Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

Лабораторна робота № 2

“ ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ JAVA”

з курсу “Кросплатформні засоби програмування”

Варіант: 4

Виконав:

ст.гр.КІ-205

Воробець Тетяна

Прийняв:

доцент кафедри ЕОМ

Олексів М.В.

Львів 2024

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

Теоретичні відомості: Мова Java є повністю об'єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об'єктно-орієнтованих парадигм програмування, що базуються на понятті класів.

Метод – функція-член класу, яка призначена маніпулювати станом об'єкту класу. Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом вказування різної кількості параметрів та їх типів методам з однаковими назвами.

Конструктор – спеціальний метод класу, який не повертає значення, має ім'я класу та призначений для початкової ініціалізації об'єктів класу.

Поле (властивість) – це дані-члени класу, що призначені для зберігання стану об'єкту. Поле може бути статичним (в цьому випадку воно називається полем класу), незмінним (константне поле), простим типом чи об'єктом та мати різні рівні доступу, що визначаються специфікатором доступу.

Пакет – це механізм мови Java, що дозволяє об'єднувати класи в простори імен. Об'єднання класів в пакети дозволяє відділяти класи, що розроблені одними розробниками, від класів, що розроблені іншими розробниками, забезпечуючи тим самим унікальність імен класів в межах програми та усуваючи можливі конфлікти імен класів. Пакети можуть бути вкладеними одні в одних, утворюючи цим самим ієрархії пакетів.

Створення пакетів відбувається за допомогою оператора package з вказуванням назв пакету і під пакетів (за необхідності), що розділені крапкою. Оператор package вказується на початку тексту програми перед операторами import та визначенням класу.

Завдання: 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;
• клас має містити мінімум 3 поля, що є об'єктами класів, які описують складові частини предметної області;
• клас має містити кілька конструкторів та мінімум 10 методів;
• для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
• методи класу мають вести протокол своєї діяльності, що записується у файл;
• розробити механізм коректного завершення роботи з файлом (не надіятися на метод finalize());
• програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.
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

Код програми:

Cat.java

```
/**
 *
 */
package KI_305.Vorobets.Lab2;

import java.io.FileWriter;
import java.io.IOException;
/**
 * Class <code>Cat</code> implements cat
 */
public class Cat {

     private String name;
    private Breed breed;
    private Master number, nameMaster;
    private Collar collar;
    private int energy;
    private int food;
    private String location;


    /**
     * Constructor
     */
    public Cat() {
        this.collar = new Collar();
        this.breed = new Breed();
        this.number = new Master();
        this.nameMaster = new Master();
        logActivity("Cat's name: None "
                + "Cat's bread: None"+
                "Cat's master: None, None"+
                "Cat's colar: None");
    }

    /**
     * Constructor
     * @param <code>name</code> name cats
     * @param <code>breed</code> breed cats
     * @param <code>number</code> number master
     * @param <code>nameMaster</code> name master
     * @param <code>info_collar</code> info_collar
     */
```

```
    public Cat(String name, String breed, String number, String
nameMaster, boolean info_collar, String location) {
     this();
     this.name = name;
     this.energy = 5;
     this.food = 5;
     this.location = location;
     this.collar.setPresence(info_collar);
        this.breed.setBreed(breed);
        this.number.setNumber(number);
        this.nameMaster.setNameMaster(nameMaster);
        logActivity("Cat's name: "+name
                + ", Cat's bread: "+ breed +
                ", Number master: "+number + ", Name master:
"+nameMaster+
                ", Cat's colar: " + info_collar);
    }

    /**
    * Method gets the name cat
    */

    public String getName() {
        return name;
    }

    /**
    * Method sets the name cat
    * @param <code>name</code> name cat
    */

    public void setName(String name) {
        this.name = name;
        logActivity("New name for cat's: " + name);
    }

    /**
    * Method action cat play
    * @param <code>game</code> game for cat
    */

    public void play(String game) {
     if (game == "mouse") {
           this.energy = this.energy - 1;
           this.food = this.food + 1;
            logActivity(name+ " plays with mouse: energy-1, food+1");
     }
     else if(game == "bug") {
           this.energy = this.energy + 1;
           this.food = this.food - 1;
           this.location = "bugs";
           logActivity(name+ " plays in bug: energy+1, food-1");
     }
```

```
    else {
          this.energy = this.energy - 2;
          this.food = this.food - 1;
          this.location = "outside";
          logActivity(name+ " plays in outside: energy-2, food-1");
    }
  }

  /**
  * Method action cat sleep
  */

  public void sleep() {
    this.energy = this.energy +1;
       logActivity(name + " sleeps: energy+1");
  }

  /**
  * Method action cat clean
  */

  public void clean() {
    this.energy = this.energy - 1;
       logActivity(name + " cleans: energy-1");
  }

  /**
  * Method action cat night_vision
  * @param <code>visor</code> on or off night vision
  */

  public void night_vision(boolean visor) {
    this.energy = this.energy - 1;
       logActivity(name + " night vision "+visor+": energy-1");
  }

  /**
  * Method action cat mew
  */

  public void mew() {
       logActivity(name + " said mew");
  }

  /**
  * Method location cat
  */

  public void setPlace() {
       logActivity(name + " in " + location);
  }
```

```
    /**
    * Method action cat eat
    */

    public void eat(String food) {
     if (food == "fish") {
           this.food = this.food + 1;
            logActivity(name+ " eats fish: food+1");
     }
     else if(food == "meat") {
           this.food = this.food + 2;
           logActivity(name+ " eats meat: food+2");
     }
     else {
           this.food = this.food - 1;
           logActivity(name+ " don't want to eat candy: food-1");
     }
    }

    /**
    * Method cat status
    */

    public void status() {
     if (this.energy == 5 && this.food == 5) {
           logActivity("Status: Nice");
     }
     else if (this.energy < 5 || this.food < 5) {
           logActivity("Status: "+name+" need to sleep or eat");
     }
     else {
           logActivity("Status: "+name+" want to play");
     }
     logActivity("Level energy: " + this.energy+ ", Level food: " +
this.food);
    }


    /**
    * Method information about cat
    */

    public void info_cat() {
        logActivity("Cat's name: "+name
                + ", Cat's bread: "+ breed.getBreed() +
                ", Number master: "+ number.getNumber() + ", Name
master: "+nameMaster.getNameMaster()+
                ", Cat's colar: " + collar.getPresence() + ", Cat's
energy: " + energy);
    }

    /**
    * Method logging info
```

```
    */

    public void logActivity(String message) {
        try (FileWriter fw = new FileWriter("cat_activity.log", true)) {
            fw.write(message + "\n");
            System.out.println(message);
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}

/**
 *  Class describing a breed of cat's
 */
class Breed {
    private String breed;

    /**
    * Method gets the name of breed
    */

    public String getBreed() {
        return breed;
    }
    /**
    * Method sets the name of room
    * @param <code>breed</code> name of room
    */
    public void setBreed(String breed) {
        this.breed = breed;
    }
}

/**
 *  Class describing a collar of cat's
 */
class Collar {
    private boolean collar;

    /**
    * Method gets the presence of a collar
    */

    public boolean getPresence() {
        return collar;
    }
    /**
    * Method sets the presence of a collar
    * @param <code>collar</code> the presence of a collar
    */

    public void setPresence(boolean collar) {
```

```
        this.collar = collar;
    }
}

/**
 *  Class describing a master of cat's
 */
class Master {
    private String nameMaster;
    private String number;

    /**
    * Method sets the presence of a collar
    * @param <code>collar</code> the presence of a collar
    */

    public String getNameMaster() {
        return nameMaster;
    }

    /**
    * Method sets the presence of a collar
    * @param <code>collar</code> the presence of a collar
    */

    public void setNameMaster(String nameMaster) {
        this.nameMaster = nameMaster;
    }

    /**
    * Method gets the number master
    */

    public String getNumber() {
        return number;
    }

    /**
    * Method sets the number master
    * @param <code>number</code> number master
    */

    public void setNumber(String number) {
        this.number = number;
    }
}
```

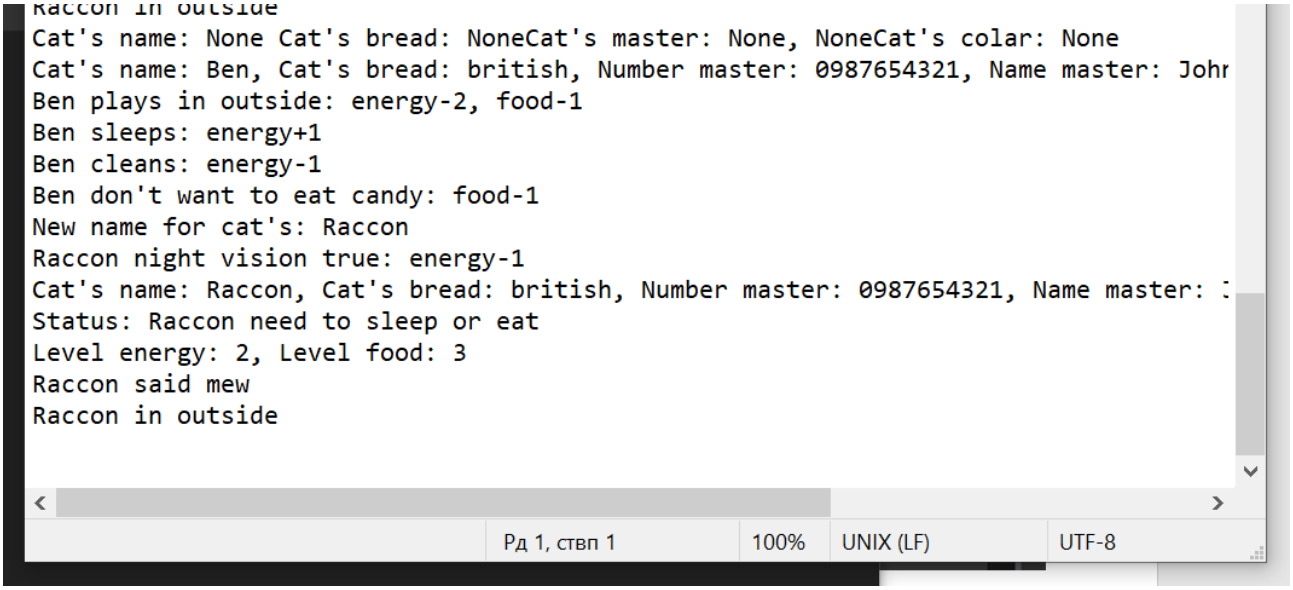
CatApp.java

```
package KI_305.Vorobets.Lab2;
import java.util.Random;

/**
 * Клас CatDriver для тестування класу Cat.
 */
```

```java
public class CatApp {

	/**
	 * @param args
	 */
    public static void main(String[] args) {
	  Random random = new Random();
	  int randomNumber = random.nextInt(3);
	    String food[] = {"fish", "meat", "candy"};
	    String game[] = {"mouse", "bug", "village"};


	    Cat myCat = new Cat("Ben", "british", "0987654321", "John", true,
"home");

	    myCat.play(game[randomNumber]);
	    myCat.sleep();
	    myCat.clean();
	    myCat.eat(food[randomNumber]);
	    myCat.setName("Raccon");
	    myCat.night_vision(true);
	    myCat.info_cat();
	    myCat.status();
	    myCat.mew();
	    myCat.setPlace();
    }
}
```

Результат:

```
<terminated> CatApp [Java Application] C:\Users\Tanya\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-
Cat's name: None Cat's bread: NoneCat's master: None, NoneCat's colar: None
Cat's name: Ben, Cat's bread: british, Number master: 0987654321, Name master: John, Ca
Ben plays in outside: energy-2, food-1
Ben sleeps: energy+1
Ben cleans: energy-1
Ben don't want to eat candy: food-1
New name for cat's: Raccon
Raccon night vision true: energy-1
Cat's name: Raccon, Cat's bread: british, Number master: 0987654321, Name master: John,
Status: Raccon need to sleep or eat
Level energy: 2, Level food: 3
Raccon said mew
Raccon in outside
```



Посилання на репозиторій:

https://github.com/NikaDe7/CPPT_Vorobets_TI_KI-35_1.git

Документація:

Constructor Summary

Constructors

Constructor	Description
`Cat()`	Constructor
`Cat(String name, String breed, String number, String nameMaster, boolean info_collar, String location)`	Constructor

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	`clean()`	Method action cat clean
void	`eat(String food)`	Method action cat eat
`String`	`getName()`	Method gets the name cat
void	`info_cat()`	Method information about cat
void	`logActivity(String message)`	Method logging info
void	`mew()`	Method action cat mew
void	`night_vision(boolean visor)`	Method action cat night_vision
void	`play(String game)`	Method action cat play
void	`setName(String name)`	Method sets the name cat
void	`setPlace()`	Method location cat

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