Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

# Лабораторна робота №8

із дисципліни «Програмування. Частина 1. Основи програмування»

**Тема: «Основи ООП»**

Виконали:
Студентки групи ІА-24
Ганжа Х. М.
Кійко А. О.
Мелешко Ю. С.

Перевірив:
Колеснік Валерій Миколайович

Київ-2022

# Варіант-5
# Хід роботи:

## 1. Код програми:

Lab8_Main.java 1, U | Automobile.java 1, U | Ship.java 1, U | VehicleBase.java 1, U | VehicleInterface.java 1, U

D: > мое > progr_repo > PF1_2022 > IA-24 > Team 5 > LAB 8 > src > Lab8_Main.java > Lab8_Main > main(String[])

```java
package lab8;

/**
 * Lab 8 Main class (need for checking task)
 *
 * @author Kiiko Anna
 */
public class Lab8_Main {

    /**
     * Entry point. Tests the different classes on this task.
     *
     * @param args the standart parameter of the function.
     */
    Run | Debug
    public static void main(String[] args) {
        try {
            System.out.println(x: "-------------------------------------------------");
            System.out.println(x: "\tLab 8. Team 5.");
            System.out.println(x: "-------------------------------------------------");

            VehicleInterface interface1 =
                new Automobile(model: "Ferrari", colour: "red", EngineType.Gasoline);
            System.out.println("interface1.toString() => " + interface1.toString());
            System.out.println(x: "-------------------------------------------------");

            Automobile automobile1 =
                new Automobile(model: "Renault Daster", colour: "black", EngineType.Diesel);
            System.out.println("automobile1.toString() => " + automobile1.toString());
            automobile1.setRegistration(regNumber: "AA2022EX");
            System.out.println("Після присвоєння реєстраційного номера - automobile1.toString() =>\n" + automobile1.toString());
            System.out.println(x: "-------------------------------------------------");

            Automobile automobile2 =
                new Automobile(model: "Volkswagen Golf 5", colour: "pink", EngineType.Diesel);
            System.out.println("automobile2.toString() => " + automobile2.toString());
            VehicleInterface interface2 = automobile2;

            automobile2.setRegistration(regNumber: "AX2020EX");
            automobile2.repainting(newColor: "dark green");
            automobile2.updateFuelSystem(EngineType.Gas, capacityFuel: 55.0);
            System.out.println("Після виконання методів - automobile2.toString() =>\n" + automobile2.toString());
            System.out.println(x: "-------------------------------------------------");

            Ship ship1 =
                new Ship(model: "Ulsan (FFX Batch III)", colour: "blue", EngineType.Nuclear, displacement_tons: 3200);
            System.out.println("ship1.toString() => " + ship1.toString());
            VehicleInterface interface3 = ship1;
            System.out.println(x: "-------------------------------------------------");

            System.out.println(x: "після присвоєння реєстраційного номеру (з використанням методів інтерфейсу)...");
            interface1.setRegistration(regNumber: "AA0001AA");
            interface2.setRegistration(regNumber: "AA0002BB");
            interface3.setRegistration(regNumber: "AA0003CC");
            System.out.println("interface1.toString() => " + interface1.toString());
            System.out.println("interface2.toString() => " + interface2.toString());
            System.out.println("interface3.toString() => " + interface3.toString());

            System.out.println(x: "-------------------------------------------------");
            System.out.println("interface1.equals(interface2): " + interface1.equals(interface2));
            System.out.println("interface1.equals(interface3): " + interface1.equals(interface3));
            System.out.println(x: "-------------------------------------------------");

        } catch (Exception ex) {
            System.out.println("EXCEPTION! " + ex.getMessage());
        }
    }
}

```

Lab8_Main.java 1, U | Automobile.java 1, U | Ship.java 1, U | VehicleBase.java 1, U | VehicleInterface.java 1, U

D: > мое > progr_repo > PF1_2022 > IA-24 > Team 5 > LAB 8 > src > Automobile.java > Automobile > toString()

```java
package lab8;

/**
 * Lab 8: Automobile class
 *
 * @author Kiiko Anna
 */
public class Automobile extends VehicleBase implements VehicleInterface{
    public Automobile(String model, String colour, EngineType typeEngin){
        super(model, colour, typeEngin);
        super.capacityFuel = 40.0;
    }

    // перевантаження (overloading) конструктору
    public Automobile(String model, String colour, EngineType typeEngin, int seatsNumber){
        this(model, colour, typeEngin); // can be used super(model, colour, typeEngin) instead of this()
        this.seatsNumber = seatsNumber;
    }

    @Override
    public String toString(){
        return "Автомобіль " + super.toString() + ", " + getSeatsNumber() + " місця (місць).";
    }

    @Override
    public Boolean equals(VehicleInterface refMoovingTool){
        return (refMoovingTool instanceof Automobile);
    }

    private int seatsNumber = 4;
    public int getSeatsNumber(){
        return seatsNumber;
    }

}
```

Lab8_Main.java 1, U | Automobile.java 1, U | Ship.java 1, U | VehicleBase.java 1, U | VehicleInterface.java 1, U

D: > мое > progr_repo > PF1_2022 > IA-24 > Team 5 > LAB 8 > src > Ship.java > Ship > toString()

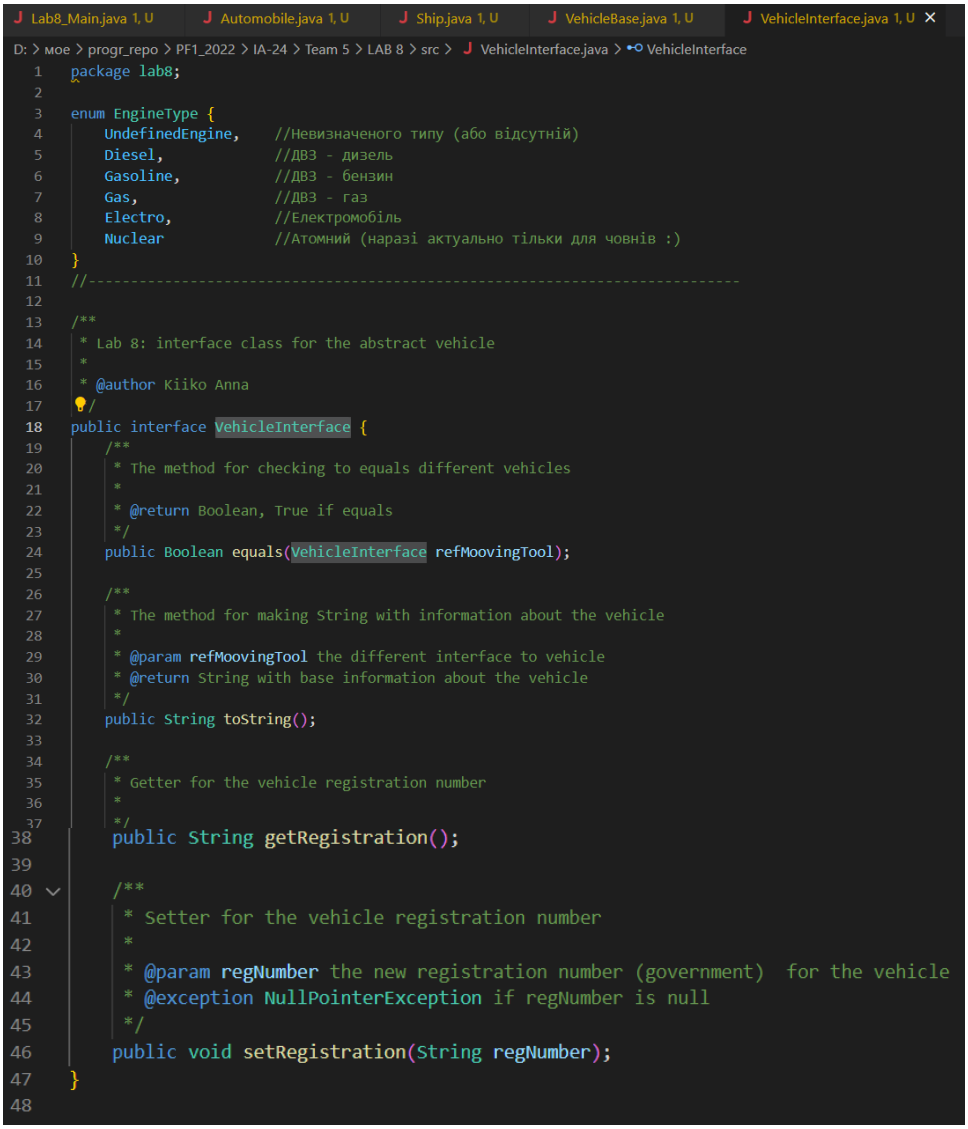
```java
package lab8;

/**
 * Lab 8: Ship class
 *
 * @author Kiiko Anna
 */
public class Ship extends VehicleBase{
    public Ship(String model, String colour, EngineType typeEngin, int displacement_tons){
        super(model, colour, typeEngin);
        this.shipDisplacement = displacement_tons;
    }

    @Override
    public String toString(){
        return "Човен " + super.toString() + ". Човен має водотзміщення " + getShipDisplacement() + " тон.";
    }

    @Override
    public Boolean equals(VehicleInterface refMoovingTool){
        return (refMoovingTool instanceof Ship);
    }

    // the ship displacement in tons
    private Integer shipDisplacement;
    public Integer getShipDisplacement(){
        return shipDisplacement;
    }
}
```

Lab8_Main.java 1, U | Automobile.java 1, U | Ship.java 1, U | VehicleBase.java 1, U | VehicleInterface.java 1, U

D: > мое > progr_repo > PF1_2022 > IA-24 > Team 5 > LAB 8 > src > VehicleBase.java > VehicleBase > toString()

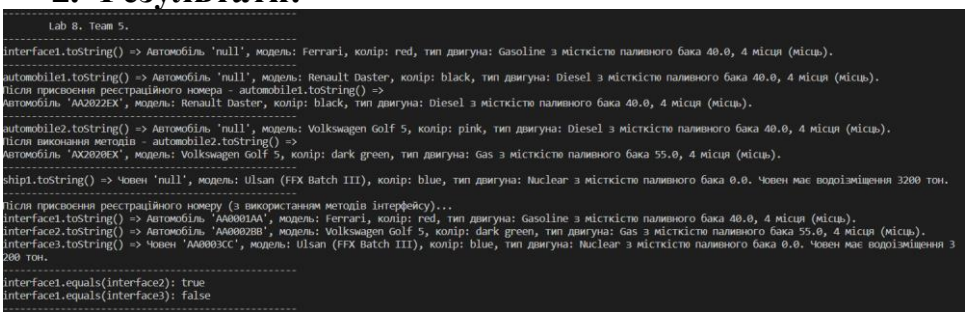
```java
package lab8;

/**
 * Lab 8: abstract class with the base methods for vehicles
 *
 * @author Kiiko Anna
 */
abstract class VehicleBase implements VehicleInterface{
    /**
     * The parametrized constructor (can be used only in heirs)
     */
    protected VehicleBase(String model, String colour, EngineType typeEngin){
        this.model = model;
        this.colour = colour;
        this.typeEngin = typeEngin;
    }

    /**
     * The method for making String with information about the vehicle
     * (base parameters for the abstract class), but not well that the interface not
     * extracted to diferent GUI class, it will be better according to SOLID-principies etc.
     *
     * @return String with base information about the vehicle
     */
    public String toString(){
        String prepStr = "'" + regNumber + "'";
        prepStr += ", модель: " + model;
        prepStr += ", кол□р: " + colour;
        prepStr += ", тип двигуна: ";
        switch(typeEngin) {
            case UndefinedEngine:
                prepStr += "Undefined";
                break;
            case Diesel:
                prepStr += "Diesel";
                break;
            case Gasoline:
                prepStr += "Gasoline";
                break;
            case Gas:
                prepStr += "Gas";
                break;
            case Electro:
                prepStr += "Electro";
                break;
            case Nuclear:
                prepStr += "Nuclear";
                break;
            default:
                prepStr += "UNKNOWN";
                break;
        }
        prepStr += " з м□стк□стю паливного бака " + capacityFuel.toString();
        return prepStr;
    }

    private String model;
    private String colour;
    protected Double capacityFuel = 0.0; //can be updated into heirs constructors.
    private EngineType typeEngin = EngineType.UndefinedEngine;
    private String regNumber;

    /**
     * Update the fuel system (all setter for fuel system information)
     *
     * @param newEngineType the new engine type (after upgrade).
     * @param newCapacityFuel new fuel capacity (after upgrade).
     * @exception NullPointerException if parameters are null
     * @exception IllegalArgumentException if capacityFuel is negative
     */
    public void updateFuelSystem(EngineType newEngineType, Double capacityFuel){
        if ((null == newEngineType) || (null == capacityFuel)) {
            throw new NullPointerException(s: "Arguments are null!");
        }
```

```
        if (capacityFuel < 0.0) {
            throw new IllegalArgumentException(s: "The capacityFuel cannot be negative!");
        }
        typeEngin = newEngineType;
        this.capacityFuel = capacityFuel;
    }

    /**
     * Update the fuel system (part setter) -> перевантаження (overloading)
     *
     * @param newCapacityFuel new fuel capacity (after upgrade).
     * @exception NullPointerException if newCapacityFuel is null
     * @exception IllegalArgumentException if newCapacityFuel is negative
     */
    public void updateFuelSystem(Double newCapacityFuel){ //перевантаження (overloading) updateFuelSystem()
        if (null == newCapacityFuel) {
            throw new NullPointerException(s: "The capacityFuel is null!");
        }
        if (newCapacityFuel < 0.0) {
            throw new IllegalArgumentException(s: "The capacityFuel cannot be negative!");
        }
        capacityFuel = newCapacityFuel;
    }

    /**
     * Repainting the vehicle colour (as setter)
     *
     * @param newColor the new colour for the vehicle
     * @exception NullPointerException if newColor is null
     */
    public void repainting(String newColor){
        if (null == newColor) {
            throw new NullPointerException(s: "The newColor is null!");
        }
        colour = newColor;
    }

    /**
     * Setter for the vehicle registration number
     *
     * @param regNumber the new registration number (government)  for the vehicle
     * @exception NullPointerException if regNumber is null
     */
    public void setRegistration(String regNumber){
        if (null == regNumber) {
            throw new NullPointerException(s: "The regNumber is null!");
        }
        this.regNumber = regNumber;
    }

    /**
     * Getter for the vehicle registration number
     *
     */
    public String getRegistration(){
        return this.regNumber;
    }
}
```



```java
package lab8;

enum EngineType {
    UndefinedEngine,    //Невизначеного типу (або відсутній)
    Diesel,             //ДВЗ - дизель
    Gasoline,           //ДВЗ - бензин
    Gas,                //ДВЗ - газ
    Electro,            //Електромобіль
    Nuclear             //Атомний (наразі актуально тільки для човнів :)
}
//---------------------------------------------------------------------------

/**
 * Lab 8: interface class for the abstract vehicle
 *
 * @author Kiiko Anna
 */
public interface VehicleInterface {
    /**
     * The method for checking to equals different vehicles
     *
     * @return Boolean, True if equals
     */
    public Boolean equals(VehicleInterface refMoovingTool);

    /**
     * The method for making String with information about the vehicle
     *
     * @param refMoovingTool the different interface to vehicle
     * @return String with base information about the vehicle
     */
    public String toString();

    /**
     * Getter for the vehicle registration number
     *
     */
    public String getRegistration();

    /**
     * Setter for the vehicle registration number
     *
     * @param regNumber the new registration number (government)  for the vehicle
     * @exception NullPointerException if regNumber is null
     */
    public void setRegistration(String regNumber);
}
```

## 2. Результати:



```
------------------------------------------------
        Lab 8. Team 5.
------------------------------------------------
interface1.toString() => Автомобіль 'null', модель: Ferrari, колір: red, тип двигуна: Gasoline з місткістю паливного бака 40.0, 4 місця (місць).
------------------------------------------------
automobile1.toString() => Автомобіль 'null', модель: Renault Daster, колір: black, тип двигуна: Diesel з місткістю паливного бака 40.0, 4 місця (місць).
Після присвоєння реєстраційного номера - automobile1.toString() =>
Автомобіль 'AA2022EX', модель: Renault Daster, колір: black, тип двигуна: Diesel з місткістю паливного бака 40.0, 4 місця (місць).
------------------------------------------------
automobile2.toString() => Автомобіль 'null', модель: Volkswagen Golf 5, колір: pink, тип двигуна: Diesel з місткістю паливного бака 40.0, 4 місця (місць).
Після виконання методів - automobile2.toString() =>
Автомобіль 'AX2020EX', модель: Volkswagen Golf 5, колір: dark green, тип двигуна: Gas з місткістю паливного бака 55.0, 4 місця (місць).
------------------------------------------------
ship1.toString() => Човен 'null', модель: Ulsan (FFX Batch III), колір: blue, тип двигуна: Nuclear з місткістю паливного бака 0.0. Човен має водоізміщення 3200 тон.
------------------------------------------------
Після присвоєння реєстраційного номеру (з використанням методів інтерфейсу)...
interface1.toString() => Автомобіль 'AA0001AA', модель: Ferrari, колір: red, тип двигуна: Gasoline з місткістю паливного бака 40.0, 4 місця (місць).
interface2.toString() => Автомобіль 'AA0002BB', модель: Volkswagen Golf 5, колір: dark green, тип двигуна: Gas з місткістю паливного бака 55.0, 4 місця (місць).
interface3.toString() => Човен 'AA0003CC', модель: Ulsan (FFX Batch III), колір: blue, тип двигуна: Nuclear з місткістю паливного бака 0.0. Човен має водоізміщення 3
200 тон.
------------------------------------------------
interface1.equals(interface2): true
interface1.equals(interface3): false
------------------------------------------------
```

**Висновок: розроблено інтерфейс (VehicleInterface), що визначає декларації методів, а також базовий абстрактний клас (VehicleBase), що реалізує даний інтерфейс, з реалізацією базових методів для довільного засобу пересування. Створено класи-нащадки базового класу: Автомобіль (Automobile) та Човен (Ship), що містять перевизначені (override) методи toString() та equals(), а також використано перевантаження (overloading) конструктору в класі Автомобіль (Automobile), а також перевантаження методу updateFuelSystem() в базовому класі.**
**Продемонстровано реалізацію сетерів та гетерів як в базовому класі, так і в класах-нащадках, а також використання: this, super.**