

AMRITA SCHOOL OF ENGINEERING
AMRITA VISHWA VIDYAPEETHAM
CHENNAI CAMPUS



19CSE204
Object Oriented Paradigm
Lab Assessment

Submitted by:

ANUSRI S

CH.EN.U4CCE22002

Submitted to:

Dr. S Suthir

Exercise 1

Basic Java Programs

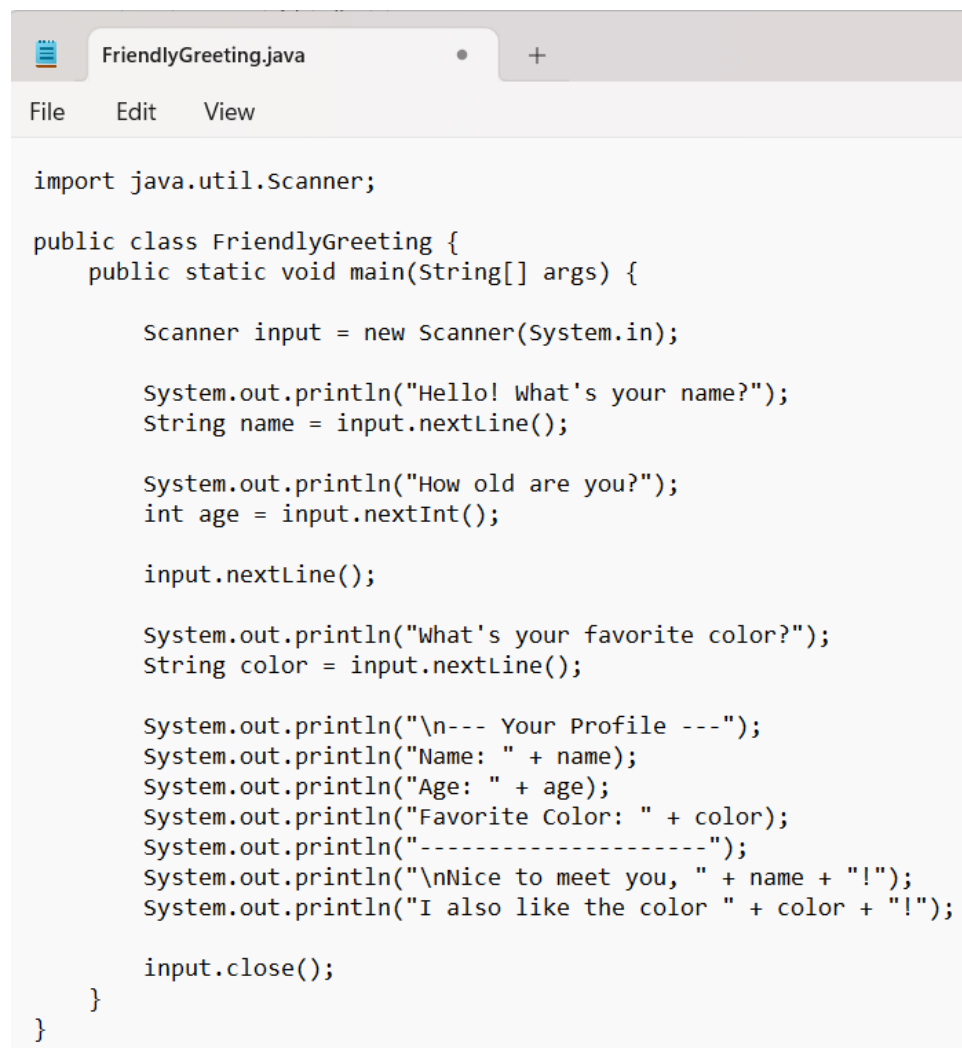
Aim

To implement the following Java programs:

- (a) Input/Output
- (b) Data Types
- (c) Data Operators
- (d) Access Modifiers
- (e) Control Statements

Code

(a) Input/Output



```
import java.util.Scanner;

public class FriendlyGreeting {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        System.out.println("Hello! What's your name?");
        String name = input.nextLine();

        System.out.println("How old are you?");
        int age = input.nextInt();

        input.nextLine();

        System.out.println("What's your favorite color?");
        String color = input.nextLine();

        System.out.println("\n--- Your Profile ---");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Favorite Color: " + color);
        System.out.println("-----");
        System.out.println("\nNice to meet you, " + name + "!");
        System.out.println("I also like the color " + color + "!");

        input.close();
    }
}
```

Output

```
Command Prompt

F:\Java\Programs>javac FriendlyGreeting.java

F:\Java\Programs>java FriendlyGreeting
Hello! What's your name?
Anusri
How old are you?
20
What's your favorite color?
Yellow

--- Your Profile ---
Name: Anusri
Age: 20
Favorite Color: Yellow
-----

Nice to meet you, Anusri!
I also like the color Yellow!
```

(b) Data Types

```
DataTypesDemo.java

File Edit View

public class DataTypesDemo {

    enum ProductCategory {ELECTRONICS, CLOTHING, GROCERIES}

    public static void main(String[] args) {

        byte age = 30;
        short productCode = 1234;
        int quantity = 100;
        long productId = 9876543210L;
        float price = 29.99f;
        double discount = 15.5;
        boolean inStock = true;
        char qualityGrade = 'A';

        String productName = "Wireless Headphones";
        String[] features = {"Bluetooth", "Noise Cancelling", "20hr Battery"};
        ProductCategory category = ProductCategory.ELECTRONICS;
        Object productDetails = "Latest model with premium sound";
        Integer warrantyYears = 2; // Wrapper class

        System.out.println("=== Product Information ===");
        System.out.println("Name: " + productName);
        System.out.println("Category: " + category);
        System.out.println("Price: $" + price);
        System.out.println("Discount: " + discount + "%");
        System.out.println("In Stock: " + (inStock ? "Yes" : "No"));
        System.out.println("Quality Grade: " + qualityGrade);
        System.out.println("Warranty: " + warrantyYears + " years");
        System.out.println("Features: " + String.join(", ", features));
        System.out.println("Product ID: " + productId);
        System.out.println("Age Recommendation: " + age + "+");
        System.out.println("Product Code: " + productCode);
        System.out.println("Available Quantity: " + quantity);
        System.out.println("Details: " + productDetails);
    }
}
```

Output

```
Command Prompt
F:\Java\Programs>java DataTypesDemo
=== Product Information ===
Name: Wireless Headphones
Category: ELECTRONICS
Price: $29.99
Discount: 15.5%
In Stock: Yes
Quality Grade: A
Warranty: 2 years
Features: Bluetooth, Noise Cancelling, 20hr Battery
Product ID: 9876543210
Age Recommendation: 30+
Product Code: 1234
Available Quantity: 100
Details: Latest model with premium sound
```

(c) Data Operators

```
DataOperationsDemo.java
File Edit View

import java.util.Arrays;

public class DataOperationsDemo {
    public static void main(String[] args) {

        int a = 15, b = 4;
        System.out.println("\n Arithmetic Operations:");
        System.out.println(a + " + " + b + " = " + (a + b));
        System.out.println(a + " - " + b + " = " + (a - b));
        System.out.println(a + " * " + b + " = " + (a * b));
        System.out.println(a + " / " + b + " = " + (a / b));
        System.out.println(a + " % " + b + " = " + (a % b));
        System.out.println(a + " / " + b + " (float) = " + ((float)a / b));

        System.out.println("\nRelational Operations:");
        System.out.println(a + " == " + b + " : " + (a == b));
        System.out.println(a + " != " + b + " : " + (a != b));
        System.out.println(a + " > " + b + " : " + (a > b));
        System.out.println(a + " < " + b + " : " + (a < b));
        System.out.println(a + " >= " + b + " : " + (a >= b));
        System.out.println(a + " <= " + b + " : " + (a <= b));

        boolean x = true, y = false;
        System.out.println("\nLogical Operations:");
        System.out.println(x + " && " + y + " : " + (x && y));
        System.out.println(x + " || " + y + " : " + (x || y));
        System.out.println("! " + x + " : " + (!x));

        int num1 = 5; // 0101
        int num2 = 3; // 0011
        System.out.println("\nBitwise Operations:");
        System.out.println(num1 + " & " + num2 + " : " + (num1 & num2));
        System.out.println(num1 + " | " + num2 + " : " + (num1 | num2));
        System.out.println(num1 + " ^ " + num2 + " : " + (num1 ^ num2));
        System.out.println("~" + num1 + " : " + (~num1));
        System.out.println(num1 + " << 1 : " + (num1 << 1));
        System.out.println(num1 + " >> 1 : " + (num1 >> 1));

        System.out.println("\nAssignment Operations:");
        int c = 10;
        System.out.println("c = " + c);
        c += 5; System.out.println("c += 5 → " + c);
        c -= 3; System.out.println("c -= 3 → " + c);
        c *= 2; System.out.println("c *= 2 → " + c);
        c /= 4; System.out.println("c /= 4 → " + c);
        c %= 3; System.out.println("c %= 3 → " + c);
```

```

DataOperationsDemo.java
File Edit View
c %= 3; System.out.println("c %= 3 → " + c);

System.out.println("\nIncrement/Decrement:");
int counter = 5;
System.out.println("Original: " + counter);
System.out.println("Post-increment: " + counter++);
System.out.println("After increment: " + counter);
System.out.println("Pre-increment: " + ++counter);
System.out.println("Post-decrement: " + counter--);
System.out.println("After decrement: " + counter);

System.out.println("\nTernary Operation:");
int max = (a > b) ? a : b;
System.out.println("Max between " + a + " and " + b + " is " + max);

System.out.println("\nType Casting:");
double d = 123.456;
int i = (int)d; // Explicit casting
System.out.println("double " + d + " → int " + i);

char ch = 'A';
int ascii = ch; // Implicit casting (char to int)
System.out.println("char '" + ch + "' → ASCII " + ascii);

System.out.println("\nString Operations:");
String str1 = "Hello";
String str2 = "World";
System.out.println("Concatenation: " + str1 + " " + str2);
System.out.println("Length: " + str1.length());
System.out.println("Uppercase: " + str1.toUpperCase());
System.out.println("Substring: " + str1.substring(1, 3));
System.out.println("Equals: " + str1.equals("hello"));
System.out.println("EqualsIgnoreCase: " + str1.equalsIgnoreCase("hello"));

System.out.println("\nArray Operations:");
int[] numbers = {3, 1, 4, 2};
System.out.println("Original: " + Arrays.toString(numbers));
Arrays.sort(numbers);
System.out.println("Sorted: " + Arrays.toString(numbers));
System.out.println("Binary Search (4): " + Arrays.binarySearch(numbers, 4));
}

```

Output

```

Command Prompt
F:\Java\Programs>java DataOperationsDemo

Arithmetic Operations:
15 + 4 = 19
15 - 4 = 11
15 * 4 = 60
15 / 4 = 3
15 % 4 = 3
15 / 4 (float) = 3.75

Relational Operations:
15 == 4 : false
15 != 4 : true
15 > 4 : true
15 < 4 : false
15 >= 4 : true
15 <= 4 : false

Logical Operations:
true && false : false
true || false : true
!true : false

Bitwise Operations:
5 & 3 : 1
5 | 3 : 7
5 ^ 3 : 6
~5 : -6
5 << 1 : 10
5 >> 1 : 2

Assignment Operations:
c = 10
c += 5 ? 15
c -= 3 ? 12
c *= 2 ? 24
c /= 4 ? 6
c %= 3 ? 0

```

```
Command Prompt

Increment/Decrement:
Original: 5
Post-increment: 5
After increment: 6
Pre-increment: 7
Post-decrement: 7
After decrement: 6

Ternary Operation:
Max between 15 and 4 is 15

Type Casting:
double 123.456 ? int 123
char 'A' ? ASCII 65

String Operations:
Concatenation: Hello World
Length: 5
Uppercase: HELLO
Substring: el
Equals: false
EqualsIgnoreCase: true

Array Operations:
Original: [3, 1, 4, 2]
Sorted: [1, 2, 3, 4]
Binary Search (4): 3
```

(d) Access Modifiers

```
AccessDemo.java

File Edit View

public class AccessDemo {
    public static void main(String[] args) {
        Person p1 = new Person("Alice");
        p1.display();
        Person p2 = new Person("Bob");
        p2.secretMethod();
    }
}

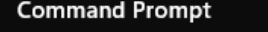
class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }

    public void display() {
        System.out.println("Name: " + name);
    }

    protected void secretMethod() {
        System.out.println("Shhh! This is protected");
    }
}
```

Output



The screenshot shows a Windows Command Prompt window with a dark background. The title bar at the top reads "Command Prompt" and includes standard window controls (minimize, maximize, close). The command prompt shows the following text:

```
F:\Java\Programs>java AccessDemo
Name: Alice
Shhh! This is protected
```

(e) Control Statements

```

import java.util.Scanner;

public class AdventureGame {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int health = 100;
        int gold = 0;
        boolean hasKey = false;
        boolean gameOver = false;

        System.out.println("Welcome to the Dragon's Lair Adventure!");
        System.out.println("-----");
        System.out.println("Your village is under attack! Defeat the dragon to save your people.");
        System.out.println("Health: " + health + " | Gold: " + gold + "\n");

        while (!gameOver) {
            System.out.println("\nChoose your action:");
            System.out.println("1. Explore the forest");
            System.out.println("2. Enter the cave");
            System.out.println("3. Visit the shop");
            System.out.println("4. Face the dragon");
            System.out.println("5. Quit game");
            System.out.print("Enter choice (1-5): ");

            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.println("\nYou find a healing herb in the forest!");
                    health = Math.min(100, health + 20);
                    System.out.println("Health increased to: " + health);
                    break;

                case 2:
                    if (!hasKey) {
                        System.out.println("\nYou discover a golden key in the cave!");
                        hasKey = true;
                    } else {
                        System.out.println("\nYou already have the key. Nothing else here.");
                    }
                    break;

                case 3:
                    System.out.println("\nShopkeeper: '5 gold for a health potion!'");
                    if (gold >= 5) {
                        System.out.println("Do you want to buy? (1=Yes, 2=No)");
                        int buyChoice = scanner.nextInt();

                        if (buyChoice == 1) {
                            gold -= 5;
                            health = Math.min(100, health + 30);
                            System.out.println("Health restored to: " + health);
                        } else {
                            System.out.println("Maybe next time!");
                        }
                    } else {
                        System.out.println("Not enough gold!");
                    }
                    break;

                case 4:
                    System.out.println("\nYou face the dragon!");
                    health -= 50;
                    System.out.println("Your health is now: " + health);
                    if (health <= 0) {
                        System.out.println("You were defeated by the dragon. Game Over!");
                        gameOver = true;
                    }
                    break;

                case 5:
                    System.out.println("Thank you for playing! Goodbye!");
                    gameOver = true;
                    break;

                default:
                    System.out.println("Invalid choice. Please enter a number between 1 and 5.");
            }
        }
    }
}

```

```

AdventureGame.java
File Edit View

    case 4:
        System.out.println("\nThe dragon roars as you approach!");

        String weaponStatus = hasKey ? "dragon-slaying sword" : "rusty dagger";
        System.out.println("You attack with your " + weaponStatus + "!");

        if (!hasKey) {
            System.out.println("Your dagger breaks! The dragon burns you (-50 health)");
            health -= 50;
        } else if (Math.random() > 0.3) {
            System.out.println("You slay the dragon! Victory!");
            gold += 100;
            gameOver = true;
        } else {
            System.out.println("The dragon dodges! You get burned (-30 health)");
            health -= 30;
        }

        System.out.println("Health: " + health);

        if (health <= 0) {
            System.out.println("You died! Game over.");
            gameOver = true;
        }
        break;

    case 5:
        System.out.println("\nThanks for playing!");
        gameOver = true;
        break;

    default:
        System.out.println("\nInvalid choice! Try again.");
}

if (!gameOver && Math.random() > 0.7) {
    int foundGold = (int)(Math.random() * 10) + 1;
    gold += foundGold;
    System.out.println("\nYou found " + foundGold + " gold on the ground!");
}

System.out.println("\nHealth: " + health + " | Gold: " + gold);
}
scanner.close();
}
}

```

Output

```

Command Prompt

F:\Java\Programs>java AdventureGame
Welcome to the Dragon's Lair Adventure!
-----
Your village is under attack! Defeat the dragon to save your people.
Health: 100 | Gold: 0

Choose your action:
1. Explore the forest
2. Enter the cave
3. Visit the shop
4. Face the dragon
5. Quit game
Enter choice (1-5): 2

You discover a golden key in the cave!

You found 9 gold on the ground!

Health: 100 | Gold: 9

Choose your action:
1. Explore the forest
2. Enter the cave
3. Visit the shop
4. Face the dragon
5. Quit game
Enter choice (1-5): 4

The dragon roars as you approach!
You attack with your dragon-slaying sword!
You slay the dragon! Victory!
Health: 100

Health: 100 | Gold: 109

```


Conclusion

Therefore, all five fundamental Java programs have been successfully implemented. These programs demonstrated the following: getting input from the user and displaying output; exploring the function of various data types; performing data operations (including arithmetic, logical, increment/decrement, assignment, array, string, relational, bitwise, and ternary); controlling access to class variables using public and private modifiers; and utilizing control statements such as switch-case, while, and if-else.

Exercise 2

Inheritance & Polymorphism

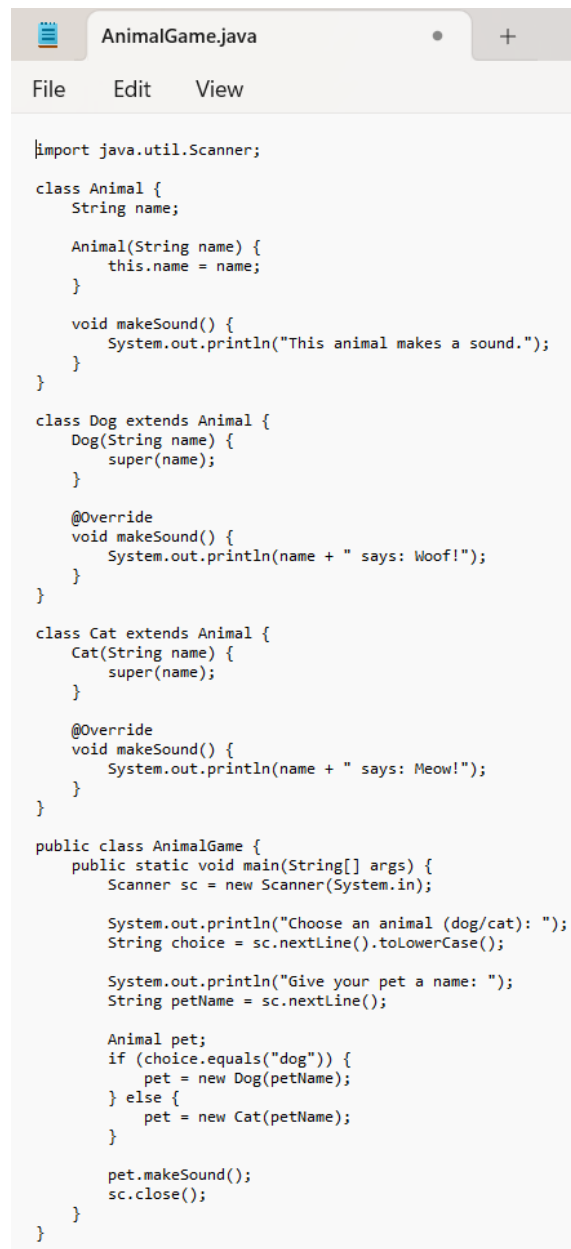
Aim

To demonstrate code reusability and behavioural flexibility in Java by practically implementing inheritance and polymorphism.

Code

(a) Inheritance

(i) Single-level



```
AnimalGame.java

File Edit View

import java.util.Scanner;

class Animal {
    String name;

    Animal(String name) {
        this.name = name;
    }

    void makeSound() {
        System.out.println("This animal makes a sound.");
    }
}

class Dog extends Animal {
    Dog(String name) {
        super(name);
    }

    @Override
    void makeSound() {
        System.out.println(name + " says: Woof!");
    }
}

class Cat extends Animal {
    Cat(String name) {
        super(name);
    }

    @Override
    void makeSound() {
        System.out.println(name + " says: Meow!");
    }
}

public class AnimalGame {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

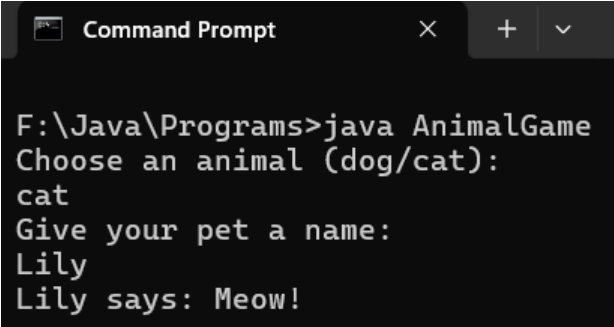
        System.out.println("Choose an animal (dog/cat): ");
        String choice = sc.nextLine().toLowerCase();

        System.out.println("Give your pet a name: ");
        String petName = sc.nextLine();

        Animal pet;
        if (choice.equals("dog")) {
            pet = new Dog(petName);
        } else {
            pet = new Cat(petName);
        }

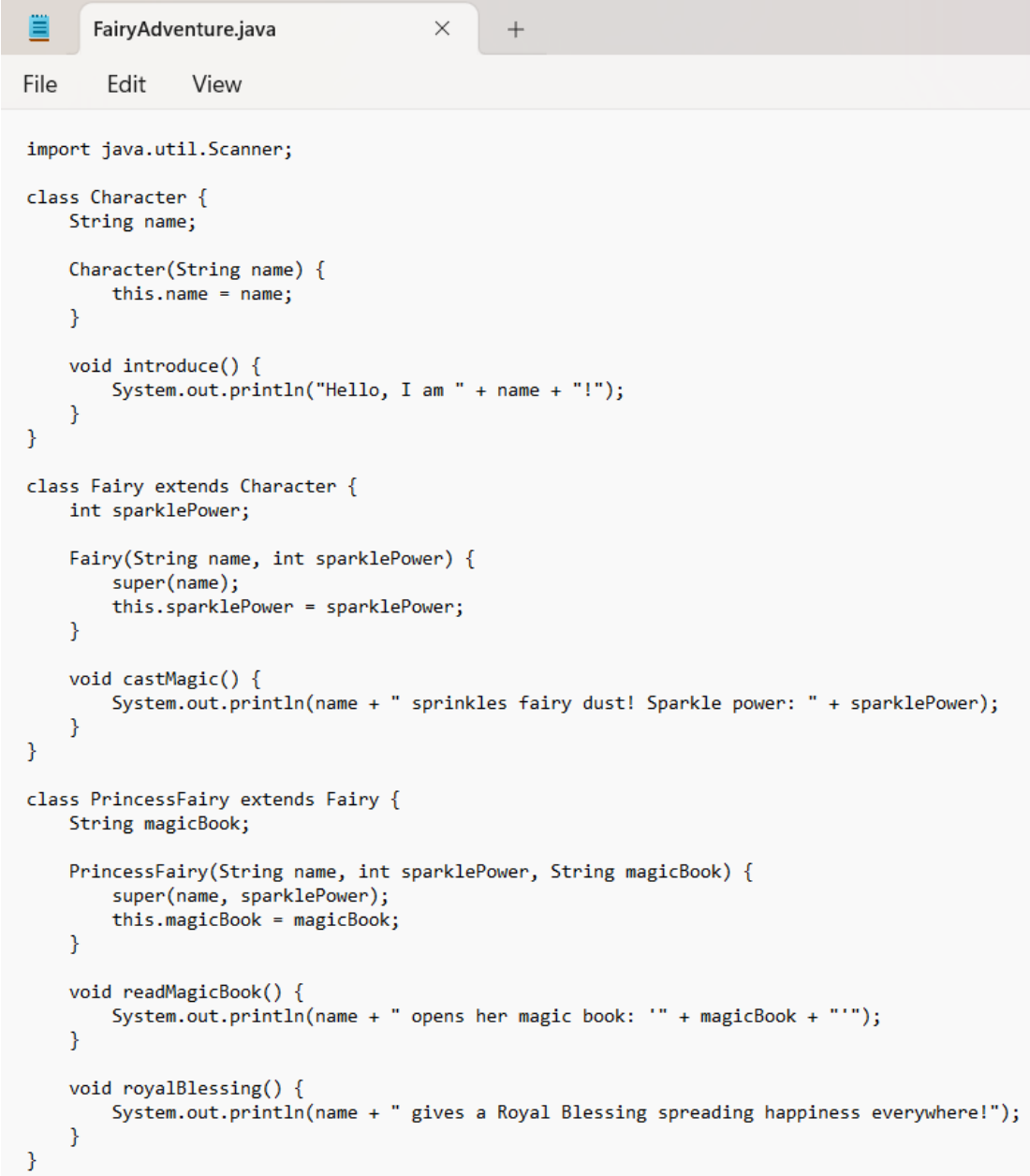
        pet.makeSound();
        sc.close();
    }
}
```

Output



```
Command Prompt
F:\Java\Programs>java AnimalGame
Choose an animal (dog/cat):
cat
Give your pet a name:
Lily
Lily says: Meow!
```

(ii) Multi-level



```
FairyAdventure.java
File Edit View

import java.util.Scanner;

class Character {
    String name;

    Character(String name) {
        this.name = name;
    }

    void introduce() {
        System.out.println("Hello, I am " + name + "!");
    }
}

class Fairy extends Character {
    int sparklePower;

    Fairy(String name, int sparklePower) {
        super(name);
        this.sparklePower = sparklePower;
    }

    void castMagic() {
        System.out.println(name + " sprinkles fairy dust! Sparkle power: " + sparklePower);
    }
}

class PrincessFairy extends Fairy {
    String magicBook;

    PrincessFairy(String name, int sparklePower, String magicBook) {
        super(name, sparklePower);
        this.magicBook = magicBook;
    }

    void readMagicBook() {
        System.out.println(name + " opens her magic book: '" + magicBook + "'");
    }

    void royalBlessing() {
        System.out.println(name + " gives a Royal Blessing spreading happiness everywhere!");
    }
}
```

```
FairyAdventure.java
File Edit View
}

public class FairyAdventure {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your fairy name: ");
        String fairyName = sc.nextLine();

        System.out.print("Enter your sparkle power (1-100): ");
        int sparkle = sc.nextInt();
        sc.nextLine();

        System.out.print("Enter the title of your magic book: ");
        String book = sc.nextLine();

        PrincessFairy fairy = new PrincessFairy(fairyName, sparkle, book);

        System.out.println("\nWelcome to the Fairy Tale Adventure!");
        fairy.introduce();

        boolean playing = true;
        while (playing) {
            System.out.println("\nChoose an action:");
            System.out.println("1) Sprinkle Magic Dust");
            System.out.println("2) Read from Magic Book");
            System.out.println("3) Give a Royal Blessing");
            System.out.println("4) Exit Game");

            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    fairy.castMagic();
                    break;
                case 2:
                    fairy.readMagicBook();
                    break;
                case 3:
                    fairy.royalBlessing();
                    break;
                case 4:
                    System.out.println("Goodbye, Princess Fairy " + fairy.name + "!");
                    playing = false;
                    break;
                default:
                    System.out.println("Choose a valid option, sweet fairy!");
            }
        }

        sc.close();
    }
}
```

Output

```
Command Prompt
F:\Java\Programs>java FairyAdventure
Enter your fairy name: Spark
Enter your sparkle power (1-100): 90
Enter the title of your magic book: Enchanted World

Welcome to the Fairy Tale Adventure!
Hello, I am Spark!

Choose an action:
1) Sprinkle Magic Dust
2) Read from Magic Book
3) Give a Royal Blessing
4) Exit Game
3
Spark gives a Royal Blessing spreading happiness everywhere!
```

```

Choose an action:
1) Sprinkle Magic Dust
2) Read from Magic Book
3) Give a Royal Blessing
4) Exit Game
1
Spark sprinkles fairy dust! Sparkle power: 90

Choose an action:
1) Sprinkle Magic Dust
2) Read from Magic Book
3) Give a Royal Blessing
4) Exit Game
2
Spark opens her magic book: 'Enchanted World'

Choose an action:
1) Sprinkle Magic Dust
2) Read from Magic Book
3) Give a Royal Blessing
4) Exit Game
4
Goodbye, Princess Fairy Spark!

```

(b) Polymorphism

(i)

```

PaymentPolymorphism.java
File Edit View

import java.util.Scanner;

class Payment {
    void pay(double amount) {
        System.out.println("Processing generic payment of $" + amount);
    }
}

class CreditCard extends Payment {
    @Override
    void pay(double amount) {
        System.out.println("Credit Card: Paid $" + amount);
    }
}

class PayPal extends Payment {
    @Override
    void pay(double amount) {
        System.out.println("PayPal: Paid $" + amount);
    }
}

class UPI extends Payment {
    @Override
    void pay(double amount) {
        System.out.println("UPI: Paid $" + amount);
    }
}

public class PaymentPolymorphism {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter amount to pay: ");
        double amount = sc.nextDouble();

        System.out.println("Choose payment method: ");
        System.out.println("1) Credit Card 2) PayPal 3) UPI");
        int choice = sc.nextInt();

        Payment p;
        switch (choice) {
            case 1: p = new CreditCard(); break;
            case 2: p = new PayPal(); break;
            case 3: p = new UPI(); break;
            default: p = new Payment();
        }

        p.pay(amount); // Runtime Polymorphism
        sc.close();
    }
}

```

Output

```
Command Prompt
F:\Java\Programs>java PaymentPolymorphism
Enter amount to pay: 500
Choose payment method:
1) Credit Card  2) PayPal  3) UPI
1
Credit Card: Paid $500.0
```

(ii)

```
SocialMediaPolymorphism.java
File Edit View

import java.util.Scanner;

class Notification {
    void sendNotification() {
        System.out.println("You have a new notification!");
    }
}

class WhatsApp extends Notification {
    @Override
    void sendNotification() {
        System.out.println("WhatsApp: You got a new message!");
    }
}

class Instagram extends Notification {
    @Override
    void sendNotification() {
        System.out.println("Instagram: Someone liked your photo!");
    }
}

class YouTube extends Notification {
    @Override
    void sendNotification() {
        System.out.println("YouTube: New video uploaded from your subscription!");
    }
}

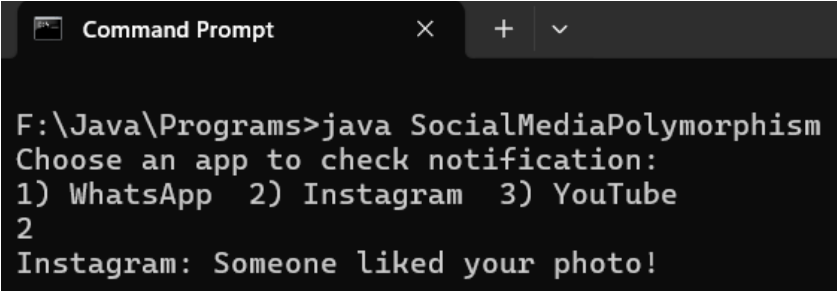
public class SocialMediaPolymorphism {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Choose an app to check notification:");
        System.out.println("1) WhatsApp 2) Instagram 3) YouTube");
        int choice = sc.nextInt();

        Notification n;
        switch (choice) {
            case 1: n = new WhatsApp(); break;
            case 2: n = new Instagram(); break;
            case 3: n = new YouTube(); break;
            default: n = new Notification();
        }

        n.sendNotification(); // Runtime Polymorphism
        sc.close();
    }
}
```

Output

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt" with standard window controls (close, maximize, and a dropdown arrow). The command prompt shows the following text:

```
F:\Java\Programs>java SocialMediaPolymorphism
Choose an app to check notification:
1) WhatsApp 2) Instagram 3) YouTube
2
Instagram: Someone liked your photo!
```

Conclusion

The inheritance programs successfully demonstrated both single-level (AnimalGame) and multi-level (FairyAdventure) class hierarchies for code reuse. The polymorphism codes (Payment and SocialMedia) effectively demonstrated method overriding, allowing a single interface to trigger different behaviours dynamically. All programs combined show how inheritance creates structured code, while polymorphism makes it flexible and powerful.

Exercise 3

Encapsulation & Abstraction

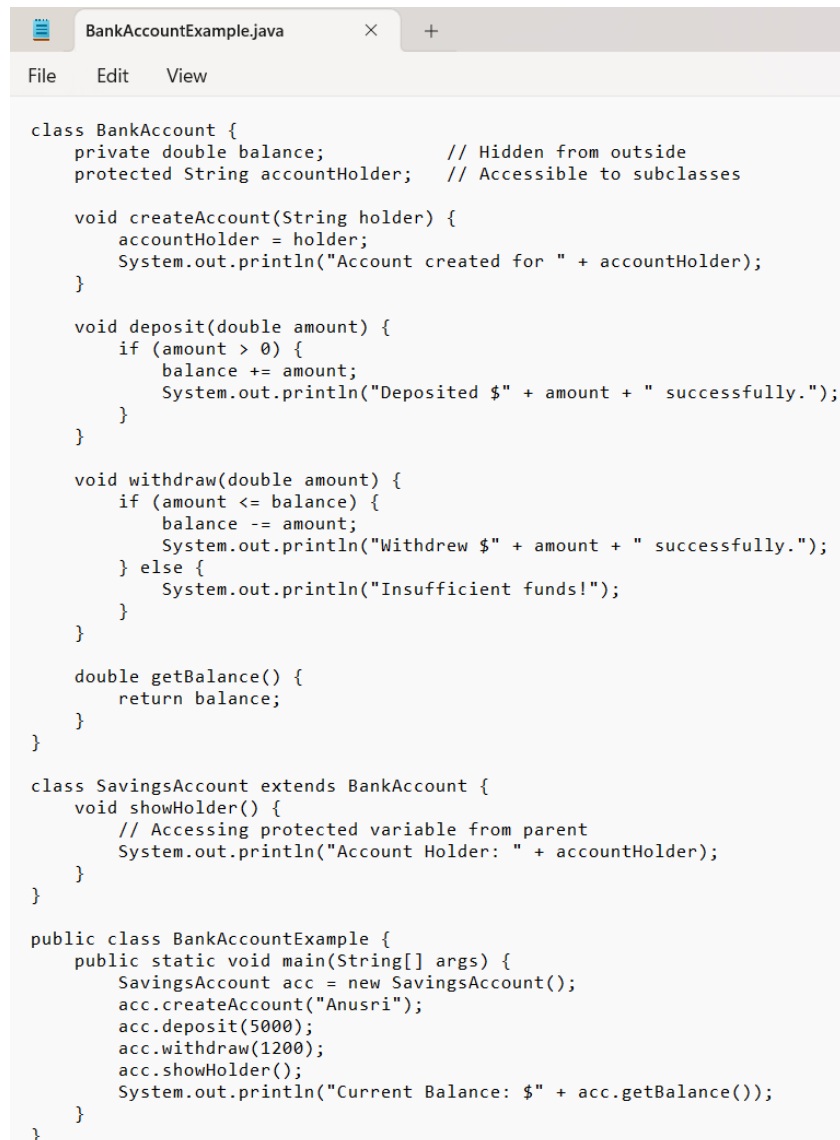
Aim

To implement and demonstrate the principles of encapsulation and abstraction by creating Java classes that use access modifiers to hide internal data and expose only essential operations through public methods.

Code

(a) Encapsulation

(i)



```
class BankAccount {
    private double balance;           // Hidden from outside
    protected String accountHolder;  // Accessible to subclasses

    void createAccount(String holder) {
        accountHolder = holder;
        System.out.println("Account created for " + accountHolder);
    }

    void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited $" + amount + " successfully.");
        }
    }

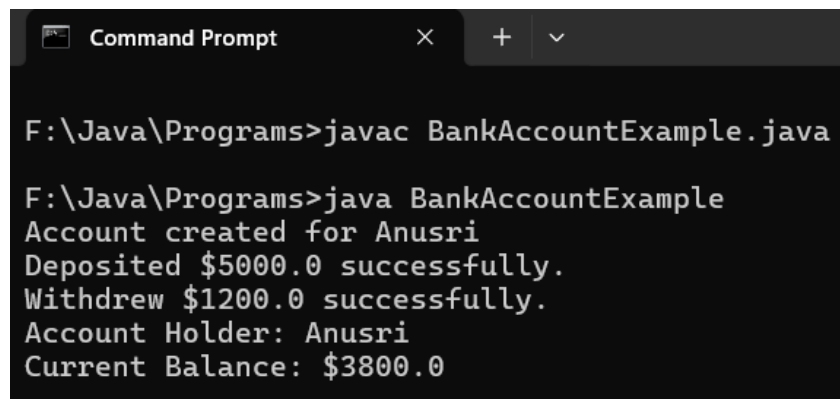
    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew $" + amount + " successfully.");
        } else {
            System.out.println("Insufficient funds!");
        }
    }

    double getBalance() {
        return balance;
    }
}

class SavingsAccount extends BankAccount {
    void showHolder() {
        // Accessing protected variable from parent
        System.out.println("Account Holder: " + accountHolder);
    }
}

public class BankAccountExample {
    public static void main(String[] args) {
        SavingsAccount acc = new SavingsAccount();
        acc.createAccount("Anusri");
        acc.deposit(5000);
        acc.withdraw(1200);
        acc.showHolder();
        System.out.println("Current Balance: $" + acc.getBalance());
    }
}
```

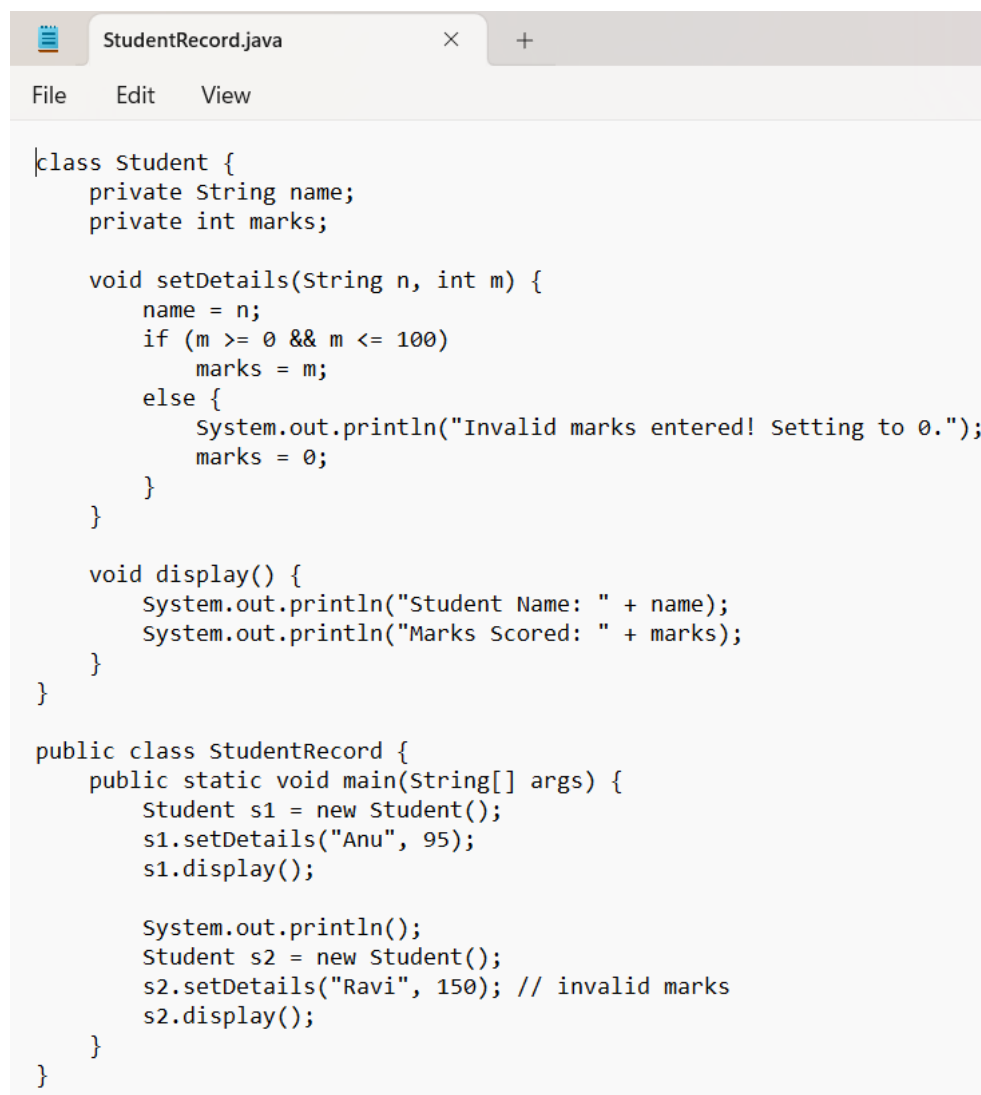

Output



```
F:\Java\Programs>javac BankAccountExample.java

F:\Java\Programs>java BankAccountExample
Account created for Anusri
Deposited $5000.0 successfully.
Withdrew $1200.0 successfully.
Account Holder: Anusri
Current Balance: $3800.0
```

(ii)



```
StudentRecord.java
File Edit View

class Student {
    private String name;
    private int marks;

    void setDetails(String n, int m) {
        name = n;
        if (m >= 0 && m <= 100)
            marks = m;
        else {
            System.out.println("Invalid marks entered! Setting to 0.");
            marks = 0;
        }
    }

    void display() {
        System.out.println("Student Name: " + name);
        System.out.println("Marks Scored: " + marks);
    }
}

public class StudentRecord {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setDetails("Anu", 95);
        s1.display();

        System.out.println();
        Student s2 = new Student();
        s2.setDetails("Ravi", 150); // invalid marks
        s2.display();
    }
}
```

Output

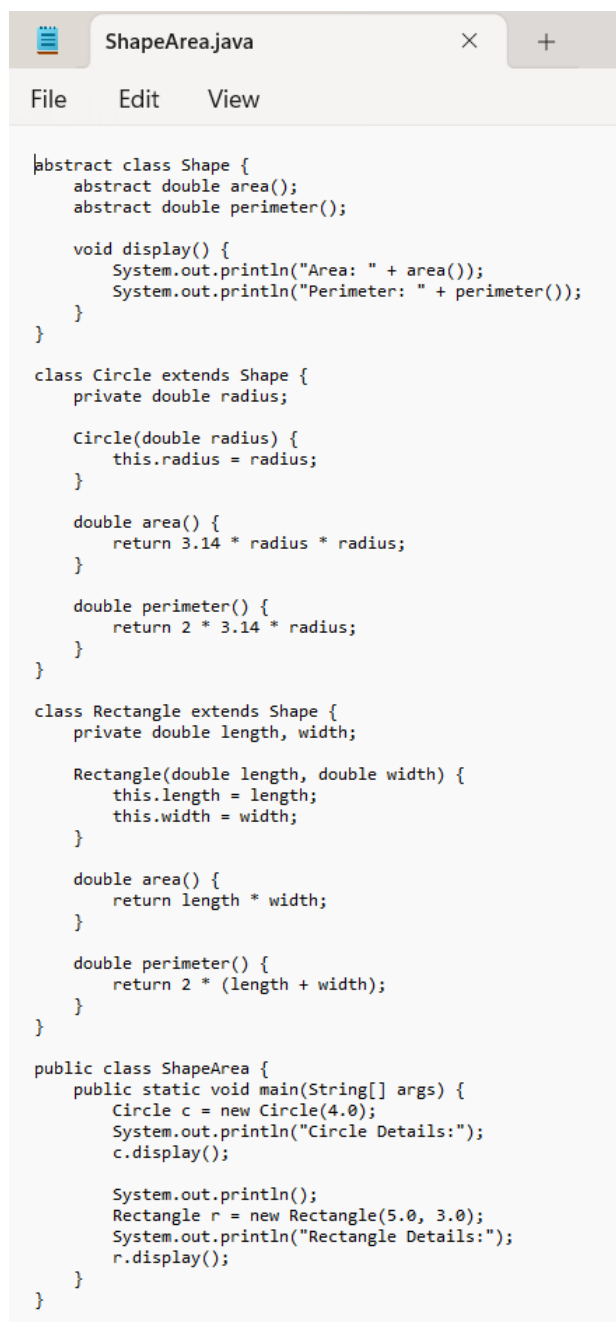
```
F:\Java\Programs>javac StudentRecord.java

F:\Java\Programs>java StudentRecord
Student Name: Anu
Marks Scored: 95

Invalid marks entered! Setting to 0.
Student Name: Ravi
Marks Scored: 0
```

(b) Abstraction

(i)



```
ShapeArea.java
File Edit View

abstract class Shape {
    abstract double area();
    abstract double perimeter();

    void display() {
        System.out.println("Area: " + area());
        System.out.println("Perimeter: " + perimeter());
    }
}

class Circle extends Shape {
    private double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    double area() {
        return 3.14 * radius * radius;
    }

    double perimeter() {
        return 2 * 3.14 * radius;
    }
}

class Rectangle extends Shape {
    private double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double area() {
        return length * width;
    }

    double perimeter() {
        return 2 * (length + width);
    }
}

public class ShapeArea {
    public static void main(String[] args) {
        Circle c = new Circle(4.0);
        System.out.println("Circle Details:");
        c.display();

        System.out.println();
        Rectangle r = new Rectangle(5.0, 3.0);
        System.out.println("Rectangle Details:");
        r.display();
    }
}
```

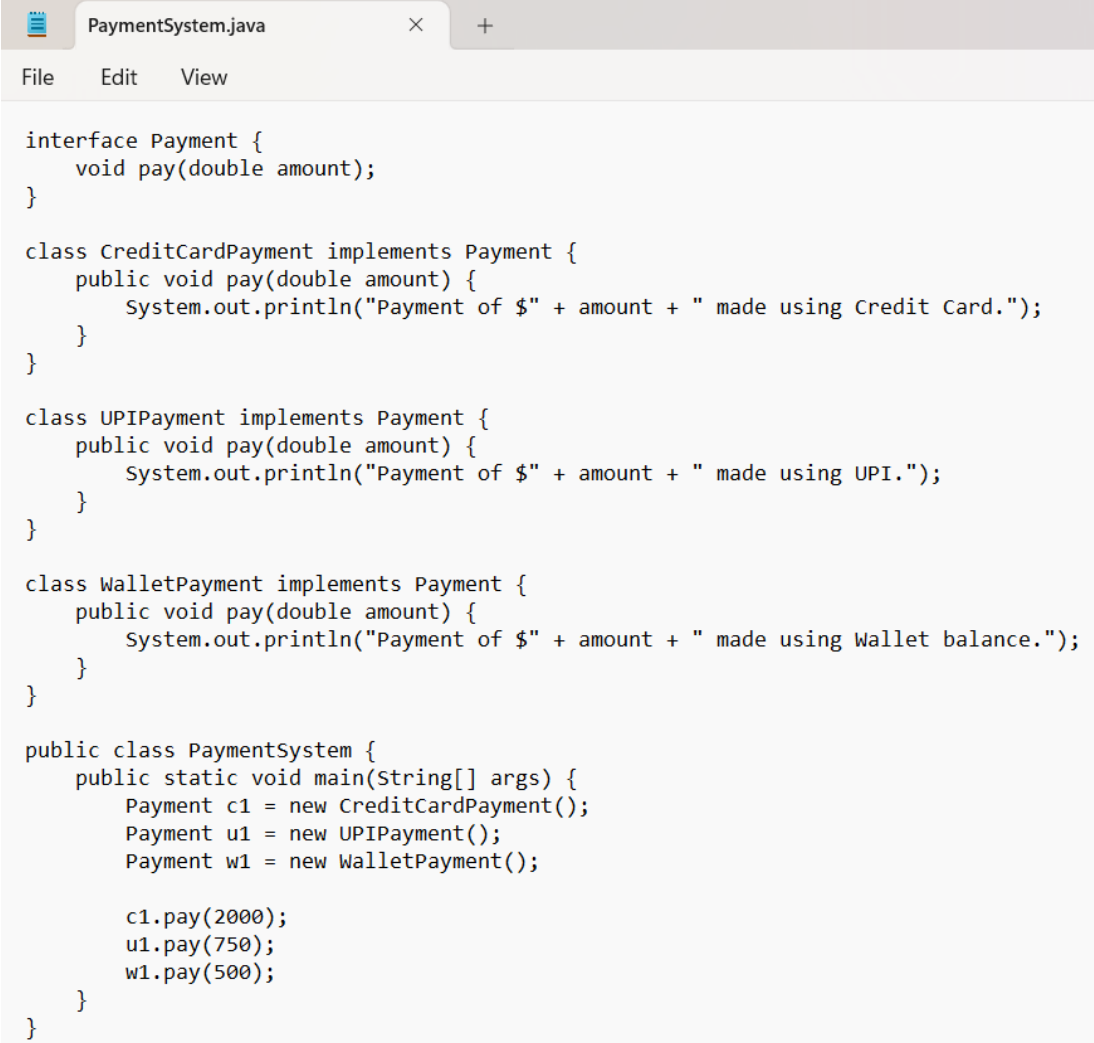
Output

```
F:\Java\Programs>javac ShapeArea.java

F:\Java\Programs>java ShapeArea
Circle Details:
Area: 50.24
Perimeter: 25.12

Rectangle Details:
Area: 15.0
Perimeter: 16.0
```

(ii)



```
PaymentSystem.java
File Edit View

interface Payment {
    void pay(double amount);
}

class CreditCardPayment implements Payment {
    public void pay(double amount) {
        System.out.println("Payment of $" + amount + " made using Credit Card.");
    }
}

class UPIPayment implements Payment {
    public void pay(double amount) {
        System.out.println("Payment of $" + amount + " made using UPI.");
    }
}

class WalletPayment implements Payment {
    public void pay(double amount) {
        System.out.println("Payment of $" + amount + " made using Wallet balance.");
    }
}

public class PaymentSystem {
    public static void main(String[] args) {
        Payment c1 = new CreditCardPayment();
        Payment u1 = new UPIPayment();
        Payment w1 = new WalletPayment();

        c1.pay(2000);
        u1.pay(750);
        w1.pay(500);
    }
}
```

Output

```
F:\Java\Programs>javac PaymentSystem.java

F:\Java\Programs>java PaymentSystem
Payment of $2000.0 made using Credit Card.
Payment of $750.0 made using UPI.
Payment of $500.0 made using Wallet balance.
```

Conclusion

BankAccountExample demonstrated encapsulation using private data and protected access. ShapeArea implemented abstraction through abstract classes and methods. PaymentSystem achieved full abstraction using interfaces and implementation classes. StudentRecord showed data protection with private variables and validation. Together, these programs successfully illustrated how encapsulation hides internal state while abstraction exposes essential features through simplified interfaces.