

**Національний технічний університет України**  
**«Київський політехнічний інститут»**  
**Факультет інформатики та обчислювальної техніки**  
**Кафедра обчислювальної техніки**

**Об'єктно-орієнтовне програмування**  
**Лабораторна робота №6**

Виконала:  
студентка групи ІО-25  
Антоненко В.С.  
Залікова книжка №2501

Перевірив  
Алещенко О.

Київ 2023р.

## Лабораторна робота №6

**Тема:** Наслідування та поліморфізм.

**Мета:** Ознайомлення з механізмом наслідування та принципом поліморфізму. Використання механізму наслідування та принципу поліморфізму в мові Java. Здобуття навичок у використанні механізму наслідування та принципу поліморфізму.

### Завдання

1. Визначити  $C_{13}$  як остачу від ділення номера залікової книжки студента на 13.
2. В залежності від  $C_{13}$  визначити варіант завдання:

$C_{13}$	Варіант завдання
7	Визначити ієрархію рухомого складу залізничного транспорту. Створити пасажирський потяг. Порахувати загальну чисельність пасажирів і багажу в потязі. Провести сортування вагонів потягу за рівнем комфортності. Знайти вагон в потязі, що відповідає заданому діапазону кількості пасажирів.

3. Створити узагальнений клас та не менше 3 класів-нащадків, що описують задану варіантом (п.2) область знань. Створити клас, що складається з масиву об'єктів, з яким можна виконати вказані варіантом дії. Необхідно обробити всі виключні ситуації, що можуть виникнути під час виконання програмного коду. Всі змінні повинні бути описані та значення їх задані у виконавчому методі. Код повинен відповідати стандартам [JCC](#) та бути детально задокументований.

### Код:

```
import java.util.Arrays;
import java.util.Comparator;

/**
 * Клас, що описує базовий рухомий склад залізничного транспорту.
 */
class RollingStock {
    private int number;

    public RollingStock(int number) {
        this.number = number;
    }

    public int getNumber() {
        return number;
    }
}

/**
 * Клас, що описує пасажирський вагон.
 */
class PassengerCarriage extends RollingStock {
    private int passengerCount;
    private int comfortLevel;

    public PassengerCarriage(int number, int passengerCount, int comfortLevel) {
        super(number);
        this.passengerCount = passengerCount;
        this.comfortLevel = comfortLevel;
    }

    public int getPassengerCount() {
        return passengerCount;
    }
}
```

```

    public int getComfortLevel() {
        return comfortLevel;
    }
}

/**
 * Клас, що описує вагон для багажу.
 */
class BaggageCarriage extends RollingStock {
    private int baggageCount;

    public BaggageCarriage(int number, int baggageCount) {
        super(number);
        this.baggageCount = baggageCount;
    }

    public int getBaggageCount() {
        return baggageCount;
    }
}

/**
 * Клас, що описує пасажирський потяг.
 */
class PassengerTrain {
    private RollingStock[] rollingStock;

    public PassengerTrain(RollingStock[] rollingStock) {
        this.rollingStock = rollingStock;
    }

    public int getTotalPassengerCount() {
        int totalPassengerCount = 0;
        for (RollingStock stock : rollingStock) {
            if (stock instanceof PassengerCarriage) {
                PassengerCarriage carriage = (PassengerCarriage) stock;
                totalPassengerCount += carriage.getPassengerCount();
            }
        }
        return totalPassengerCount;
    }

    public int getTotalBaggageCount() {
        int totalBaggageCount = 0;
        for (RollingStock stock : rollingStock) {
            if (stock instanceof BaggageCarriage) {
                BaggageCarriage carriage = (BaggageCarriage) stock;
                totalBaggageCount += carriage.getBaggageCount();
            }
        }
        return totalBaggageCount;
    }

    public void sortByComfortLevel() {
        Arrays.sort(rollingStock, new Comparator<RollingStock>() {
            @Override
            public int compare(RollingStock o1, RollingStock o2) {
                if (o1 instanceof PassengerCarriage && o2 instanceof
PassengerCarriage) {
                    PassengerCarriage carriage1 = (PassengerCarriage) o1;
                    PassengerCarriage carriage2 = (PassengerCarriage) o2;
                    return carriage1.getComfortLevel() - carriage2.getComfortLevel();
                }
            }
        });
    }
}

```

```

        return 0;
    }
    });
}

public RollingStock findCarriageByPassengerCountRange(int minPassengerCount, int
maxPassengerCount) {
    for (RollingStock stock : rollingStock) {
        if (stock instanceof PassengerCarriage) {
            PassengerCarriage carriage = (PassengerCarriage) stock;
            if (carriage.getPassengerCount() >= minPassengerCount &&
carriage.getPassengerCount() <= maxPassengerCount) {
                return carriage;
            }
        }
    }
    return null;
}
}

/**
 * Головний клас програми.
 */
public class Main {
    public static void main(String[] args) {
        // Створення вагонів потягу
        RollingStock[] rollingStock = new RollingStock[5];
        rollingStock[0] = new PassengerCarriage(1, 40, 3);
        rollingStock[1] = new PassengerCarriage(2, 30, 2);
        rollingStock[2] = new BaggageCarriage(3, 50);
        rollingStock[3] = new PassengerCarriage(4, 20, 1);
        rollingStock[4] = new PassengerCarriage(5, 35, 2);

        // Створення пасажирського потягу
        PassengerTrain train = new PassengerTrain(rollingStock);

        // Розрахунок загальної кількості пасажирів та багажу в потязі
        int totalPassengerCount = train.getTotalPassengerCount();
        int totalBaggageCount = train.getTotalBaggageCount();
        System.out.println("Total passenger count: " + totalPassengerCount);
        System.out.println("Total baggage count: " + totalBaggageCount);

        // Сортування вагонів за рівнем комфортності
        train.sortByComfortLevel();
        System.out.println("Sorted carriages by comfort level:");

        for (RollingStock stock : rollingStock) {
            if (stock instanceof PassengerCarriage) {
                PassengerCarriage carriage = (PassengerCarriage) stock;
                System.out.println("Carriage " + carriage.getNumber() + ", Comfort
Level: " + carriage.getComfortLevel());
            }
        }

        // Знаходження вагона з пасажирськими місцями в заданому діапазоні кількості
пасажирів
        int minPassengerCount = 25;
        int maxPassengerCount = 40;
        RollingStock foundCarriage =
train.findCarriageByPassengerCountRange(minPassengerCount, maxPassengerCount);
        if (foundCarriage != null) {
            System.out.println("Found carriage with passenger count in range (" +
minPassengerCount + "-" + maxPassengerCount + "): " + foundCarriage.getNumber());
        } else {

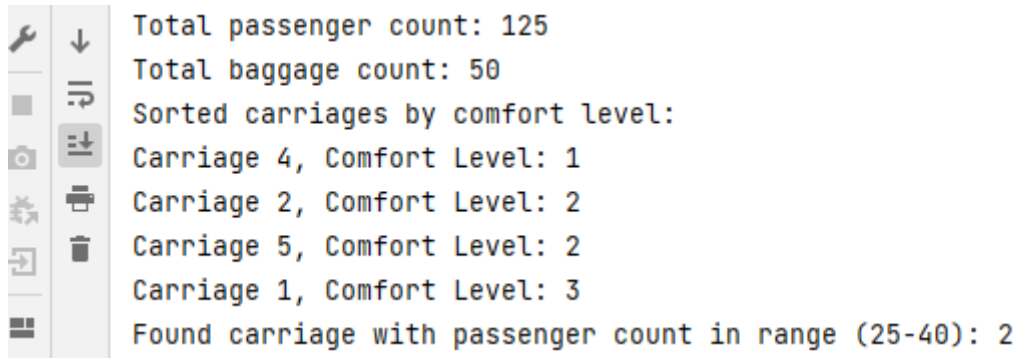
```

```

        System.out.println("No carriage found with passenger count in range (" +
minPassengerCount + "-" + maxPassengerCount + ")");
    }
}
}

```

### Приклад роботи програми:



```

Total passenger count: 125
Total baggage count: 50
Sorted carriages by comfort level:
Carriage 4, Comfort Level: 1
Carriage 2, Comfort Level: 2
Carriage 5, Comfort Level: 2
Carriage 1, Comfort Level: 3
Found carriage with passenger count in range (25-40): 2

```

Рис.1

### Алгоритм роботи програми:

1. Визначені класи:
  - RollingStock - базовий клас для рухомого складу, містить номер вагона.
  - PassengerCarriage - клас, що описує пасажирський вагон, наслідується від RollingStock і додатково містить кількість пасажирів і рівень комфортності.
  - BaggageCarriage - клас, що описує вагон для багажу, наслідується від RollingStock і містить кількість багажу.
  - PassengerTrain - клас, що описує пасажирський потяг, містить масив рухомого складу.
2. Клас PassengerTrain має такі методи:
  - Конструктор PassengerTrain(RollingStock[] rollingStock) - створює об'єкт пасажирського потягу з заданим рухомим складом.
  - Метод getTotalPassengerCount() - обчислює загальну кількість пасажирів у потязі, перебираючи вагони та підраховуючи кількість пасажирів у пасажирських вагонах.
  - Метод getTotalBaggageCount() - обчислює загальну кількість багажу у потязі, перебираючи вагони та підраховуючи кількість багажу у вагонах для багажу.
  - Метод sortByComfortLevel() - сортує вагони потягу за рівнем комфортності, використовуючи Arrays.sort() та компаратор, який порівнює комфортність двох пасажирських вагонів.
  - Метод findCarriageByPassengerCountRange(int minPassengerCount, int maxPassengerCount) - знаходить вагон з пасажирськими місцями, кількість пасажирів у якому знаходиться в заданому діапазоні. Перебираються вагони і перевіряється, чи є вони пасажирськими вагонами і чи потрапляють в діапазон кількості пасажирів.
3. Головний клас Main містить метод main, в якому виконуються наступні дії:
  - Створюється масив rollingStock, який містить рухомий склад потягу (пасажирські та вагони для багажу).
  - Створюється об'єкт PassengerTrain зі створеним рухомим складом.
  - Обчислюється загальна кількість пасажирів і багажу у потязі і виводиться на екран.
  - Вагони сортуються за рівнем комфортності і виводяться на екран.
  - Знаходиться вагон з пасажирськими місцями в заданому діапазоні кількості пасажирів і виводиться його номер на екран.

**Висновок:** під час виконання даної лабораторної роботи я ознайомила з механізмом наслідування та принципом поліморфізму. А алгоритм роботи програми включає створення об'єктів вагонів і потягу, обчислення загальної кількості пасажирів і багажу, сортування вагонів за комфортністю та пошук вагона з певною кількістю пасажирів.