МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних технологій, автоматики та метрології

кафедра "Електронних обчислювальних машин"

Звіт

з лабораторної роботи №2

дисципліни «Кросплатформні засоби програмування»

Варіант 26

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

Львів – 2024

ЛАБОРАТОРНА РОБОТА №2

Мета роботи: ознайомитися з процесом розробки класів та пакетів мовою Java.

Завдання (Варіант №26)

1.Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам: • програма має розміщуватися в пакеті Група.Прізвище.Lab2;
2.клас має містити мінімум 3 поля, що є об’єктами класів, які описують складові частини предметної області;
3.клас має містити кілька конструкторів та мінімум 10 методів;
4.для тестування і демонстрації роботи розробленого класу розробити клас-драйвер; методи класу мають вести протокол своєї діяльності, що записується у файл;
5.розробити механізм коректного завершення роботи з файлом (не надіятися на метод finalize()); • програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
6.Автоматично згенерувати документацію до розробленої програми.
7.Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
8.Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
9.Дати відповідь на контрольні запитання.

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

Виконання завдання

Location

```
package ki306.tymkiv.lab2;


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

```
 package ki306.tymkiv.lab2;

import java.util.ArrayList; import java.util.List;

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 RowingBoat(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();
        }
    }

}
```

Person

```
package ki306.tymkiv.lab2;

public class Person {

    private String name;

    private int age;

    public Person(String name, int age) {        this.name =
name;        this.age = age;
    }

    public String getName() {        return name;
    }
```

```
    public void setName(String name) {        this.name = name;
    }

    public int getAge() {        return age;
    }

    public void setAge(int age) {        this.age = age;
    }

}
```

RowingBoat package ki305.tsipotan.lab2;

```
import java.io.File; import
java.io.FileNotFoundException; import
java.io.PrintWriter; import java.util.ArrayList; import
java.util.List;

/**
 *  The RowingBoat class models a rowing boat with an owner, a list of supplies,
 *  and a location. It also logs each action performed using a
PrintWriter to  * a log
file.
 *
 *  This class supports adding/removing supplies, moving the boat,
```

```
 * and various utility methods for managing the boat and its contents. */ public class
    RowingBoat implements AutoCloseable {


   private Person owner;    private List<Supply> supplies;    private
Location location;    private PrintWriter fout = new PrintWriter(new
File("Log.txt"));


   /**
 * 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 RowingBoat(Person owner, List<Supply> supplies, Location location)
throws FileNotFoundException {        this.owner = owner;        this.supplies =
supplies;        this.location = location;
   }


   /**
 * Adds a supply to the boat and logs the action.
    *
 * @param supply The supply to be added.
    */    public void addSupply(Supply supply) {        supplies.add(supply);
fout.println("Added supply: " + supply.getName() + " (Quantity: " + supply.getQuantity()
+ ")");        fout.flush();
```

```
    }

    /**
     * Removes a supply from the boat by its name and logs the action.
     *
     * @param supplyName The name of the supply to be removed.
     */    public void removeSupply(String supplyName)
{        supplies.removeIf(supply -> supply.getName().equals(supplyName));
fout.println("Removed supply: " + supplyName);        fout.flush();
    }

    /**
     * Moves the boat to a new location and logs the move.
     *
     * @param newLocation The new location to move the boat to.
     */    public void moveToNewLocation(Location newLocation) {        fout.println("Moving
boat from " + location.getCity() + " to " + newLocation.getCity());        this.location =
newLocation;        fout.println("Boat moved to new location: " + newLocation.getCountry() + ",
" + newLocation.getCity() + ", " + newLocation.getStreet());        fout.flush();
    }

    /**
     * Gets the total number of supplies on the boat and logs the result.
     *
     * @return The total number of supplies on the boat.
```

```
    */    public int getTotalSupplies() {        int total = supplies.size();
fout.println("Total supplies on board: " + total);        fout.flush();        return total;


    }

    /**
*  Gets the owner's name and logs the action.
     *
*  @return The name of the owner.
     */    public String getOwnerName() {        fout.println("Getting
owner's name: " + owner.getName());        fout.flush();        return
owner.getName();
    }

    /**
*  Changes the owner of the boat and logs the change.
     *
*  @param newOwner The new owner of the boat.
     */    public void changeOwner(Person newOwner)
{        fout.println("Changing boat owner from " + owner.getName() + " to "
+ newOwner.getName());        this.owner = newOwner;
fout.println("Owner changed successfully.");        fout.flush();
    }

    /**
```

```
* Gets the current location of the boat and logs the action.
     *
* @return The current location of the boat.
     */     public Location getCurrentLocation() {         fout.println("Current location: " +
location.getCountry() + ", " + location.getCity() + ", " + location.getStreet());
fout.flush();         return location;
    }

    /**
* Checks if a supply exists on the boat by its name and logs the result.
     *
* @param supplyName The name of the supply to check.
* @return true if the supply exists, false otherwise.
     */     public boolean hasSupply(String supplyName) {         boolean exists =
supplies.stream().anyMatch(supply -> supply.getName().equals(supplyName));
fout.println("Checking if supply '" + supplyName + "' exists: " + (exists ? "Yes" : "No"));

        fout.flush();         return exists;
    }

    /**
* Counts the total quantity of all supplies on the boat and logs the result.
     *
* @return The total quantity of all supplies on the boat.
```

```
    */    public int getTotalSupplyQuantity() {        int totalQuantity =
supplies.stream().mapToInt(Supply::getQuantity).sum();        fout.println("Total
quantity of all supplies: " + totalQuantity);        fout.flush();        return
totalQuantity;
   }

   /**
* Prints a summary of the boat, including the owner, location, and supplies,      * and logs the
    summary.
    */    public void printBoatSummary()
{        fout.println("Boat Summary: ");
fout.println("Owner: " + owner.getName() + ", Age: " +
owner.getAge());        fout.println("Location: " +
location.getCountry() + ", " + location.getCity() + ", " +
location.getStreet());        fout.println("Supplies:");
for (Supply supply : supplies)
{            fout.println(supply.getName() + " (Quantity: " +
supply.getQuantity() + ")");
      }        fout.flush();
   }

   /**
* Closes the PrintWriter used for logging.
    *
* @throws Exception if an I/O error occurs.
```

```
    */    @Override    public void close() throws Exception
{       fout.close();
    }

    @Override    public String toString()
{       return "RowingBoat{" +
"owner=" + owner +            ", staff=" +
supplies +

          ", location=" + location +
          ", fout=" + fout +
          '}';
    }
}
```

Supply

```
package ki306.tymkiv.lab2;

public class Supply {

    private int quantity;

    private String name;

    public int getQuantity() {        return quantity;
```

```
    }

    public void setQuantity(int quantity) {        this.quantity = quantity;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {        this.name = name;
    }

    public Supply(int quantity, String name) {        this.quantity =
quantity;        this.name = name;
    }

}
```

Висновок: Я ознайомилася з процесом розробки класів та пакетів мовою Java.