МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних технологій, автоматики та метрології

кафедра "Електронних обчислювальних машин"

Звіт

з лабораторної роботи №3

дисципліни «Кросплатформні засоби програмування»

Варіант 26

Виконала:
студент групи КІ-306
Тимків С. В.
Прийняв:
Олексів М. В.

Львів – 2024

ЛАБОРАТОРНА РОБОТА №3

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java

Завдання (Варіант №26)

1.Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2.Автоматично згенерувати документацію до розробленого пакету.
3.Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
5. Дати відповідь на контрольні запитання.

GitHub Repository:
https://github.com/SofiaTymkiv/CPPT_Tymkiv_SV_KI-36_2.git

Виконання завдання

EngineRunnable

```
public interface EngineRunnable {

    void startEngine();

    void stopEngine();
}
```

Location

```
public class Location {

    private String country;

    private String city;
    private String street;

    public Location(String country, String city, String street)
{        this.country = country;        this.city =
city;        this.street = street;
    }

    public static Location unknown() {        return new Location("unknown", "unknown",
"unknown");
    }

    public String getCountry() {        return
country;
    }

    public void setCountry(String country) {        this.country = country;
    }
```

```
    public String getCity() {        return city;
    }

    public void setCity(String city) {        this.city = city;
    }

    public String getStreet() {        return street;
    }

    public void setStreet(String street) {        this.street = street;
    }
}
```

Main

```
public class Main {

    public static void main(String[] args) {        // Create owner and
initial supplies
        Person owner = new Person("John Doe", 45);        List<Supply> supplies = new
ArrayList<>();        supplies.add(new Supply(10, "Water Bottles"));
supplies.add(new Supply(5, "Food Packs"));
        Location startLocation = new Location("USA", "New York",
"Hudson River");
        try (RowingBoat boat = new MotorBoat(owner, supplies, startLocation)) {
```

```
        // 1. Add a new supply          boat.addSupply(new Supply(15, "Life Jackets"));

        // 2. Remove a supply           boat.removeSupply("Food Packs");

        // 3. Move the boat to a new location
        Location newLocation = new Location("Canada", "Toronto", "Lake Ontario");
boat.moveToNewLocation(newLocation);

        // 4. Get total supplies on the boat            int totalSupplies =
boat.getTotalSupplies();
        System.out.println("Total supplies: " + totalSupplies);

        // 5. Get owner's name
        String ownerName = boat.getOwnerName();          System.out.println("Boat owner: "
+ ownerName);

        // 6. Change the owner of the boat          Person newOwner = new
Person("Jane Smith", 30);          boat.changeOwner(newOwner);
        // 7. Get current location of the boat
        Location currentLocation = boat.getCurrentLocation();
        System.out.println("Current location: " + currentLocation.getCity());

        // 8. Check if a specific supply exists          boolean hasWater =
boat.hasSupply("Water Bottles");
        System.out.println("Has water bottles: " + hasWater);
```

```
        // 9. Get the total quantity of all supplies            int totalQuantity =
boat.getTotalSupplyQuantity();
            System.out.println("Total supply quantity: " + totalQuantity);

            // 10. Print a summary of the boat              boat.printBoatSummary();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

MotorBoat

```
public class MotorBoat extends RowingBoat implements EngineRunnable {

    private static final Logger LOG = Logger.getLogger(MotorBoat.class.getName());

    /**
*       Constructs a new RowingBoat with an owner, list of supplies, and location.
     *
*       @param owner    The owner of the boat.      * @param supplies The supplies on the
boat.      * @param location The initial location of the boat.
```

```
 *      @throws FileNotFoundException If the log file cannot be created or opened.
     */    public MotorBoat(Person owner, List<Supply> supplies, Location location)
throws FileNotFoundException {        super(owner, supplies, location);
    }

    @Override    public void printBoatSummary() {        LOG.info("just a cool
motor boat, hell yeah!!");
    }

    @Override    public void startEngine()
{        LOG.info("Engine Started");
    }

    @Override    public void stopEngine()
{        LOG.info("Engine Stopped");
    }

    @Override    public void close() throws Exception
{        super.close();        stopEngine();
    }
}
```

Old RowingBoat, but abstract

```
public abstract class RowingBoat implements AutoCloseable
```

Висновок: Я ознайомилася з спадкуванням та інтерфейсами у мові Java