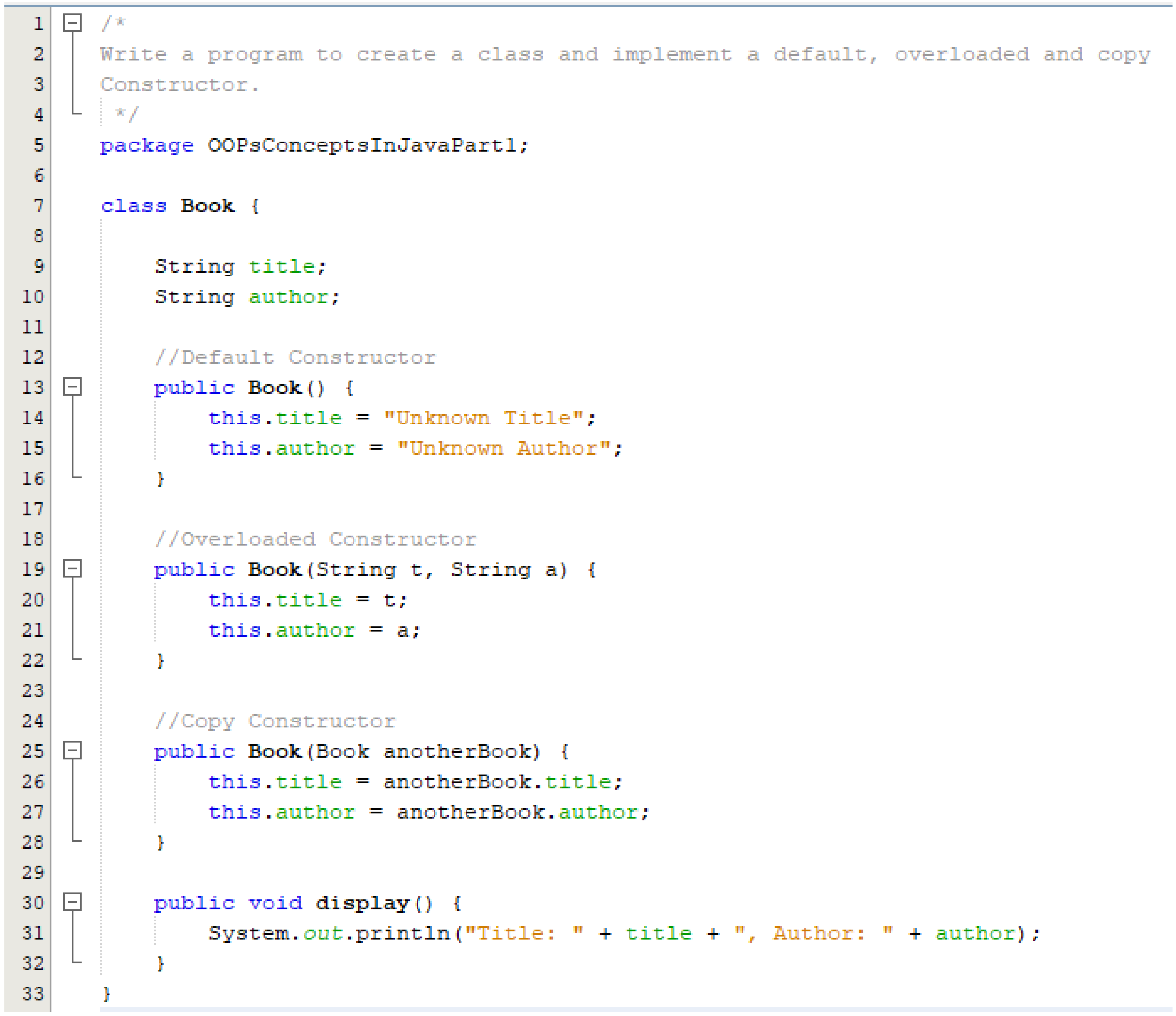
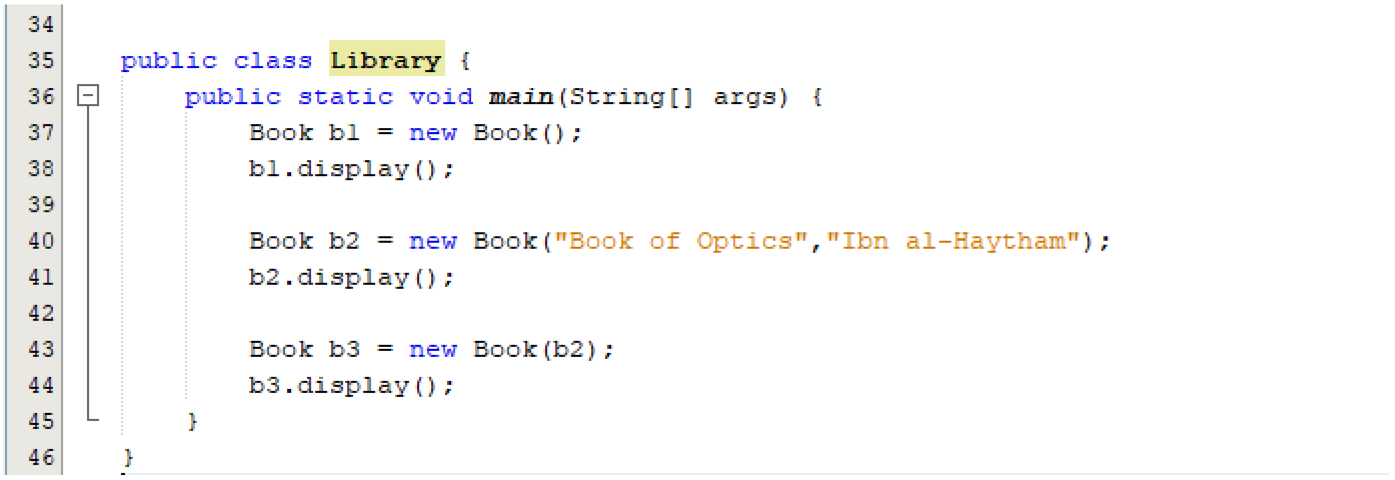
Java Programming Practical

1.a] Write a program to create a class and implement a default, overloaded and copy Constructor.

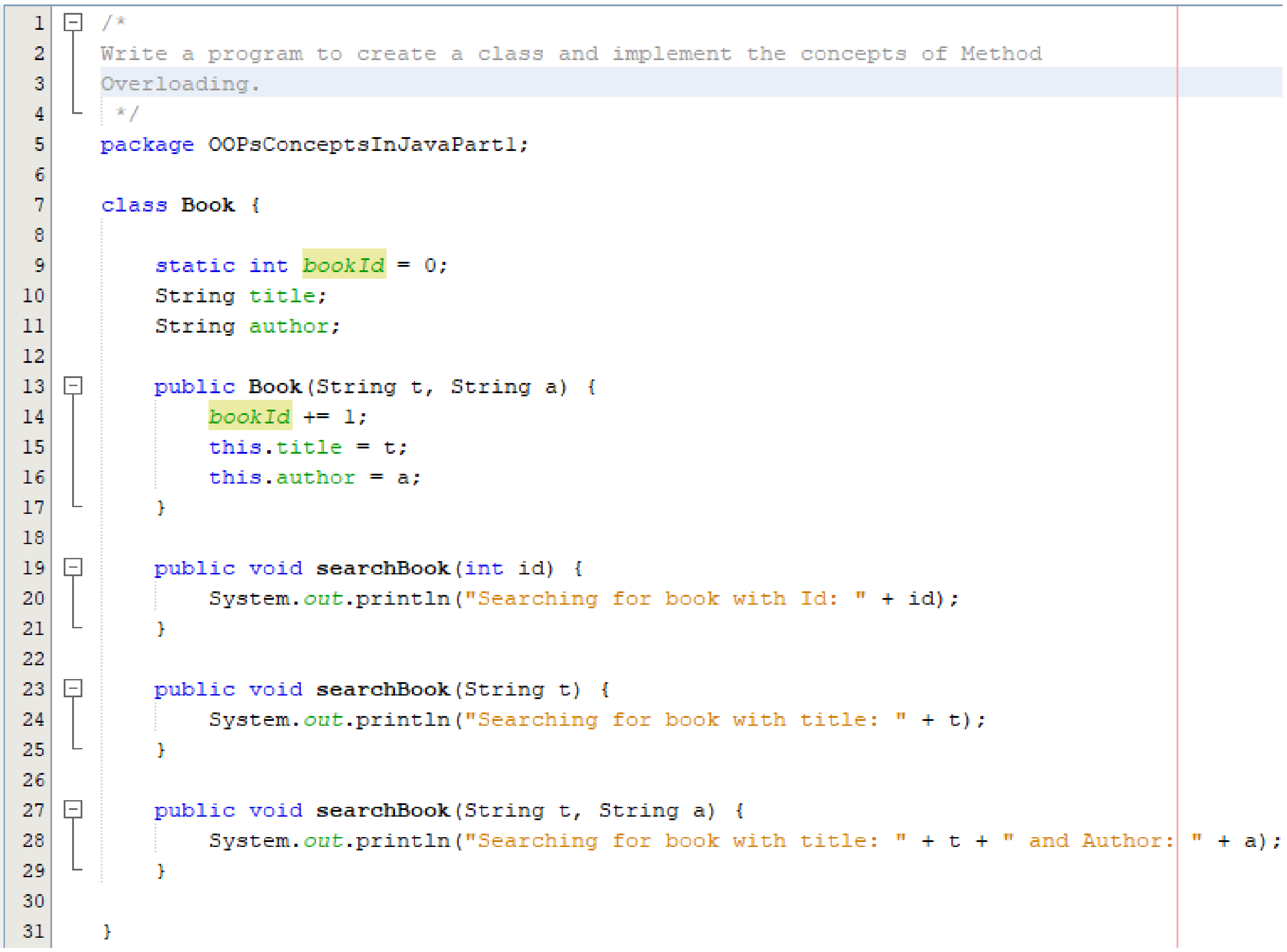


```java
/*
Write a program to create a class and implement a default, overloaded and copy
Constructor.
 */
package OOPsConceptsInJavaPart1;

class Book {

    String title;
    String author;

    //Default Constructor
    public Book() {
        this.title = "Unknown Title";
        this.author = "Unknown Author";
    }

    //Overloaded Constructor
    public Book(String t, String a) {
        this.title = t;
        this.author = a;
    }

    //Copy Constructor
    public Book(Book anotherBook) {
        this.title = anotherBook.title;
        this.author = anotherBook.author;
    }

    public void display() {
        System.out.println("Title: " + title + ", Author: " + author);
    }
}
```

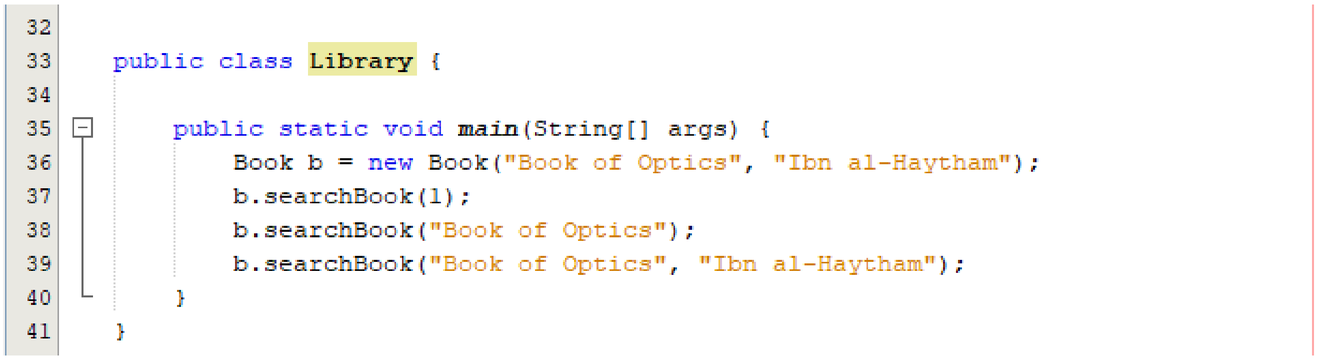


```java

public class Library {
    public static void main(String[] args) {
        Book b1 = new Book();
        b1.display();

        Book b2 = new Book("Book of Optics","Ibn al-Haytham");
        b2.display();

        Book b3 = new Book(b2);
        b3.display();
    }
}
```

1.b] Write a program to create a class and implement the concepts of Method Overloading

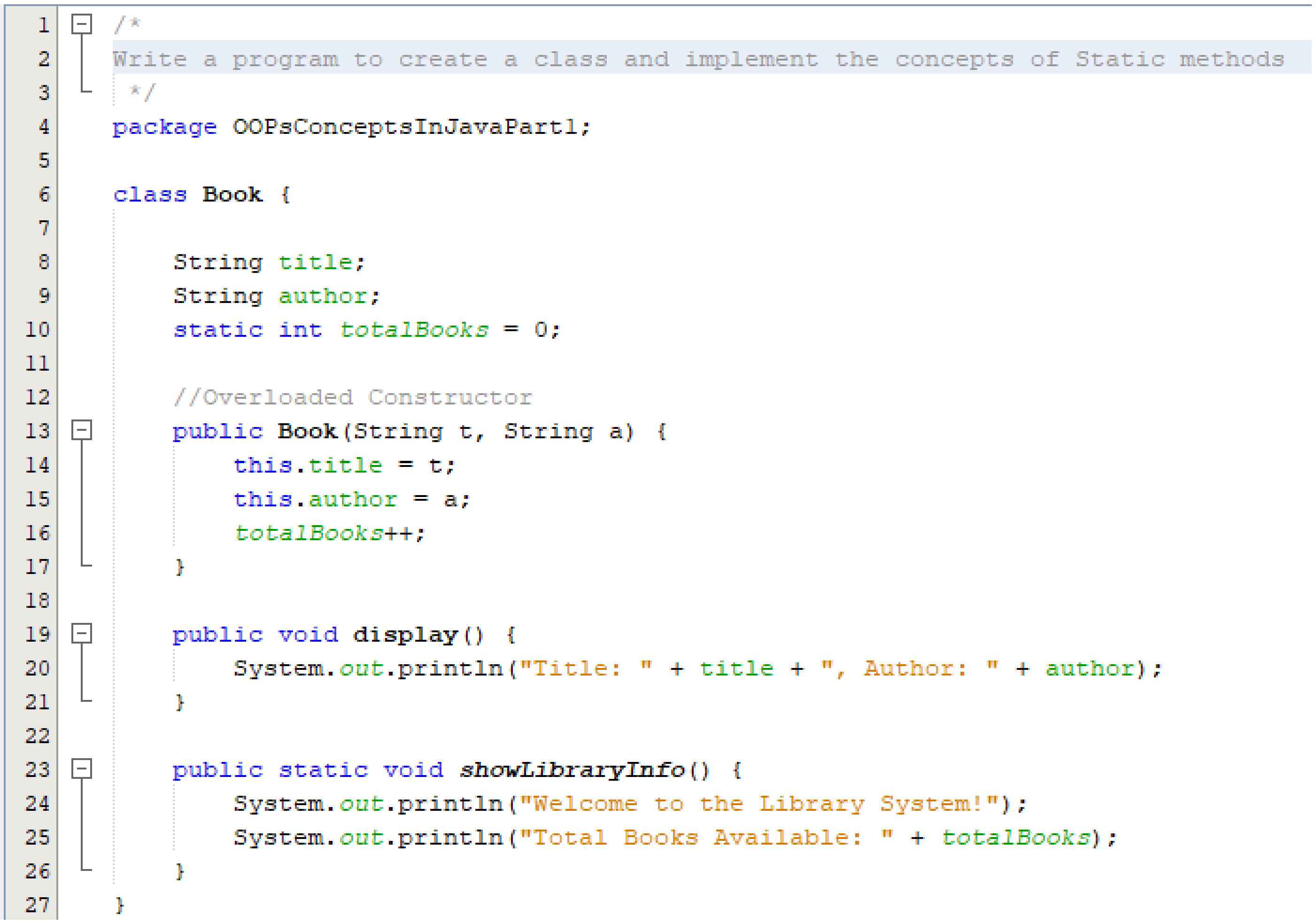


```java
/*
Write a program to create a class and implement the concepts of Method
Overloading.
 */
package OOPsConceptsInJavaPart1;

class Book {

    static int bookId = 0;
    String title;
    String author;

    public Book(String t, String a) {
        bookId += 1;
        this.title = t;
        this.author = a;
    }

    public void searchBook(int id) {
        System.out.println("Searching for book with Id: " + id);
    }

    public void searchBook(String t) {
        System.out.println("Searching for book with title: " + t);
    }

    public void searchBook(String t, String a) {
        System.out.println("Searching for book with title: " + t + " and Author: " + a);
    }

}
```

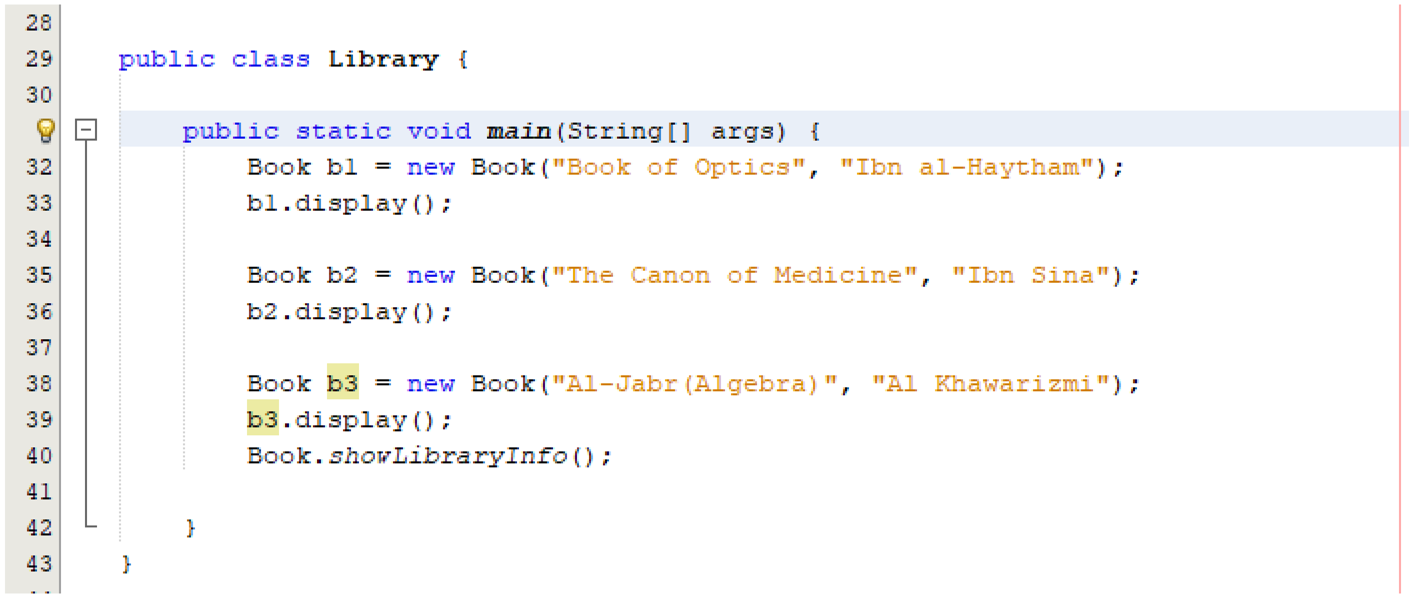


```java

public class Library {

    public static void main(String[] args) {
        Book b = new Book("Book of Optics", "Ibn al-Haytham");
        b.searchBook(1);
        b.searchBook("Book of Optics");
        b.searchBook("Book of Optics", "Ibn al-Haytham");
    }
}
```

1.c] Write a program to create a class and implement the concepts of Static methods.

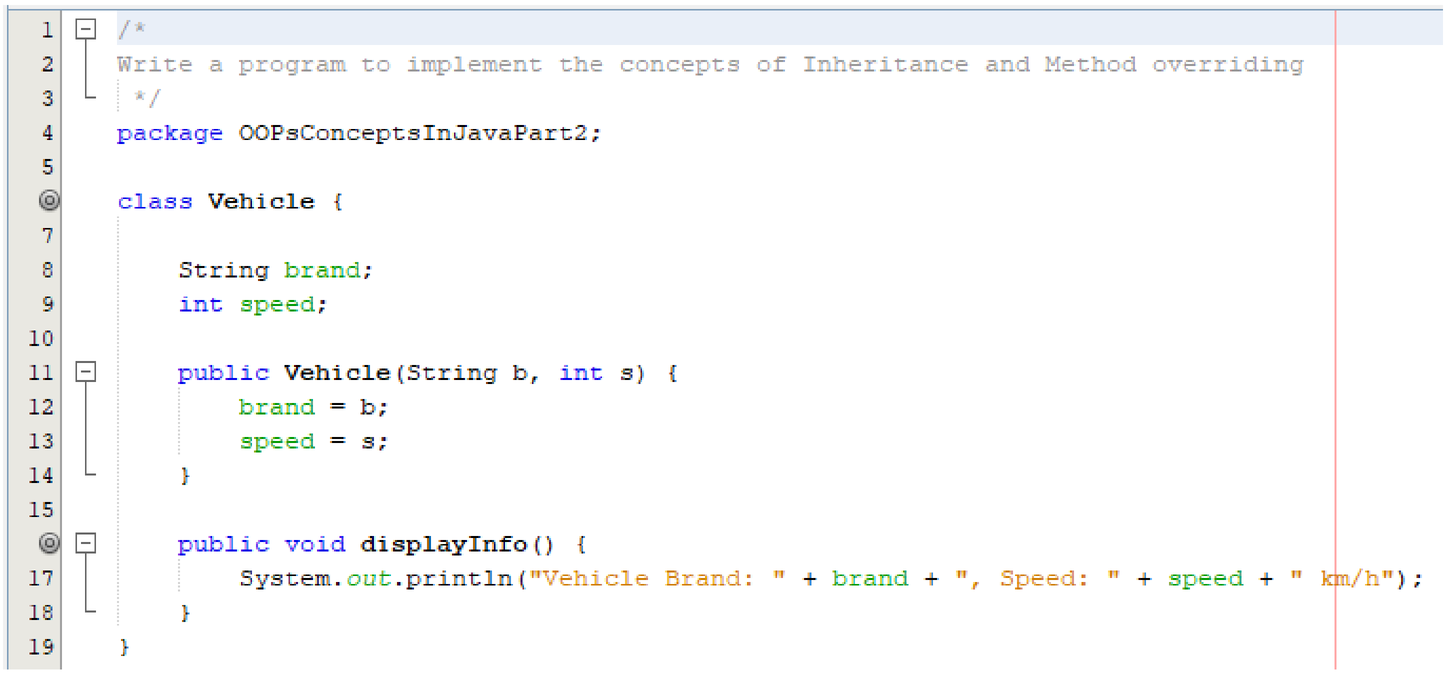


```java
/*
Write a program to create a class and implement the concepts of Static methods
 */
package OOPsConceptsInJavaPart1;

class Book {

    String title;
    String author;
    static int totalBooks = 0;

    //Overloaded Constructor
    public Book(String t, String a) {
        this.title = t;
        this.author = a;
        totalBooks++;
    }

    public void display() {
        System.out.println("Title: " + title + ", Author: " + author);
    }

    public static void showLibraryInfo() {
        System.out.println("Welcome to the Library System!");
        System.out.println("Total Books Available: " + totalBooks);
    }
}
```

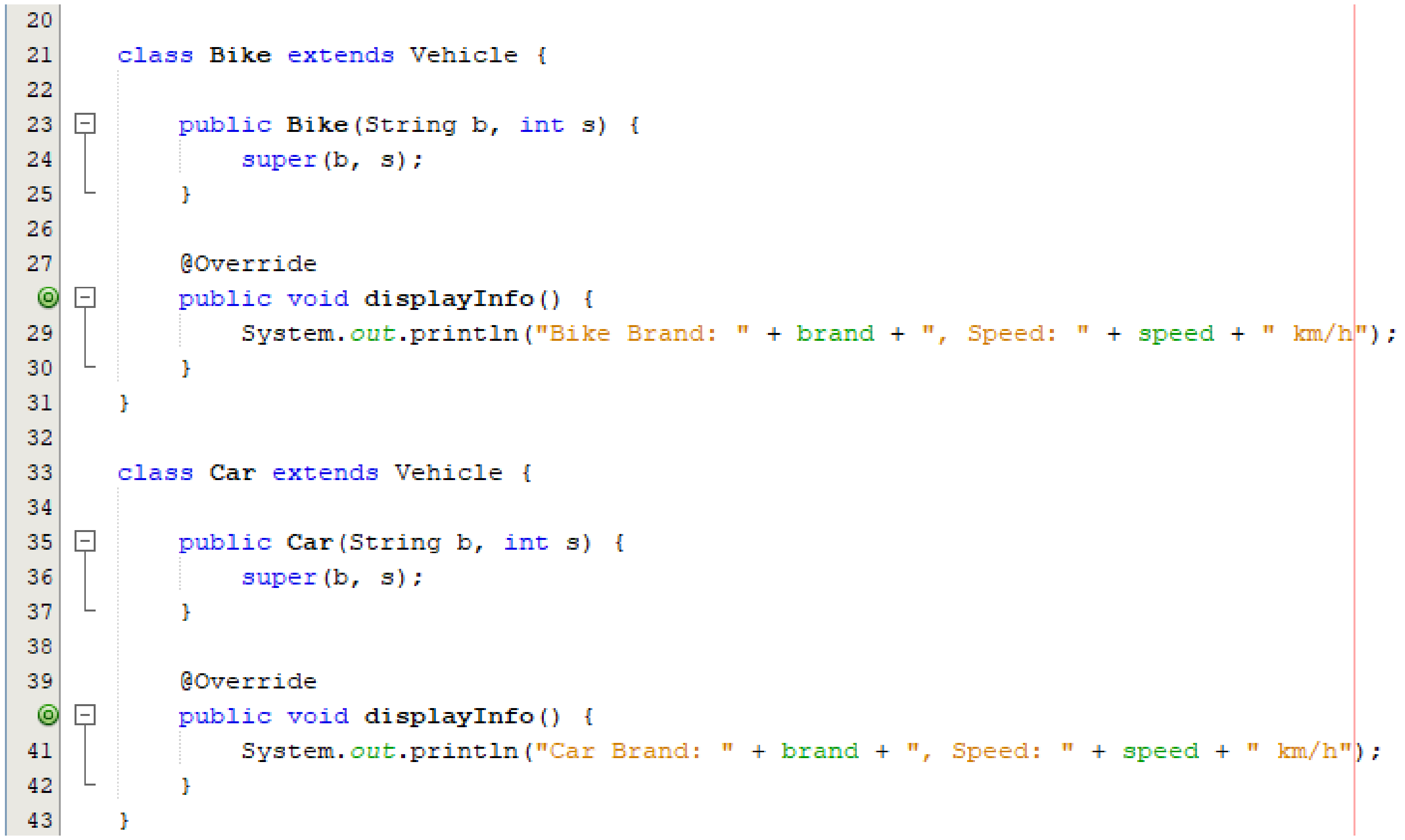


```java

public class Library {

    public static void main(String[] args) {
        Book b1 = new Book("Book of Optics", "Ibn al-Haytham");
        b1.display();

        Book b2 = new Book("The Canon of Medicine", "Ibn Sina");
        b2.display();

        Book b3 = new Book("Al-Jabr(Algebra)", "Al Khawarizmi");
        b3.display();
        Book.showLibraryInfo();

    }
}
```

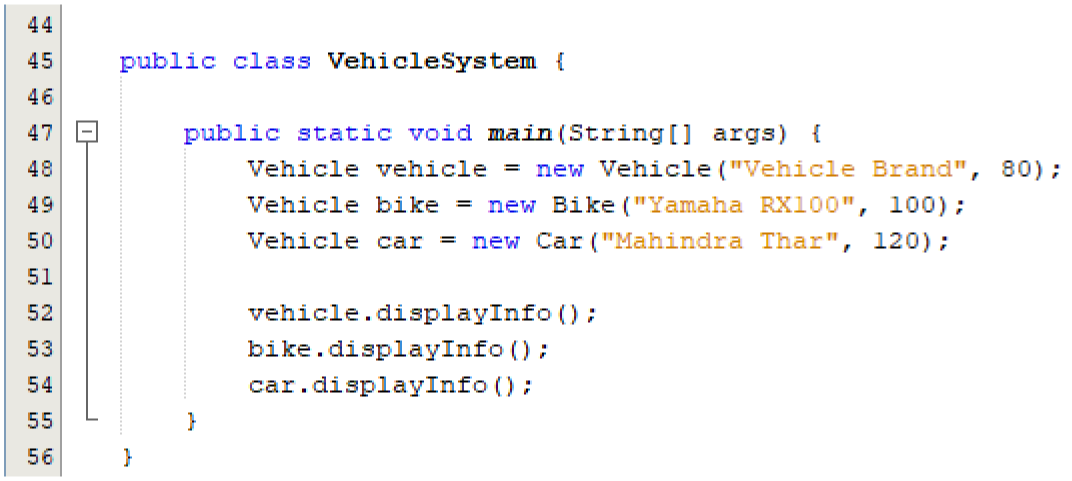
2.a] Write a program to implement the concepts of Inheritance and Method overriding.



```java
/*
Write a program to implement the concepts of Inheritance and Method overriding
 */
package OOPsConceptsInJavaPart2;

class Vehicle {

    String brand;
    int speed;

    public Vehicle(String b, int s) {
        brand = b;
        speed = s;
    }

    public void displayInfo() {
        System.out.println("Vehicle Brand: " + brand + ", Speed: " + speed + " km/h");
    }
}
```

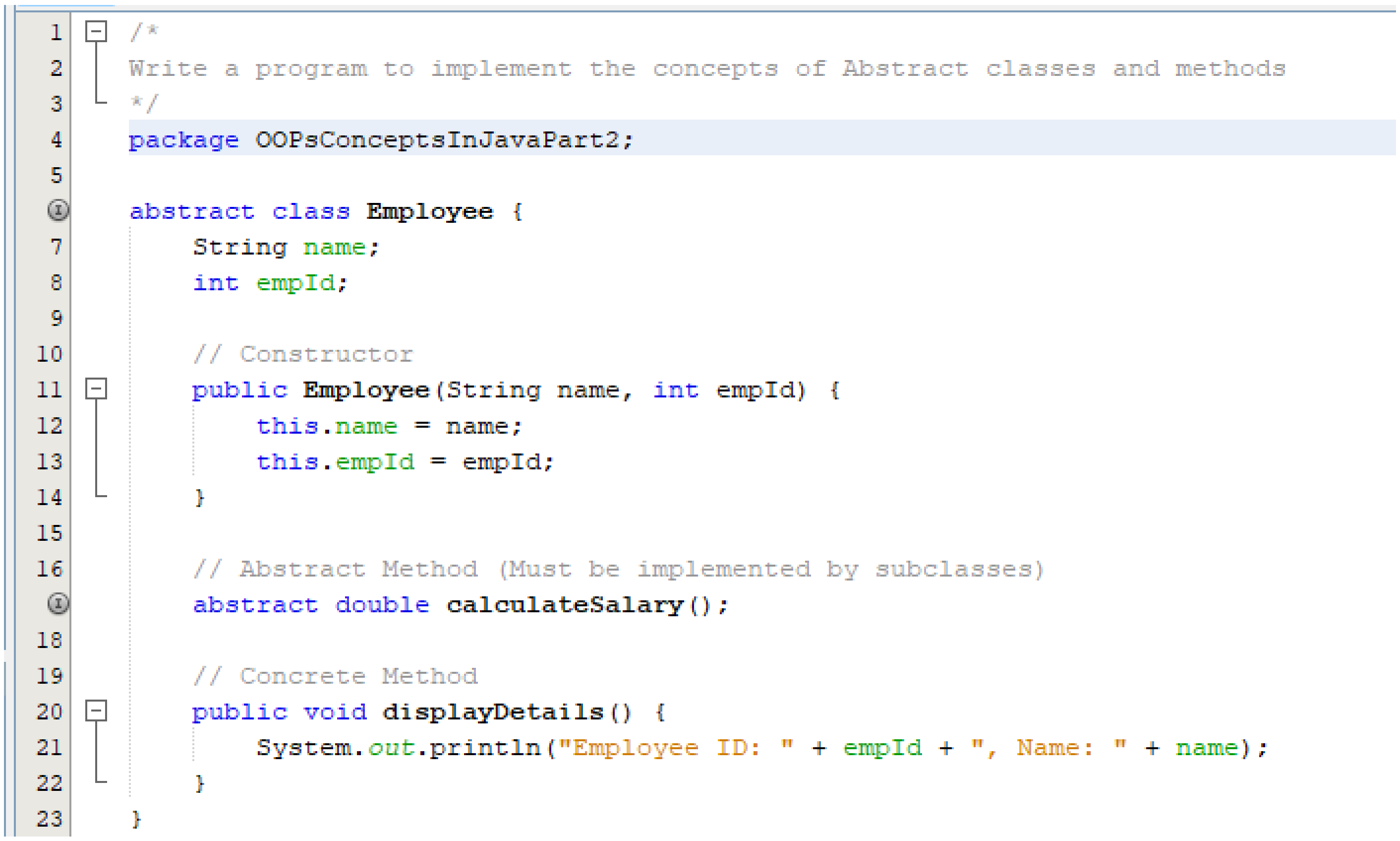


```java

class Bike extends Vehicle {

    public Bike(String b, int s) {
        super(b, s);
    }

    @Override
    public void displayInfo() {
        System.out.println("Bike Brand: " + brand + ", Speed: " + speed + " km/h");
    }
}

class Car extends Vehicle {

    public Car(String b, int s) {
        super(b, s);
    }

    @Override
    public void displayInfo() {
        System.out.println("Car Brand: " + brand + ", Speed: " + speed + " km/h");
    }
}
```

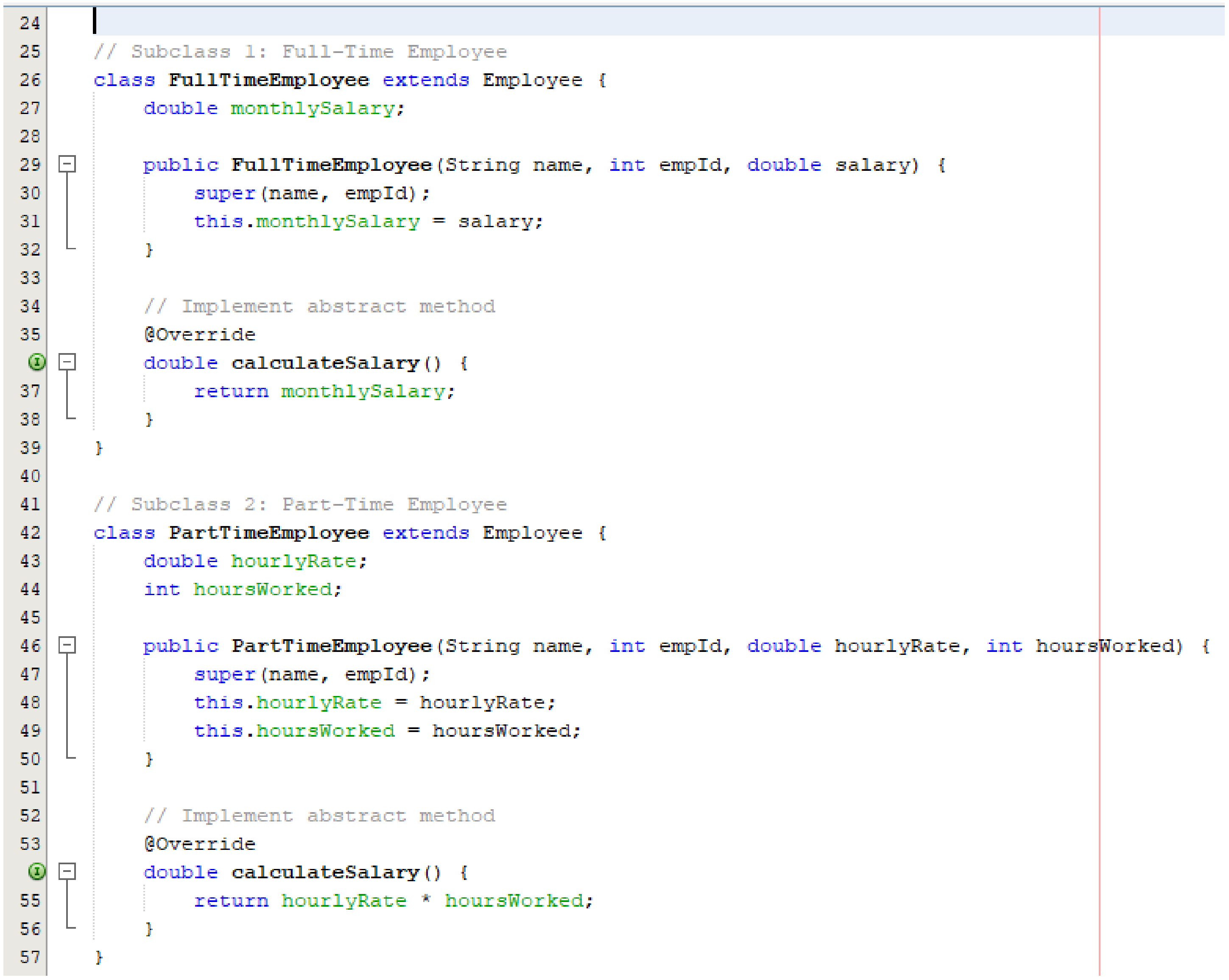


```java

public class VehicleSystem {

    public static void main(String[] args) {
        Vehicle vehicle = new Vehicle("Vehicle Brand", 80);
        Vehicle bike = new Bike("Yamaha RX100", 100);
        Vehicle car = new Car("Mahindra Thar", 120);

        vehicle.displayInfo();
        bike.displayInfo();
        car.displayInfo();
    }
}
```

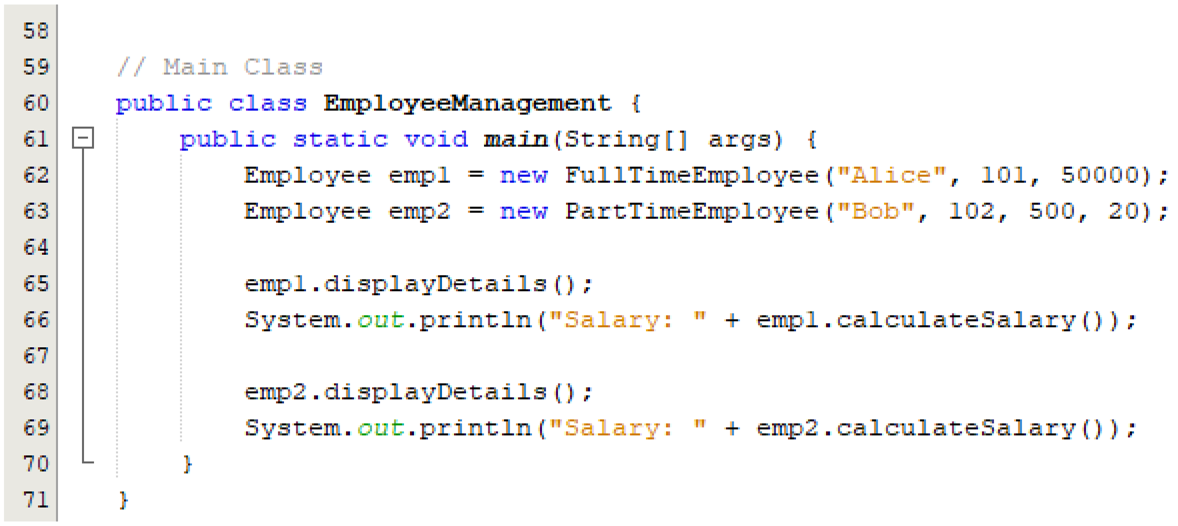
2.b] Write a program to implement the concepts of Abstract classes and methods.



```java
/*
Write a program to implement the concepts of Abstract classes and methods
*/
package OOPsConceptsInJavaPart2;

abstract class Employee {
    String name;
    int empId;

    // Constructor
    public Employee(String name, int empId) {
        this.name = name;
        this.empId = empId;
    }

    // Abstract Method (Must be implemented by subclasses)
    abstract double calculateSalary();

    // Concrete Method
    public void displayDetails() {
        System.out.println("Employee ID: " + empId + ", Name: " + name);
    }
}
```

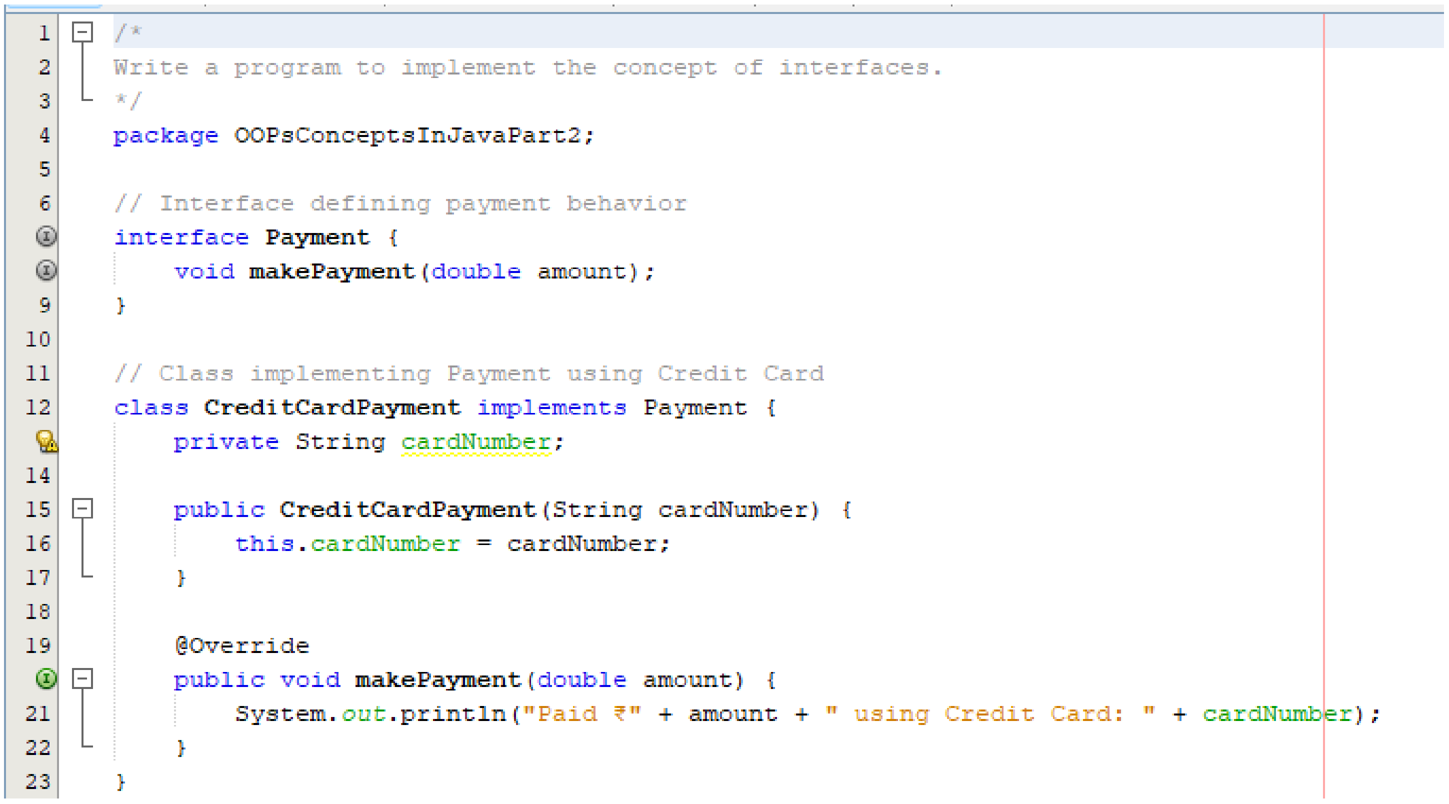


```java

// Subclass 1: Full-Time Employee
class FullTimeEmployee extends Employee {
    double monthlySalary;

    public FullTimeEmployee(String name, int empId, double salary) {
        super(name, empId);
        this.monthlySalary = salary;
    }

    // Implement abstract method
    @Override
    double calculateSalary() {
        return monthlySalary;
    }
}

// Subclass 2: Part-Time Employee
class PartTimeEmployee extends Employee {
    double hourlyRate;
    int hoursWorked;

    public PartTimeEmployee(String name, int empId, double hourlyRate, int hoursWorked) {
        super(name, empId);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    // Implement abstract method
    @Override
    double calculateSalary() {
        return hourlyRate * hoursWorked;
    }
}
```

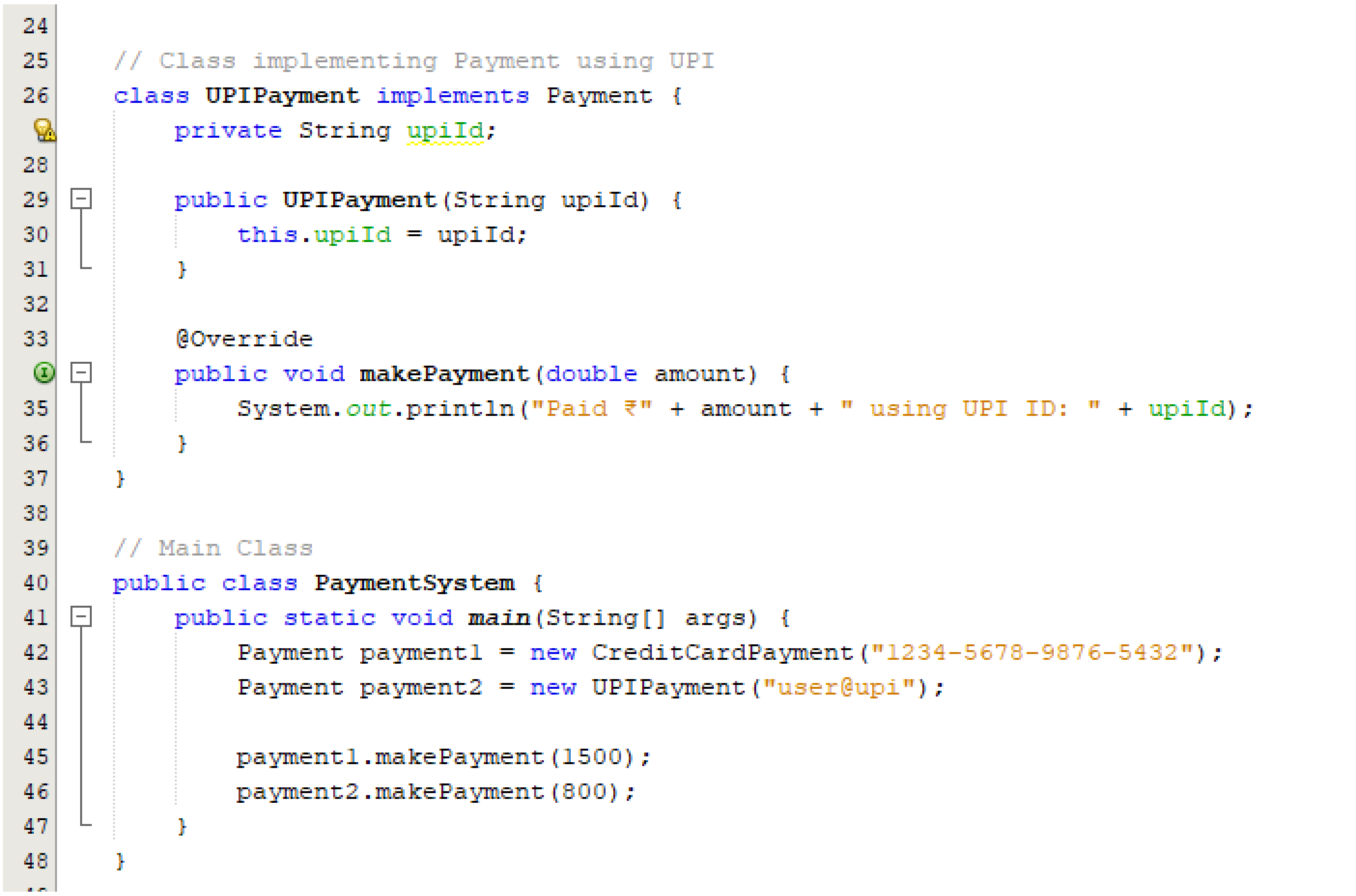


```java

// Main Class
public class EmployeeManagement {
    public static void main(String[] args) {
        Employee emp1 = new FullTimeEmployee("Alice", 101, 50000);
        Employee emp2 = new PartTimeEmployee("Bob", 102, 500, 20);

        emp1.displayDetails();
        System.out.println("Salary: " + emp1.calculateSalary());

        emp2.displayDetails();
        System.out.println("Salary: " + emp2.calculateSalary());
    }
}
```

2.c] Write a program to implement the concept of interfaces.

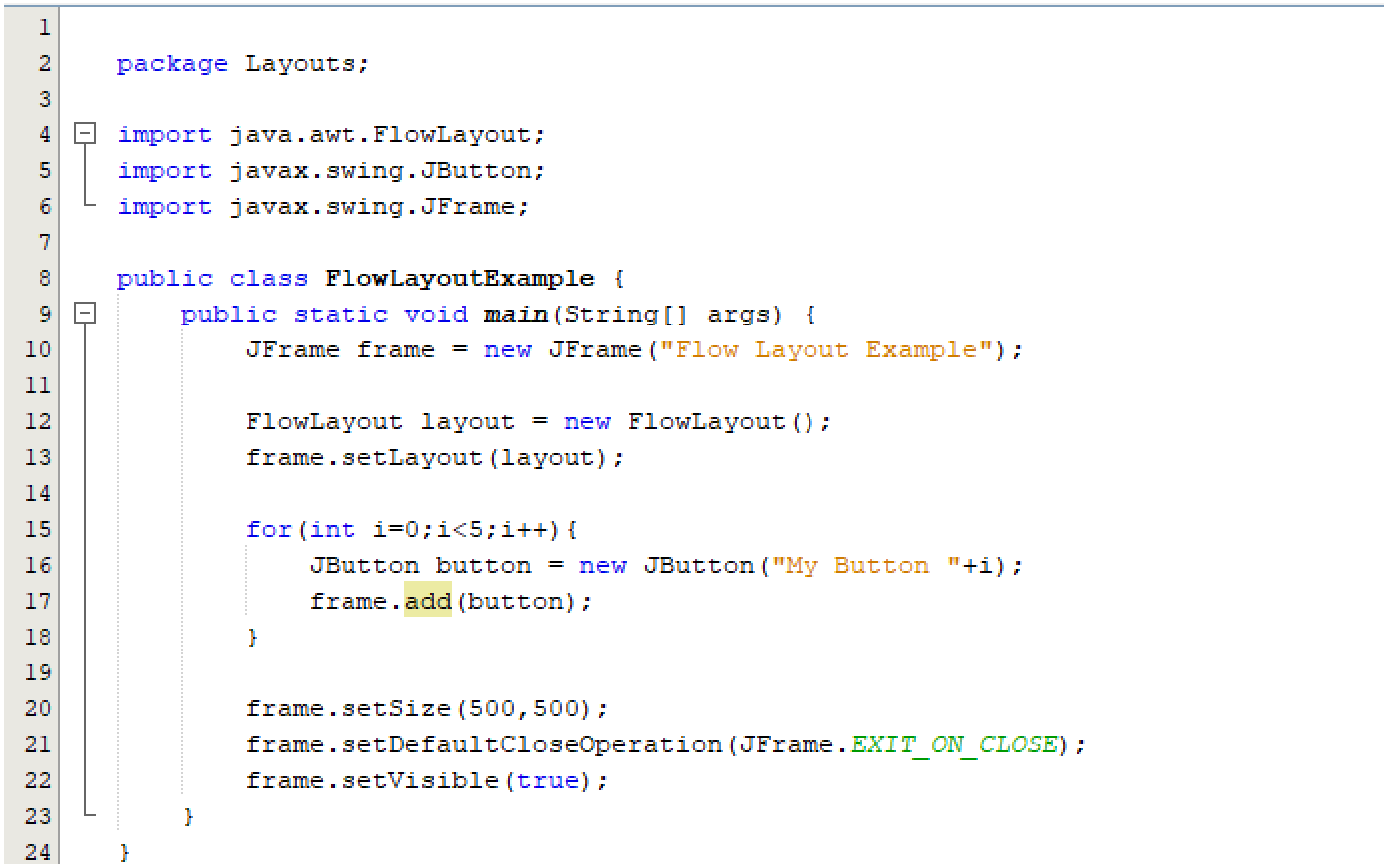


```java
/*
Write a program to implement the concept of interfaces.
*/
package OOPsConceptsInJavaPart2;

// Interface defining payment behavior
interface Payment {
    void makePayment(double amount);
}

// Class implementing Payment using Credit Card
class CreditCardPayment implements Payment {
    private String cardNumber;

    public CreditCardPayment(String cardNumber) {
        this.cardNumber = cardNumber;
    }

    @Override
    public void makePayment(double amount) {
        System.out.println("Paid ₹" + amount + " using Credit Card: " + cardNumber);
    }
}
```



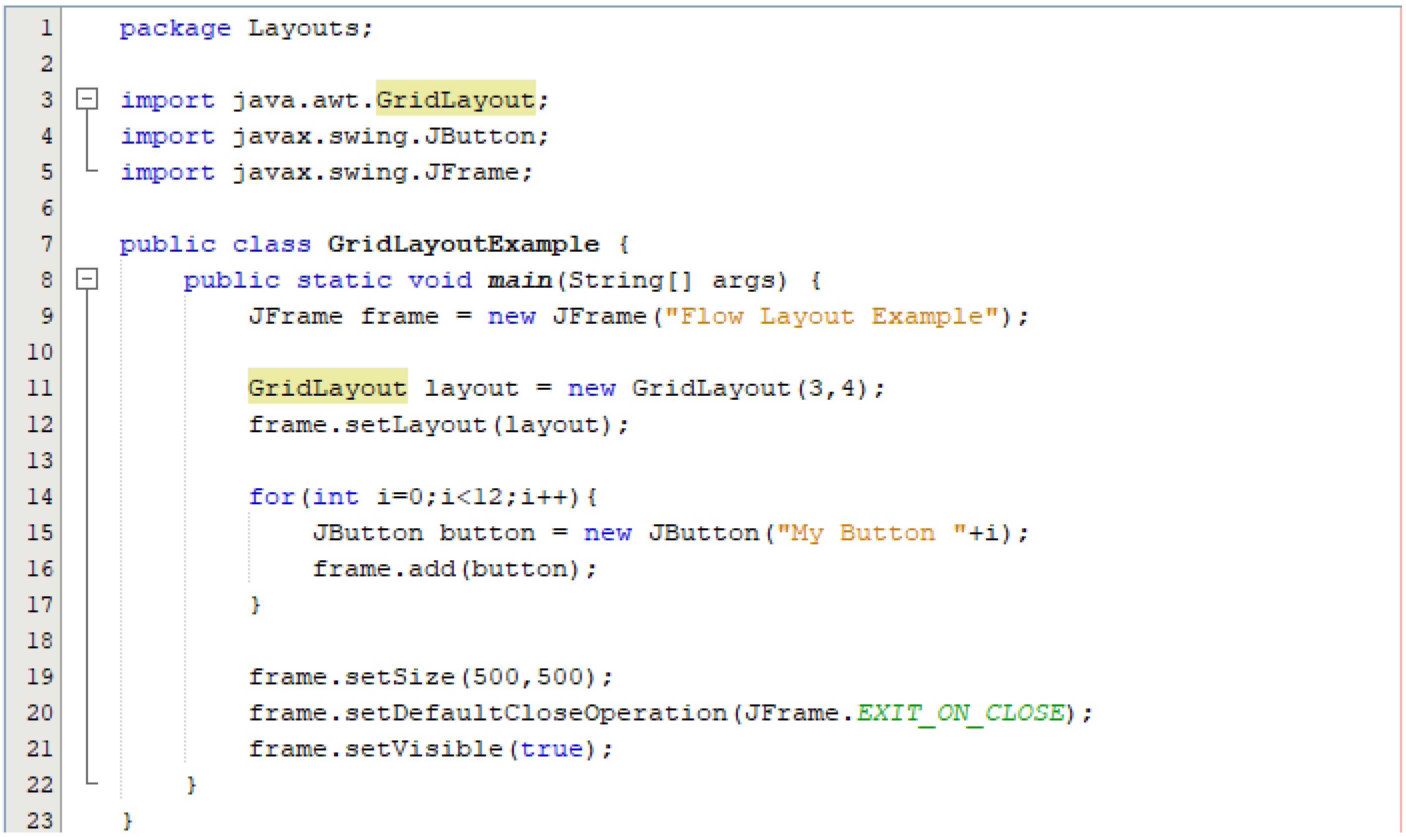
```java

// Class implementing Payment using UPI
class UPIPayment implements Payment {
    private String upiId;

    public UPIPayment(String upiId) {
        this.upiId = upiId;
    }

    @Override
    public void makePayment(double amount) {
        System.out.println("Paid ₹" + amount + " using UPI ID: " + upiId);
    }
}

// Main Class
public class PaymentSystem {
    public static void main(String[] args) {
        Payment payment1 = new CreditCardPayment("1234-5678-9876-5432");
        Payment payment2 = new UPIPayment("user@upi");

        payment1.makePayment(1500);
        payment2.makePayment(800);
    }
}
```

7.a] F low Layout.



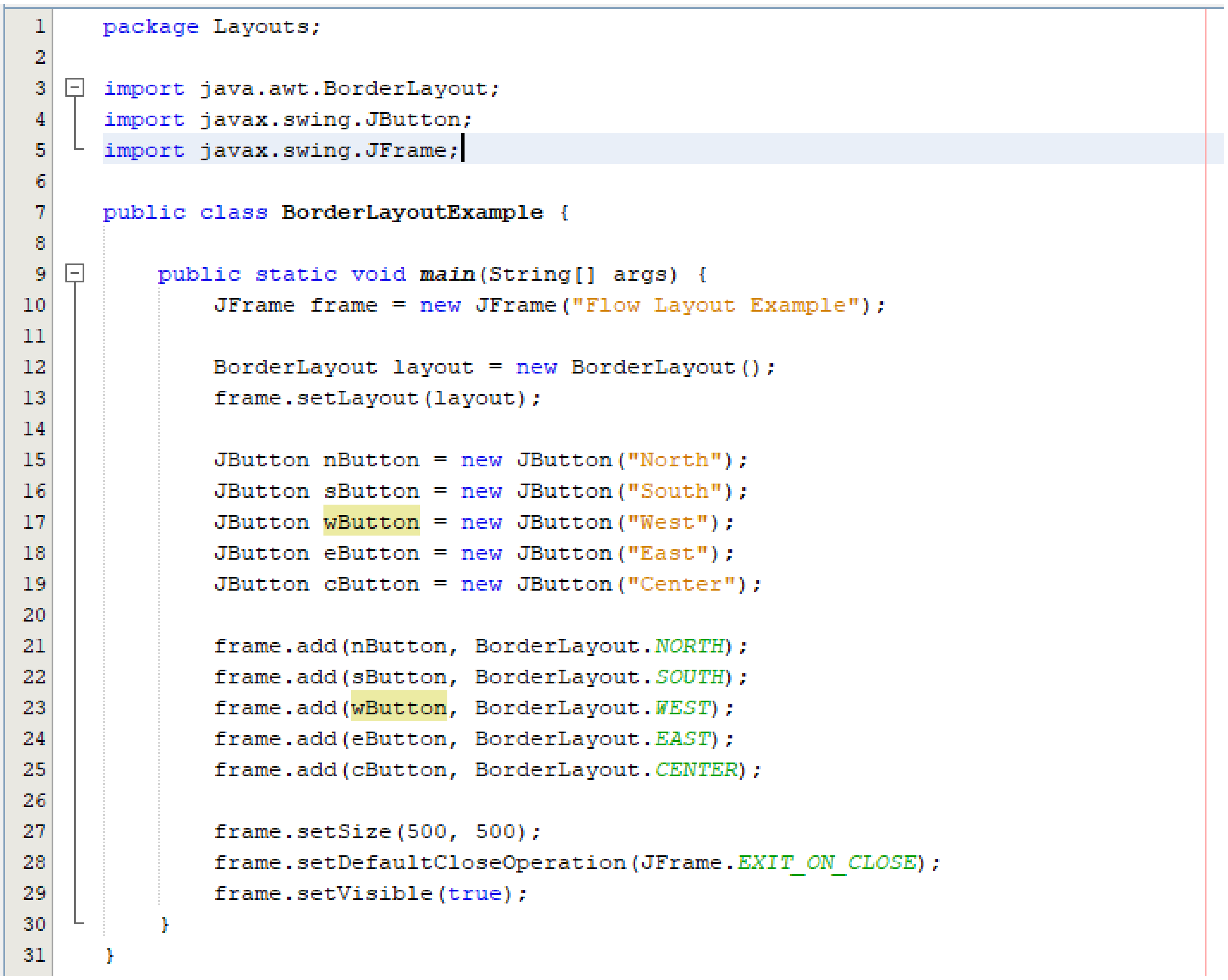
```java

package Layouts;

import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class FlowLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Flow Layout Example");

        FlowLayout layout = new FlowLayout();
        frame.setLayout(layout);

        for(int i=0;i<5;i++){
            JButton button = new JButton("My Button "+i);
            frame.add(button);
        }

        frame.setSize(500,500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

7.b] Grid Layout



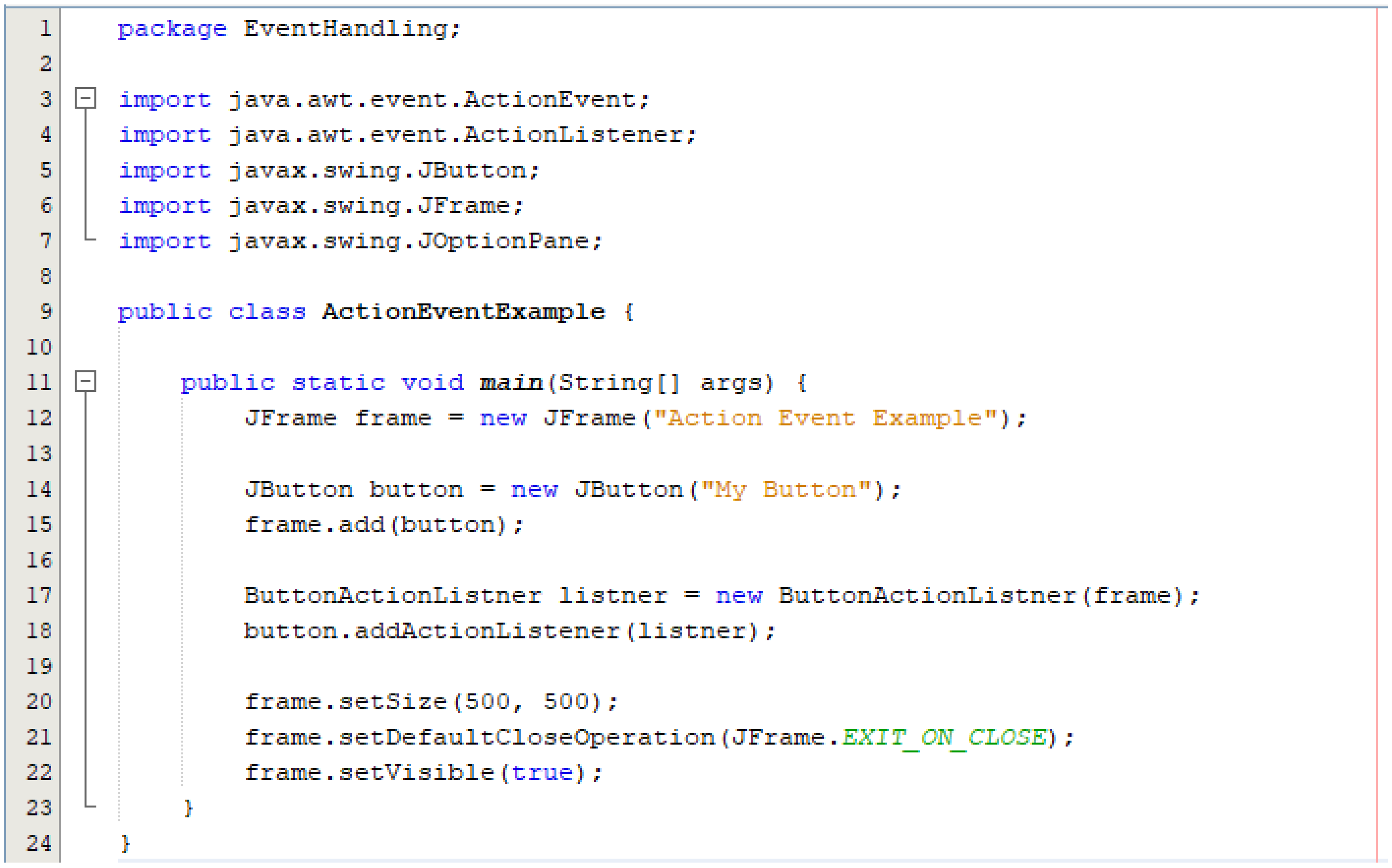
```java
package Layouts;

import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class GridLayoutExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Flow Layout Example");

        GridLayout layout = new GridLayout(3,4);
        frame.setLayout(layout);

        for(int i=0;i<12;i++){
            JButton button = new JButton("My Button "+i);
            frame.add(button);
        }

        frame.setSize(500,500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

7.c] Border Layout

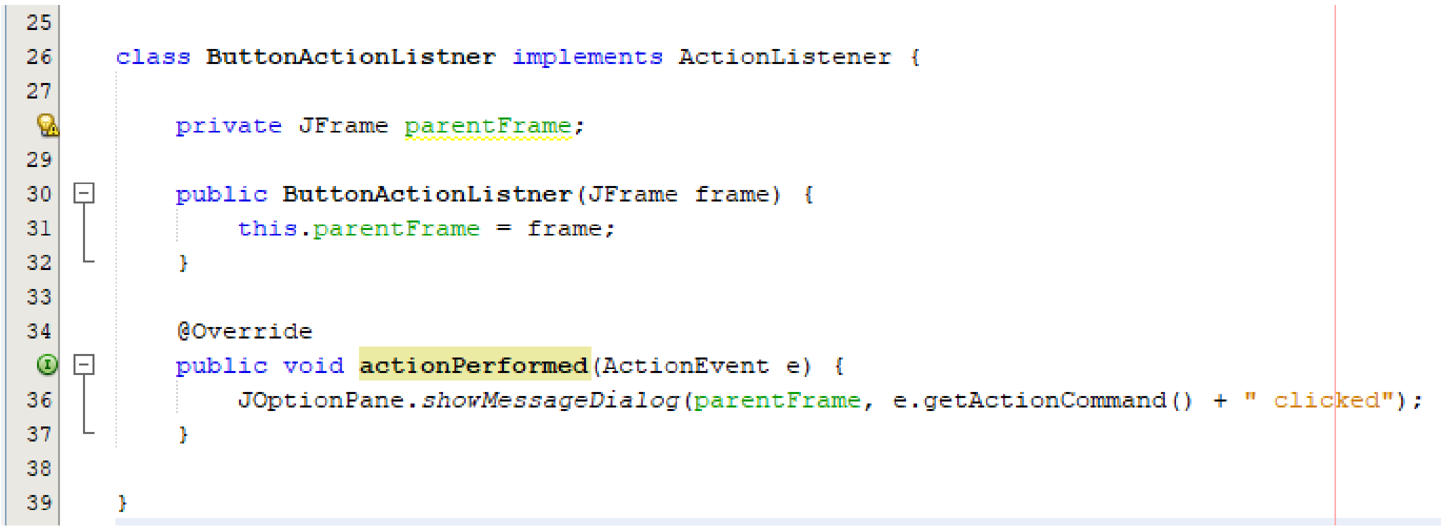


```java
package Layouts;

import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class BorderLayoutExample {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Flow Layout Example");

        BorderLayout layout = new BorderLayout();
        frame.setLayout(layout);

        JButton nButton = new JButton("North");
        JButton sButton = new JButton("South");
        JButton wButton = new JButton("West");
        JButton eButton = new JButton("East");
        JButton cButton = new JButton("Center");

        frame.add(nButton, BorderLayout.NORTH);
        frame.add(sButton, BorderLayout.SOUTH);
        frame.add(wButton, BorderLayout.WEST);
        frame.add(eButton, BorderLayout.EAST);
        frame.add(cButton, BorderLayout.CENTER);

        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

8.a] Action Event



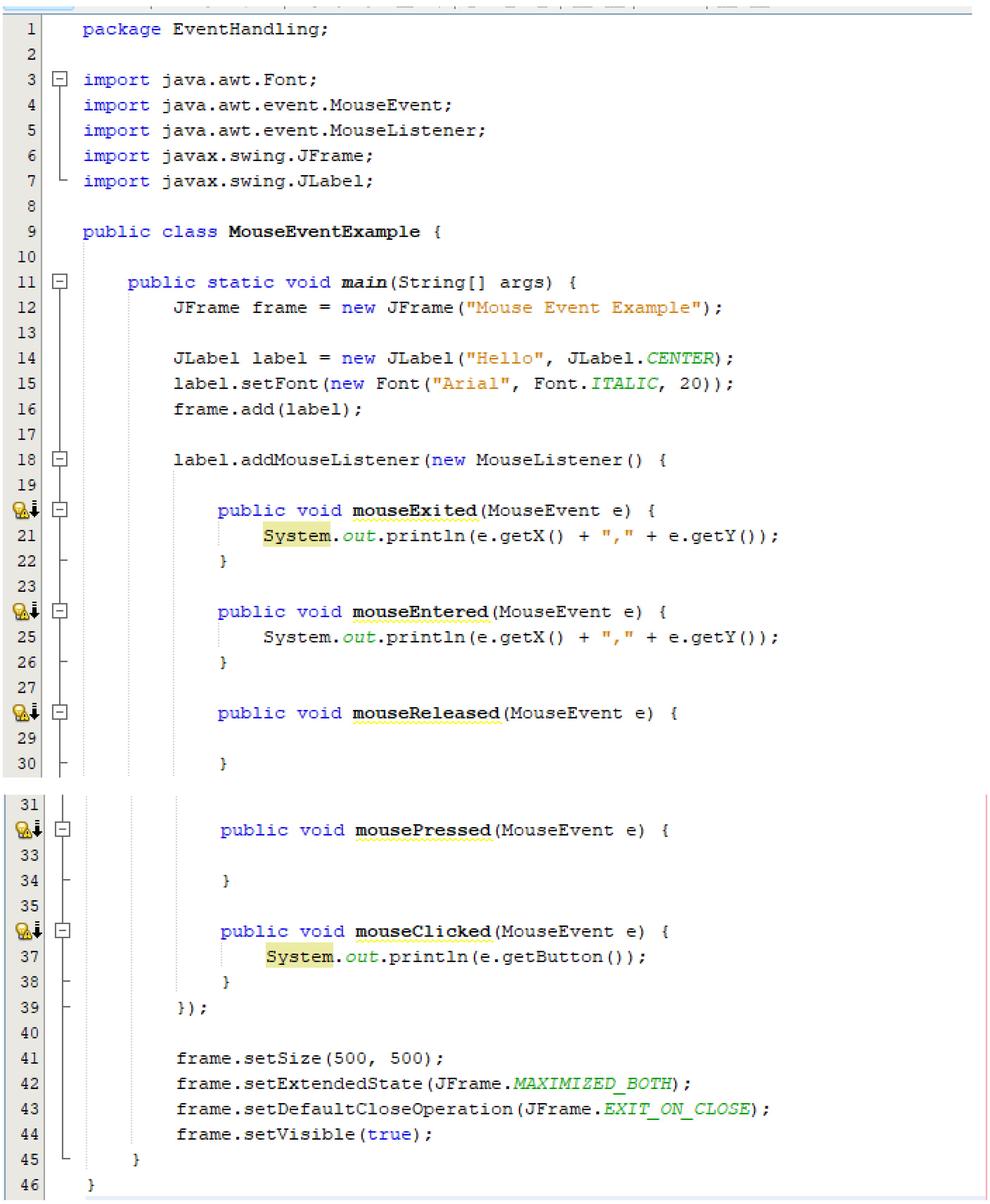
```java
package EventHandling;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class ActionEventExample {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Action Event Example");

        JButton button = new JButton("My Button");
        frame.add(button);

        ButtonActionListner listner = new ButtonActionListner(frame);
        button.addActionListener(listner);

        frame.setSize(500, 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



```java

class ButtonActionListner implements ActionListener {

    private JFrame parentFrame;

    public ButtonActionListner(JFrame frame) {
        this.parentFrame = frame;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(parentFrame, e.getActionCommand() + " clicked");
    }

}
```

8.b] Mouse Event

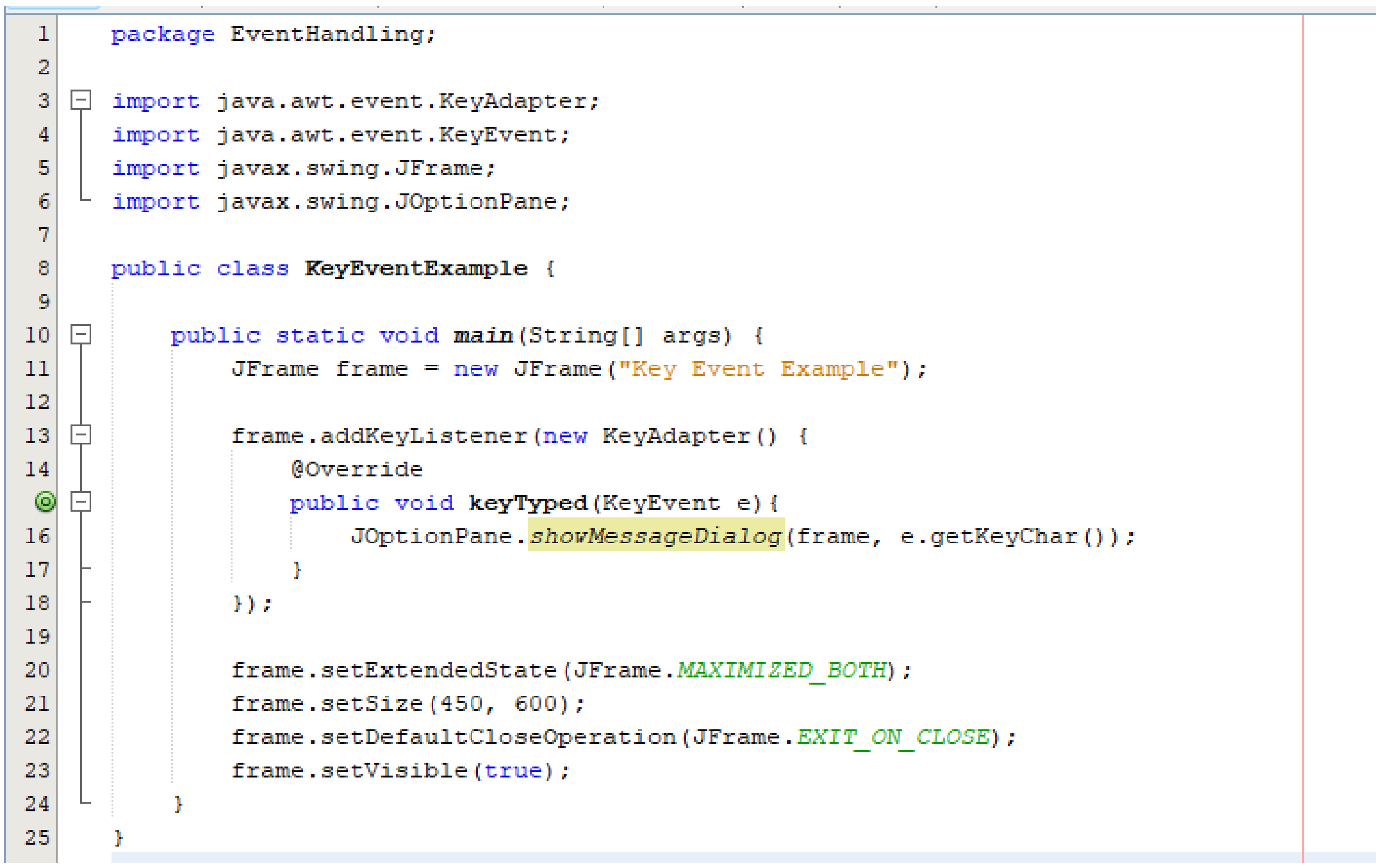
Note: Same practical can be used for practical 10.



```java
package EventHandling;

import java.awt.Font;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class MouseEventExample {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Mouse Event Example");

        JLabel label = new JLabel("Hello", JLabel.CENTER);
        label.setFont(new Font("Arial", Font.ITALIC, 20));
        frame.add(label);

        label.addMouseListener(new MouseListener() {

            public void mouseExited(MouseEvent e) {
                System.out.println(e.getX() + "," + e.getY());
            }

            public void mouseEntered(MouseEvent e) {
                System.out.println(e.getX() + "," + e.getY());
            }

            public void mouseReleased(MouseEvent e) {

            }

            public void mousePressed(MouseEvent e) {

            }

            public void mouseClicked(MouseEvent e) {
                System.out.println(e.getButton());
            }
        });

        frame.setSize(500, 500);
        frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

8.c] Key Event

Note: Same practical can be used for practical 9.



```java
package EventHandling;

import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class KeyEventExample {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Key Event Example");

        frame.addKeyListener(new KeyAdapter() {
            @Override
            public void keyTyped(KeyEvent e){
                JOptionPane.showMessageDialog(frame, e.getKeyChar());
            }
        });

        frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
        frame.setSize(450, 600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```