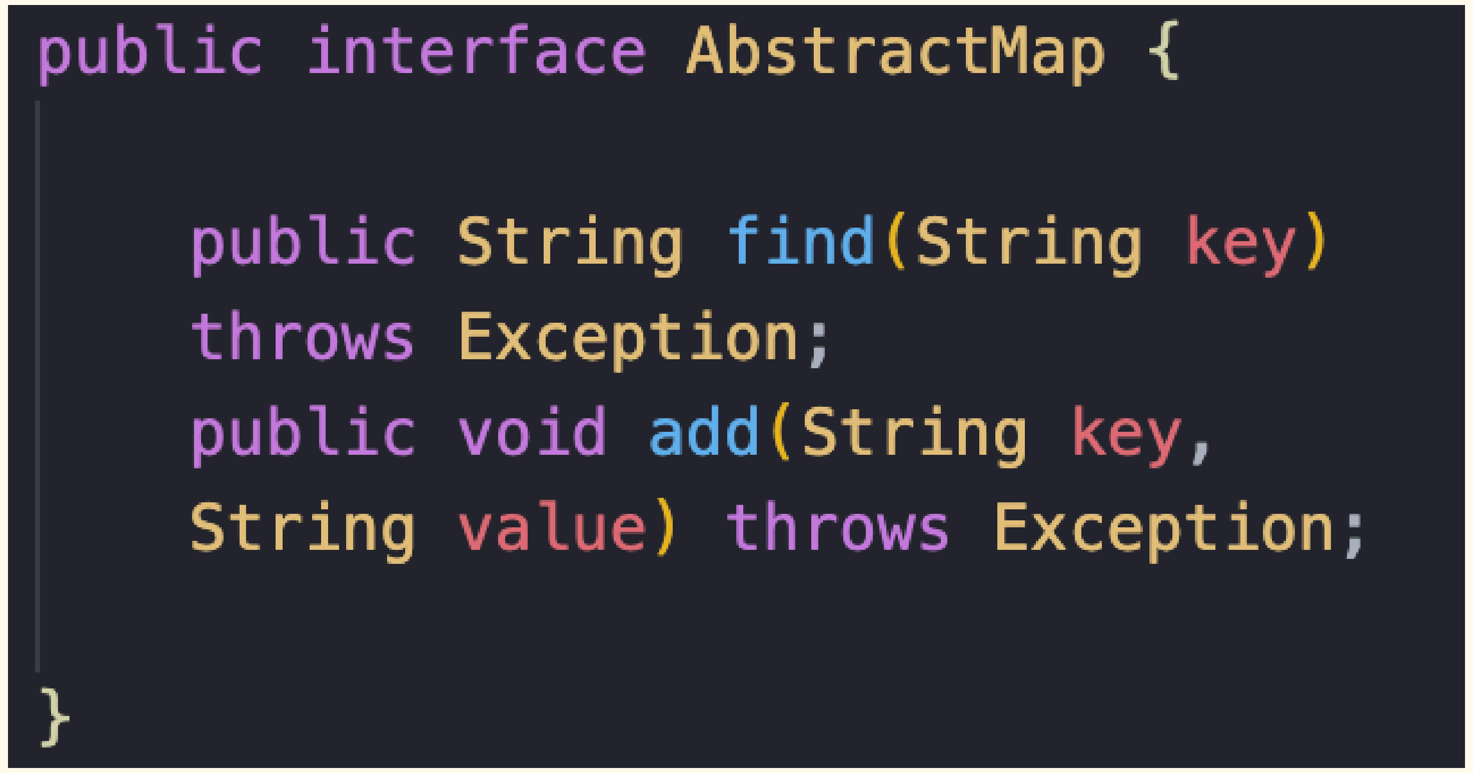
```
3/3                                302ms
lab9task3 144ms
  { } <Default Package> 144ms
    MapProxyTest 19ms
      test1 2.0ms
      test2 0.0ms
      test3 17ms
    MapTest 125ms
      test1 0.0ms
      test2 17ms
      test3 108ms
```

Client.java



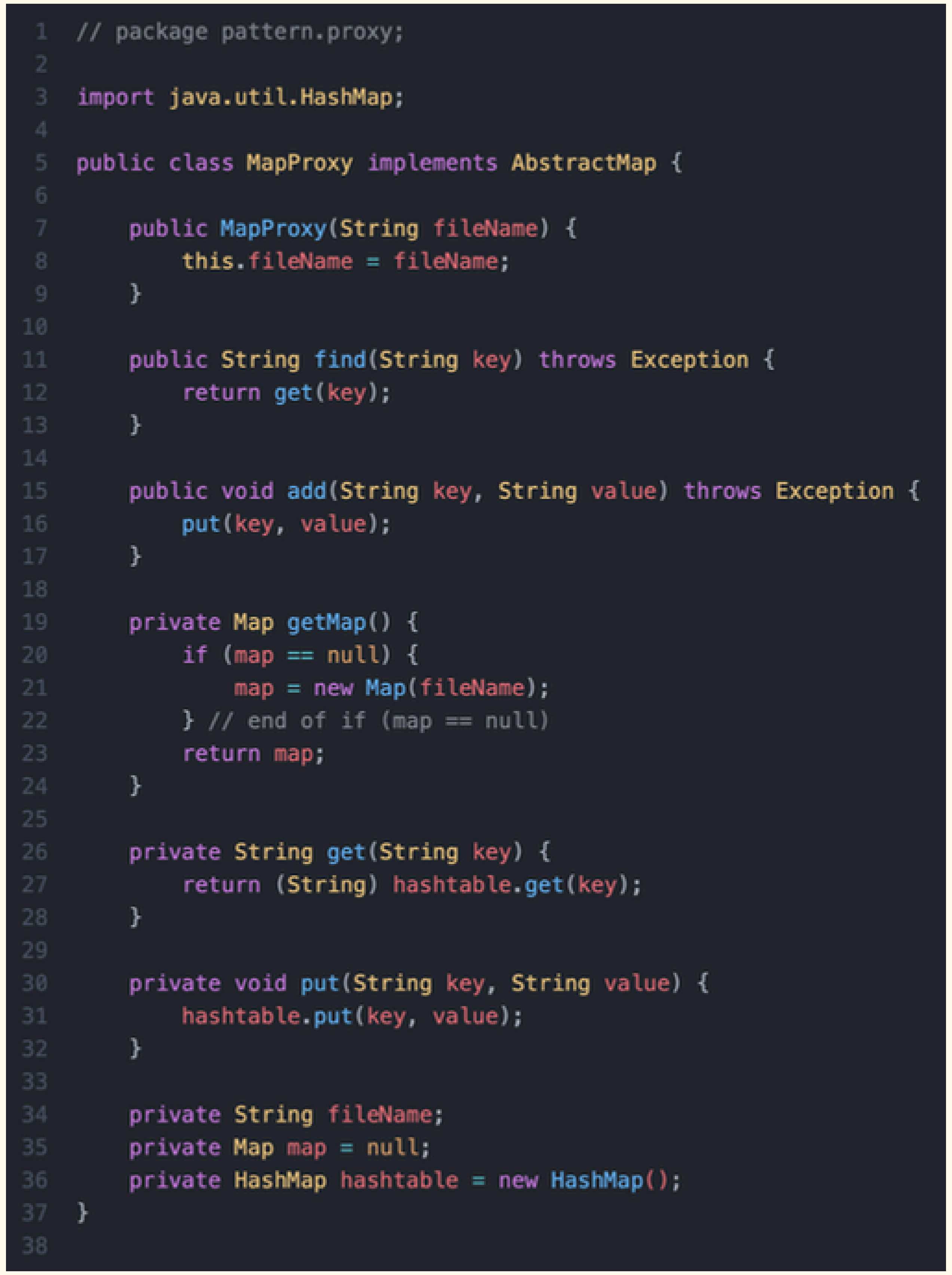
```java
import java.util.Date;

public class Client {

    private static String fileName;
    private final static int COUNT = 10000;

    public static void main(String[] args) {

        // Determine file name for properties file.
        String sep = System.getProperty("file.separator");
        if (sep.equals("/")) {
            // assume we are on Unix.
            fileName = "/tmp/key_values";
        } // end of if (sep.equals("/"))
        else {
            // Assume we are on Windows.
            fileName = "C:\\TEMP\\key_values";
        } // end of else

        System.out.println("Notice that file " + fileName + " is used to store key-value pairs.\n");

        // Performance test.
        AbstractMap map;
        String key = "name";
        String value = "Eric Dubuis";

        // Testing raw access to properties file.
        System.out.println("Accessing " + COUNT + " times a Map object:");

        try {
            map = new Map(fileName);
            map.add("name", value);
            System.out.println("  Starting loop at: " + new Date().toString());
            for (int i = 0; i < COUNT; i++) {
                map.find("name");
            } // end of for (int i = 0; i < COUNT; i++)
            System.out.println("  Finished loop at: " + new Date().toString() + "\n");

            // Testing raw access to properties file.
            System.out.println("Accessing " + COUNT + " times a Map proxy object:");

            map = new MapProxy(fileName);
            map.add("name", value);
            System.out.println("  Starting loop at: " + new Date().toString());
            for (int i = 0; i < COUNT; i++) {
                map.find("name");
            } // end of for (int i = 0; i < COUNT; i++)
            System.out.println("  Finished loop at: " + new Date().toString());

        } catch (Exception e) {
            e.printStackTrace();
        }

    } // end of main ()

}
```

AbstractMap.java



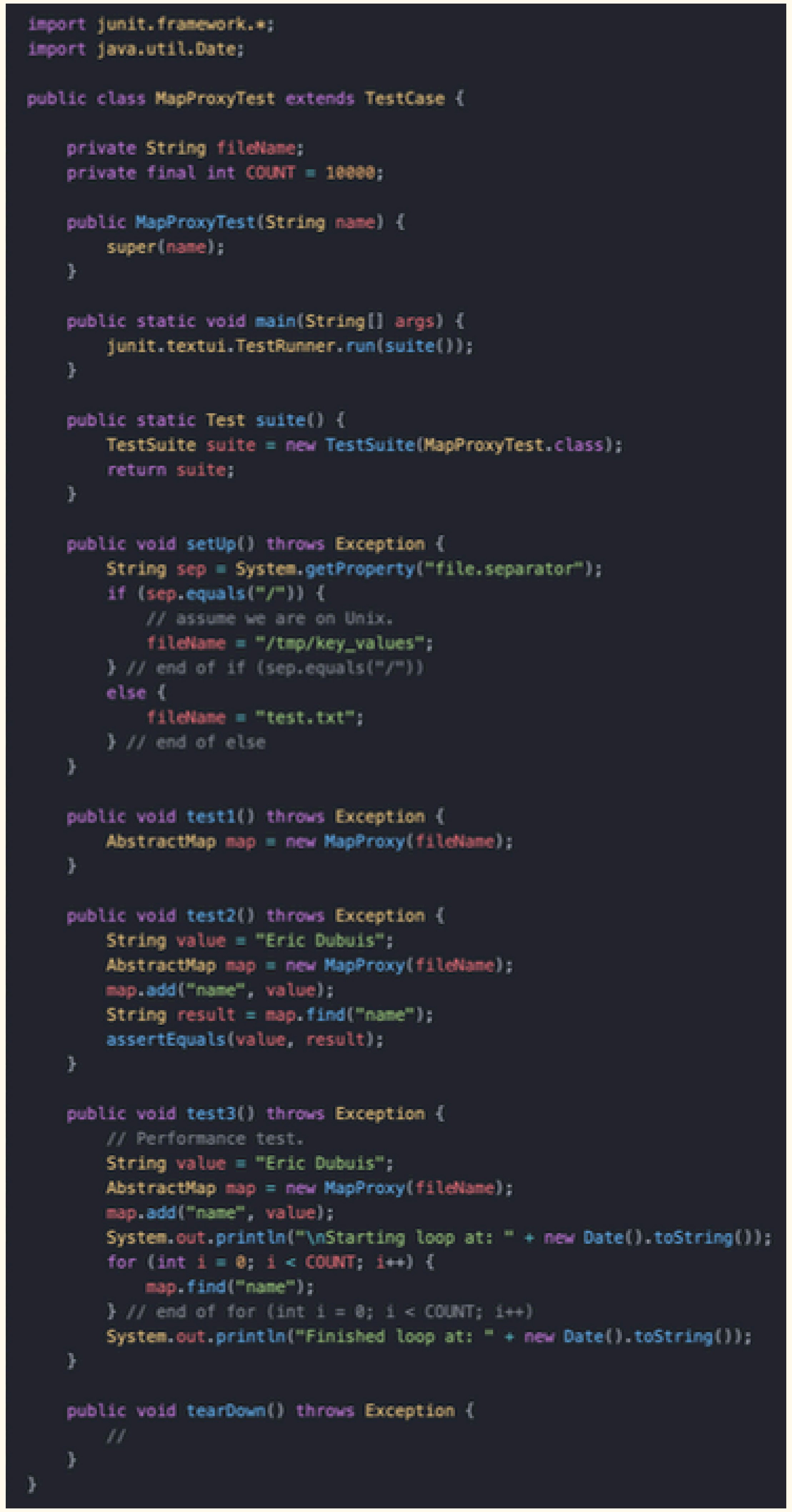
```java
public interface AbstractMap {

    public String find(String key)
    throws Exception;
    public void add(String key,
    String value) throws Exception;

}
```

MapProxy.java



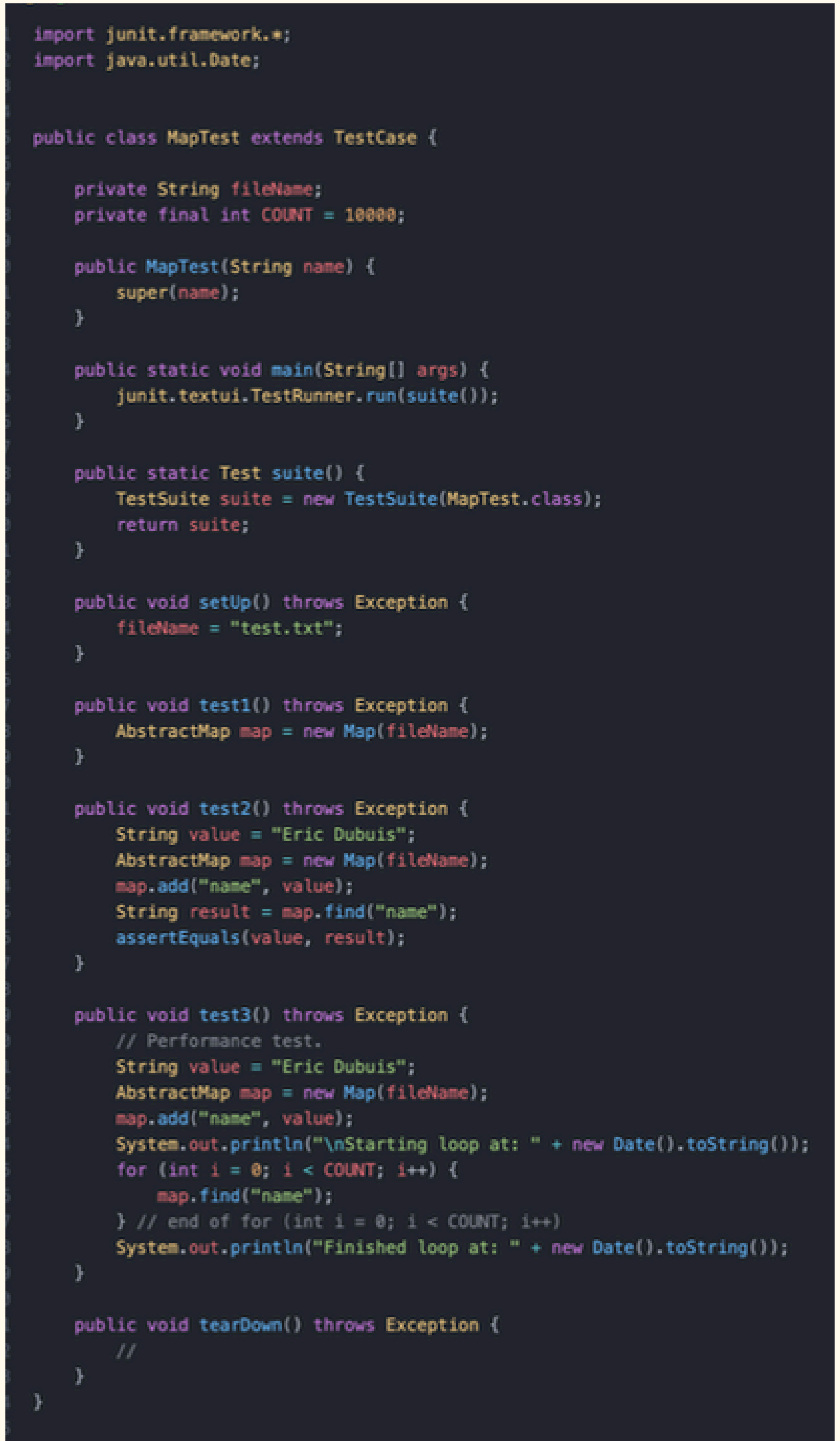
```java
// package pattern.proxy;

import java.util.HashMap;

public class MapProxy implements AbstractMap {

    public MapProxy(String fileName) {
        this.fileName = fileName;
    }

    public String find(String key) throws Exception {
        return get(key);
    }

    public void add(String key, String value) throws Exception {
        put(key, value);
    }

    private Map getMap() {
        if (map == null) {
            map = new Map(fileName);
        } // end of if (map == null)
        return map;
    }

    private String get(String key) {
        return (String) hashtable.get(key);
    }

    private void put(String key, String value) {
        hashtable.put(key, value);
    }

    private String fileName;
    private Map map = null;
    private HashMap hashtable = new HashMap();
}
```

MapProxyTest.java



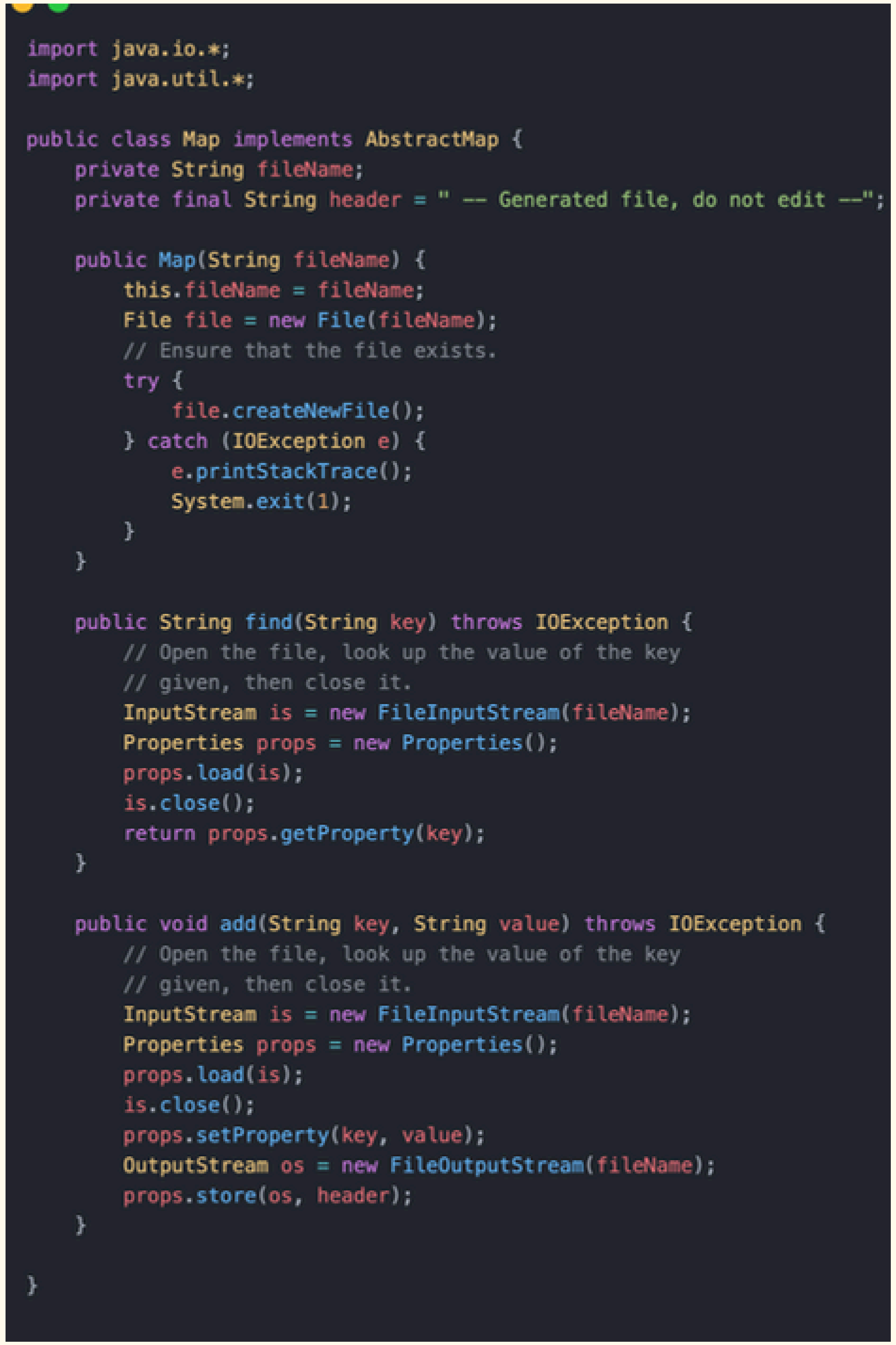
```java
import junit.framework.*;
import java.util.Date;

public class MapProxyTest extends TestCase {

    private String fileName;
    private final int COUNT = 10000;

    public MapProxyTest(String name) {
        super(name);
    }

    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }

    public static Test suite() {
        TestSuite suite = new TestSuite(MapProxyTest.class);
        return suite;
    }

    public void setUp() throws Exception {
        String sep = System.getProperty("file.separator");
        if (sep.equals("/")) {
            // assume we are on Unix.
            fileName = "/tmp/key_values";
        } // end of if (sep.equals("/"))
        else {
            fileName = "test.txt";
        } // end of else
    }

    public void test1() throws Exception {
        AbstractMap map = new MapProxy(fileName);
    }

    public void test2() throws Exception {
        String value = "Eric Dubuis";
        AbstractMap map = new MapProxy(fileName);
        map.add("name", value);
        String result = map.find("name");
        assertEquals(value, result);
    }

    public void test3() throws Exception {
        // Performance test.
        String value = "Eric Dubuis";
        AbstractMap map = new MapProxy(fileName);
        map.add("name", value);
        System.out.println("\nStarting loop at: " + new Date().toString());
        for (int i = 0; i < COUNT; i++) {
            map.find("name");
        } // end of for (int i = 0; i < COUNT; i++)
        System.out.println("Finished loop at: " + new Date().toString());
    }

    public void tearDown() throws Exception {
        //
    }
}
```

MapTest.java



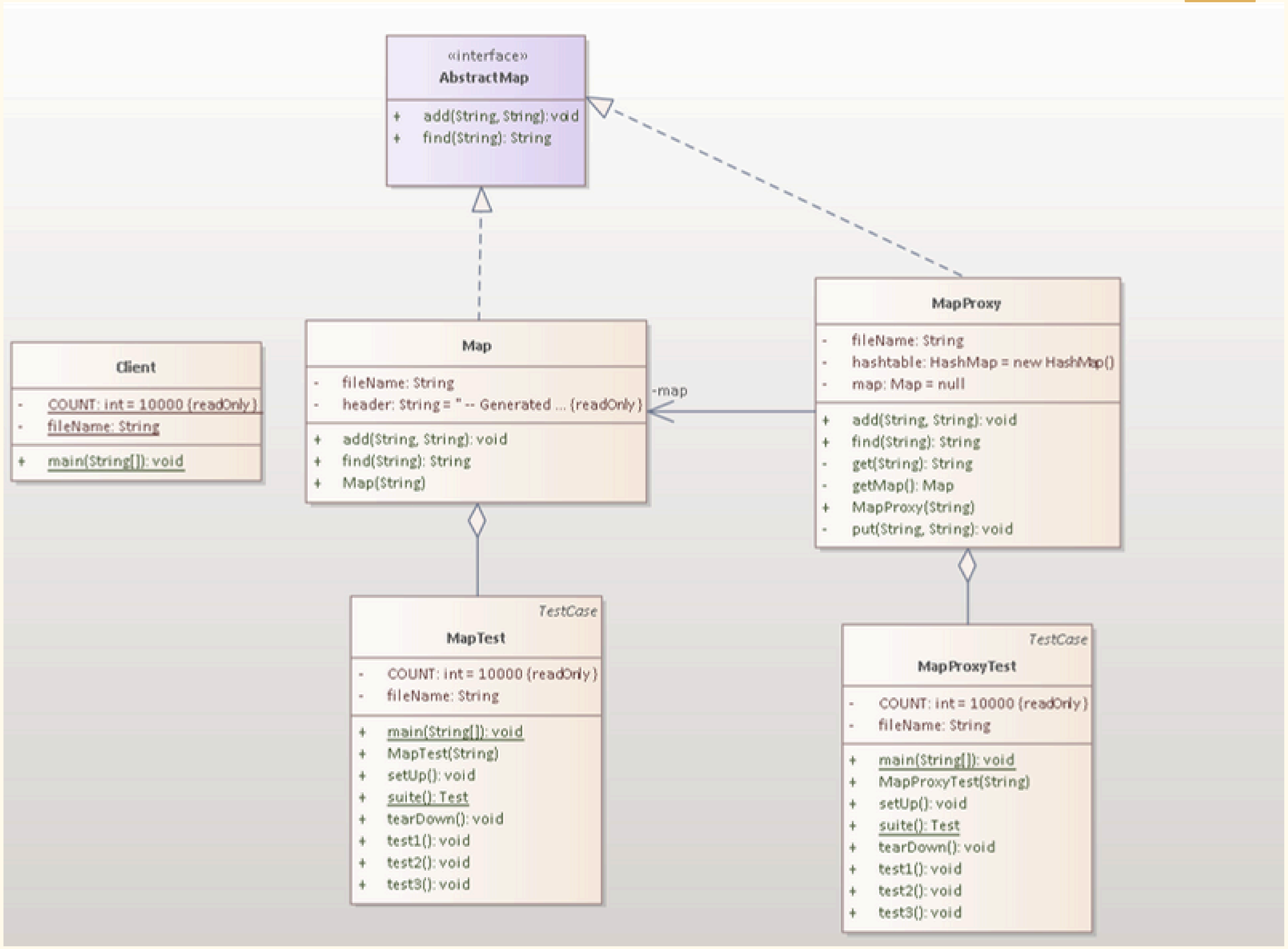
```java
import junit.framework.*;
import java.util.Date;


public class MapTest extends TestCase {

    private String fileName;
    private final int COUNT = 10000;

    public MapTest(String name) {
        super(name);
    }

    public static void main(String[] args) {
        junit.textui.TestRunner.run(suite());
    }

    public static Test suite() {
        TestSuite suite = new TestSuite(MapTest.class);
        return suite;
    }

    public void setUp() throws Exception {
        fileName = "test.txt";
    }

    public void test1() throws Exception {
        AbstractMap map = new Map(fileName);
    }

    public void test2() throws Exception {
        String value = "Eric Dubuis";
        AbstractMap map = new Map(fileName);
        map.add("name", value);
        String result = map.find("name");
        assertEquals(value, result);
    }

    public void test3() throws Exception {
        // Performance test.
        String value = "Eric Dubuis";
        AbstractMap map = new Map(fileName);
        map.add("name", value);
        System.out.println("\nStarting loop at: " + new Date().toString());
        for (int i = 0; i < COUNT; i++) {
            map.find("name");
        } // end of for (int i = 0; i < COUNT; i++)
        System.out.println("Finished loop at: " + new Date().toString());
    }

    public void tearDown() throws Exception {
        //
    }
}
```

Map.java



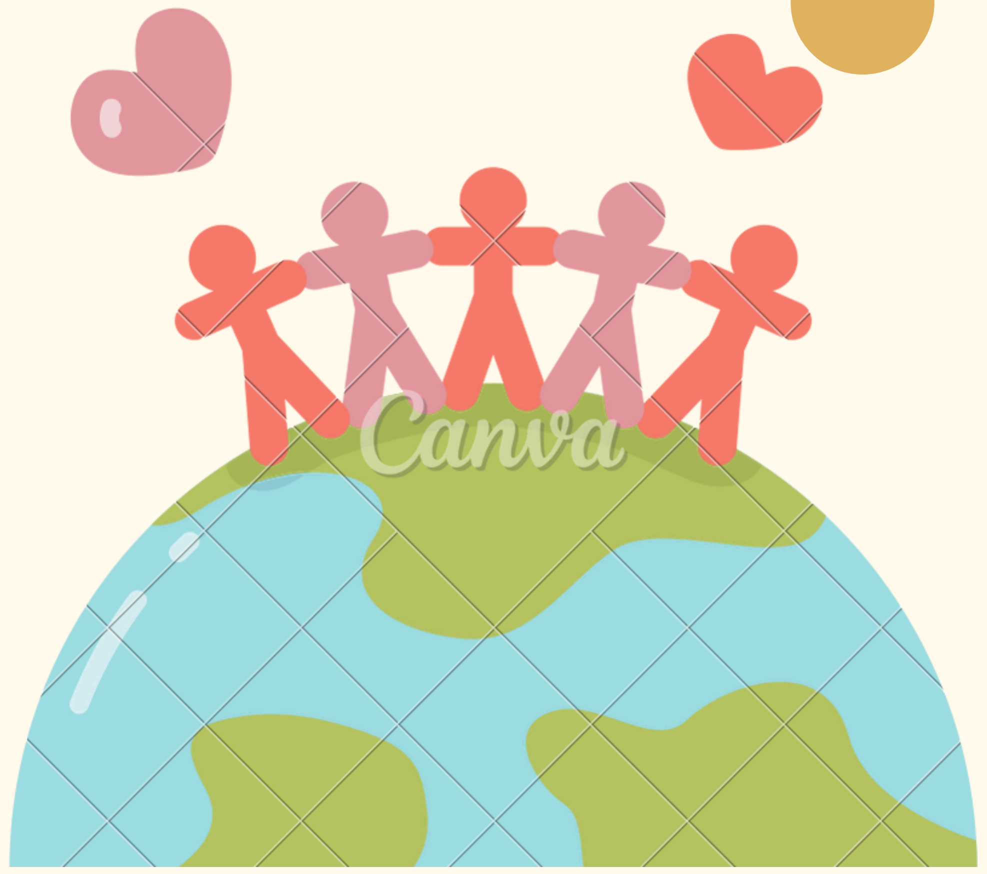
```java
import java.io.*;
import java.util.*;

public class Map implements AbstractMap {
    private String fileName;
    private final String header = " -- Generated file, do not edit --";

    public Map(String fileName) {
        this.fileName = fileName;
        File file = new File(fileName);
        // Ensure that the file exists.
        try {
            file.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }

    public String find(String key) throws IOException {
        // Open the file, look up the value of the key
        // given, then close it.
        InputStream is = new FileInputStream(fileName);
        Properties props = new Properties();
        props.load(is);
        is.close();
        return props.getProperty(key);
    }

    public void add(String key, String value) throws IOException {
        // Open the file, look up the value of the key
        // given, then close it.
        InputStream is = new FileInputStream(fileName);
        Properties props = new Properties();
        props.load(is);
        is.close();
        props.setProperty(key, value);
        OutputStream os = new FileOutputStream(fileName);
        props.store(os, header);
    }

}
```

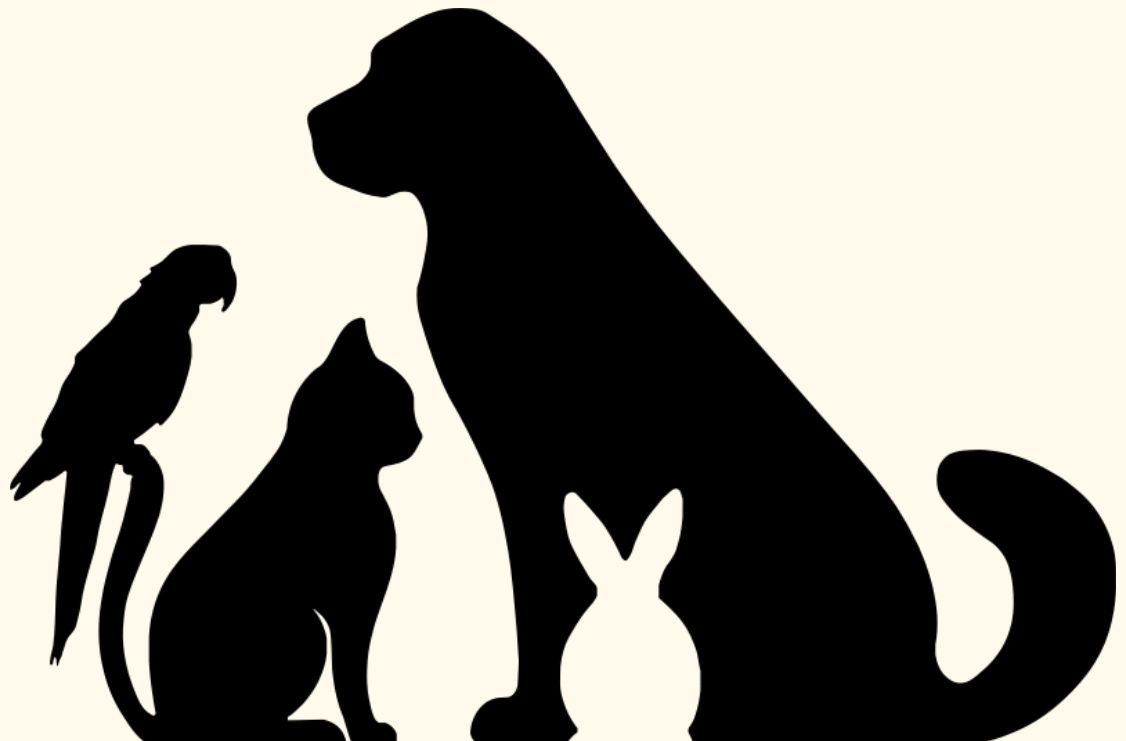
CLASS DIAGRAM



PROBLEM DESCRIPTION

In today’s world, pet owners often face the heartbreaking experience of losing their pets with little hope of finding them efficiently. Traditional methods like posters or word of mouth are slow, and finding a lost pet can often rely too much on luck. Furthermore, local businesses and communities lack an organized platform to assist in locating missing pets, and pet owners do not have a streamlined way to incentivize others to help.





CORE FEATURES:

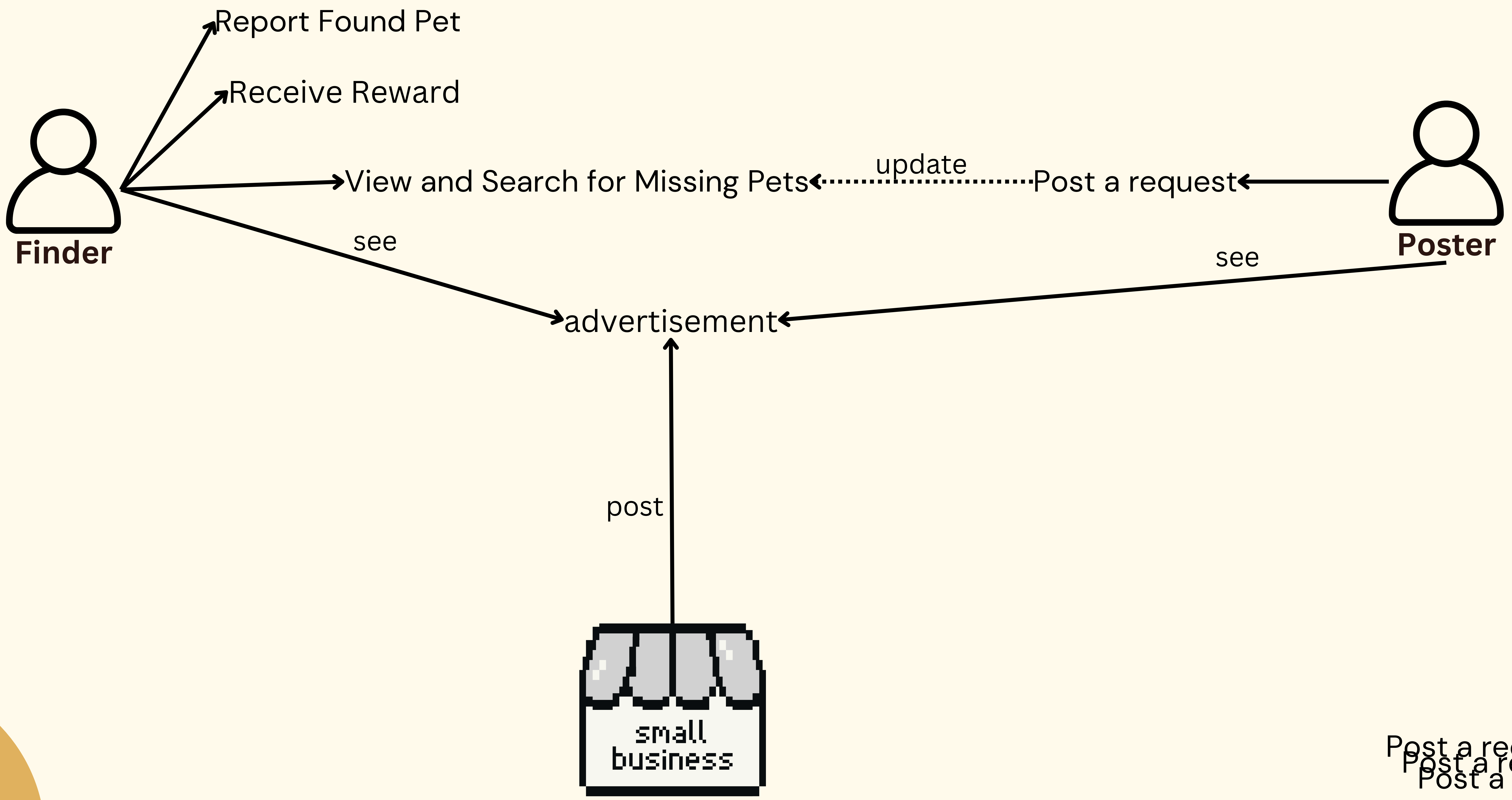
- **Location Tracking:** Pet owners can upload the last known GPS location of their pet, with real-time map integration using GoogleMaps.
- **Community Rewards:** Users who help locate missing pets will be rewarded via PayPal(Mocking), encouraging participation and support.
- **Local Business Involvement:** Pet stores and local businesses can advertise and engage with the platform, providing help in exchange for exposure through GoogleAds.
- **Simplicity:** A user-friendly interface and a monolithic core system ensure simplicity in interactions between the system and the community

PayPal

Google Ads

FUNCTIONAL REQUIREMENTS

USE CASE DIAGRAM:



Post a request

Post a request
Post a request

BRIEF USE CASE DESCRIPTION:

- Use Case 1: Report a Missing Pet (Actor: Pet Owner)
  - A pet goes missing.
- Use Case 2: Help Locate a Pet (Actor: Member, User)
  - A community member wants to help locate a pet.
- Use Case 3: Advertise Local Pet Stores (Actor: Local Business Owner)
  - A local business owner wants to advertise on the platform.

BRIEF USE CASE DESCRIPTION:

Use Case 1: Report a Missing Pet (Actor: Pet Owner)

- Pet owner logs into the "Where’s My Fluffy" app.
- The owner selects the "Report Missing Pet" option.
- Owner enters pet details, last known location, and uploads a photo.
- The system saves the details and displays the pet’s last known location on GoogleMaps.

BRIEF USE CASE DESCRIPTION:

Use Case 2: Help Locate a Pet (Actor: Member, User)

- Community member logs into the app and selects "Help Find Pets."
- The system displays nearby missing pets based on their GPS location using GoogleMaps.
- The community member spots a missing pet and reports its location.
- The system validates the report, and the pet owner is notified.
- If the pet is found, a reward is processed through PayPal to the community member. Post-condition: The pet owner is notified of the pet's location, and the helper is rewarded.

BRIEF USE CASE DESCRIPTION:

Use Case 3: Advertise Local Pet Stores (Actor: Local Business Owner)

- The business owner logs into the app and selects "Advertise Your Store."
- The system integrates with GoogleAds to display relevant ads related to pet services.
- Ads are shown to users based on location and pet-related activities on the platform. Post-condition: The business's ad is displayed, and pet owners are encouraged to engage with the local store.

SYSTEM SEQUENCE DIAGRAM:

Pet Owner

Core System

PRESENTED BY

65011277 Chanasorn Howattanakulphong

65011320 Kanokjan Singhsuwan

65011381 Napatr Sapprasert

65011400 Natthawut Lin

65011462 Phupa Denphatcharangkul

65011558 Suvijuk Samitimata

65011572 Teerapat Senanuch

THANK YOU