Final Exam

Name: Mrunmayee Tulshibagwale
ID: R11815197

Code-1: Vehicle Management

(Your Codes)

- -

```
package Vehicle_Management;

public abstract class Vehicle {
	private String type;
	private String brand;

	//Default constructor
	public Vehicle() {
		this.type = "Unknown";
		this.brand = "Unknown";
	}

	//Parameters constructor
	public Vehicle(String type,String brand) {
		this.type = type;
		this.brand =brand;
```

```
    }

    //getters
    public String getType() {
        return type;
    }

    public String getBrand() {
        return brand;
    }

    //Setters
    public void setType(String type) {
        this.type = type;
    }

    public void setBrand(String brand) {
        this.brand =brand;
    }

    //Abstract Method

    public abstract String getDetails();
}
```

```
package Vehicle_Management;
```

```
public interface VehicleDetails {
	void setColor(String color);
	String getColor();
}
```

```
package Vehicle_Management;

public class Car extends Vehicle implements VehicleDetails {
	private String color;
	private int numberofWheels;

	//Default constructor
	public Car() {
		super(); //Calls the default constructor of Vehicle
		this.color = "Unknown";
		this.numberofWheels = 4;//Default to 4 wheels
	}

	//Parameterized constructor
	public Car(String type, String brand, String color, int
numberofWheels) {
		super(type,brand);
		this.color = color;
		this.numberofWheels = numberofWheels;
	}
```

```
    //Implement  methods from vehicleDetails
    @Override
    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public String getColor() {
        return color;
    }

    //Implement abstract method from Vehicle
    @Override
    public String getDetails() {
        return "Type: " + getType() + ",Brand: " + getBrand() + ",
Color:  " + color + ", Number of Wheels: " + numberofWheels;
    }

    //Getters and Setters of the Wheels
    public int getNumberofWheels() {
        return numberofWheels;
    }

    public void setNumberofWheels(int numberofWheels) {
```

```
        this.numberofWheels = numberofWheels;
    }
}
```

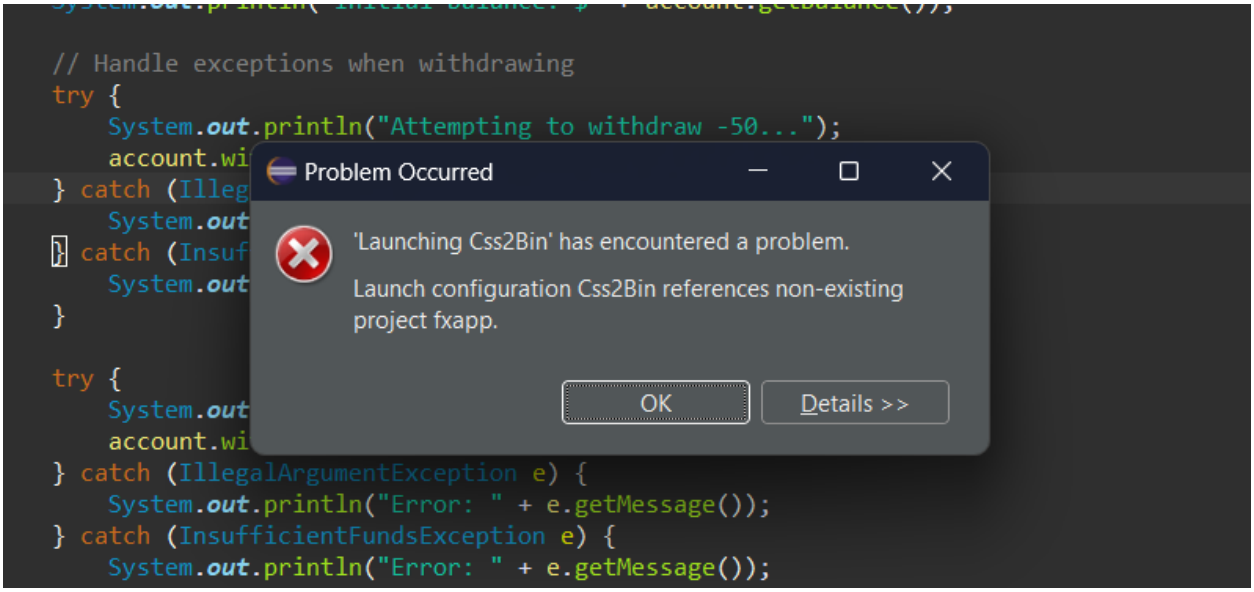
```
package Vehicle_Management;

public class VehicleDemo {

    public static void main (String[] args) {
        //Instantiate a car object
        Car car = new Car("Car", "Hyundai", "White", 4);

        //Print Details
        System.out.println("Car Details: ");
        System.out.println(car.getDetails());
    }
}
```

Code 2: bank Account Management

```
package BankAccount_Management;

class InsufficientFundsException extends Exception {
        public InsufficientFundsException(String message) {
                super(message);
        }

}
package BankAccount_Management;

// BankAccount class
class BankAccount {
    private double balance;

    public BankAccount() {
        this.balance = 50.0; // Starting balance
    }

     public void withdraw(double amount) throws
InsufficientFundsException, IllegalArgumentException {
        if (amount <= 0) {
                throw new IllegalArgumentException("Withdrawal amount
must be positive");
        }
```

```
        if (amount > balance) {
            throw new InsufficientFundsException ("Insufficient funds:
cannot withdraw " + amount);
        }

        balance -= amount;
    }

    public double getBalance() { return balance; }
}

package BankAccount_Management;

public class BankAccountDemo {
    public static void main(String[] args) {
        // Instantiate BankAccount object
        BankAccount account = new BankAccount();

        // Display initial balance
        System.out.println("Initial balance: $" + account.getBalance());

        // Handle exceptions when withdrawing
        try {
            System.out.println("Attempting to withdraw -50...");
```

```
            account.withdraw(-50);
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (InsufficientFundsException e) {
            System.out.println("Error: " + e.getMessage());
        }

        try {
            System.out.println("Attempting to withdraw 100...");
            account.withdraw(100);
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (InsufficientFundsException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
}
```



Cannot run the code due to this error. As the grader mentioned that he will run on his eclipse and see if I get the output or not.