Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 3

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Класи та пакети»

**Виконав:**

студент групи КІ-35

Дзьобань М. Б.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2022

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

**Завдання (варіант № 3)**

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту.

***Варіант-*3: Пес**

Програма має задовольняти наступним вимогам:

* + програма має розміщуватися в пакеті Група.Прізвище.Lab3;
  + 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.

1. Автоматично згенерувати документацію до розробленого пакету.
2. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.

Дати відповідь на контрольні запитання.

**Текст програми**

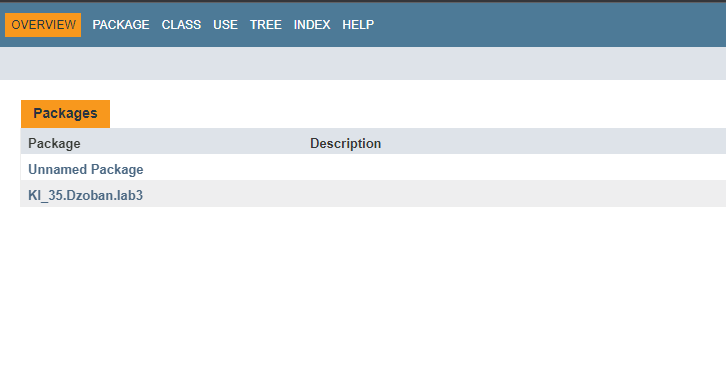
**DogApp.java**

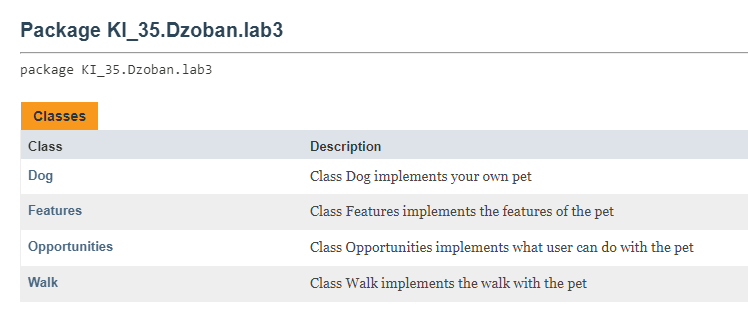
import KI\_35.Dzoban.lab3.\*;  
  
  
import java.io.\*;  
import static java.lang.System.\*;  
import java.util.Scanner;  
  
*/\*\* Class DogApp implements example of lab№3 program  
\*  
\** ***@author*** *Mykola Dzoban  
\** ***@version*** *1.0.1  
\** ***@since*** *version 1.0  
\*  
\*/*public class DogApp {  
 */\*\*  
 \** ***@param*** *args function parameter  
 \** ***@throws*** *FileNotFoundException check file existing  
 \*  
 \*/* public static void main(String[] args) throws FileNotFoundException  
 {  
 Dog dog = new Dog();  
 dog.makeDog(100, 50, 1);  
 Dog dog1 = new Dog();  
 Dog dog2 = new Dog();  
 Dog dog3 = new Dog();  
 Dog dog4 = new Dog();  
 dog1.makeDog(0, 0, 2);  
 dog2.makeDog(0, 0, 3);  
 dog3.makeDog(0, 0, 2);  
 dog4.makeDog(0, 0, 5);  
  
 *out*.println(Dog.*numOfOb* + "\n");  
 // int hp,en,age,age2,hun,dream,wash,wash2,food,hour,xPos;  
 // Scanner scanner = new Scanner(System.in);  
 // out.print("Make your own pet!\n");  
 // out.print("Please, enter your dog features:\n");  
 // out.print("Health: ");  
 // hp = scanner.nextInt();  
 // out.print("Energy: ");  
 // en = scanner.nextInt();  
 // out.print("Age: ");  
 // age = scanner.nextInt();  
 // Dog dog = new Dog();  
 // dog.makeDog(hp,en,age);  
 // dog.aboutDog();  
 // out.print("Please, enter your dog features:\n");  
 // out.print("Hunger: ");  
 // hun = scanner.nextInt();  
 // out.print("Dream: ");  
 // dream = scanner.nextInt();  
 // out.print("Purity: ");  
 // wash = scanner.nextInt();  
 // dog.oppDog(hun,dream,wash);  
 // out.print("Do you want to feed your dog? If yes - enter number greater than 0, if no - enter 0:");  
 // food = scanner.nextInt();  
 // dog.feedDog(food);  
 // out.print("Did your dog sleep? If yes - enter hours greater than 0, if no - enter 0:");  
 // hour = scanner.nextInt();  
 // dog.sleepDog(hour);  
 // out.print("Do you want to wash your dog? If yes - enter wash number greater than 0, if no - enter 0:");  
 // wash2 = scanner.nextInt();  
 // dog.washDog(wash2);  
 // dog.dogOpp();  
 // out.print("Do you want to make your dog elder? If yes - enter years greater than 0, if no - enter 0:");  
 // age2 = scanner.nextInt();  
 // dog.makeAgeBigger(age2);  
 // out.print("If you are in some position - enter coordinate greater than 0, if no - enter 0:");  
 // xPos = scanner.nextInt();  
 // dog.moveDog(xPos);  
 // out.print("Do you want to walk left with dog? If yes - enter coordinate greater than 0, if no - enter 0:");  
 // xPos = scanner.nextInt();  
 // dog.walkLeftWithDog(xPos);  
 // out.print("Do you want to walk right with dog? If yes - enter coordinate greater than 0, if no - enter 0:");  
 // xPos = scanner.nextInt();  
 // dog.walkRightWithDog(xPos);  
 // dog.getDogPosition();  
 // dog.dis();  
 }  
}

**Результат виконання програми**

Dog health: 50  
Dog energy: 45  
Dog age: 12  
Hunger: 32  
Dream: 5  
Purity: 10  
Dog's hunger: 32  
Dog's dream: 2  
Dog's purity: 22  
Dog grows: 13  
New dog position: 0  
Walk left position: -5  
Walk right position: 2

**Фрагмент згенерованої документації**





**Відповіді на контрольні запитання**

1. Синтаксис визначення класу.

[public] class НазваКласу

{

[конструктори] [методи] [поля]

}

1. Синтаксис визначення методу.

[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

[Тіло методу] [return [значення]];

}

1. Синтаксис оголошення поля.

[СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];

1. Як оголосити та ініціалізувати константне поле?

Приклад оголошення поля: private int i;

Приклад оголошення константного поля: private final int i;

1. Які є способи ініціалізації полів?
   * явно при оголошені поля класу;
   * у статичному блоці ініціалізації

**Висновок**

Під час виконання цієї лабораторної я ознайомився з процесом розробки класів та пакетів мовою Java.

**Додаток**

**Dog.java**

package KI\_35.Dzoban.lab3;  
import java.io.\*;  
import static java.lang.System.\*;  
*/\*\* Class Dog implements your own pet  
 \*  
 \** ***@author*** *Mykola Dzoban  
 \** ***@version*** *1.0.1  
 \** ***@since*** *version 1.0  
 \*  
 \*/*public class Dog {  
 public static char[] *noOfObjects*;  
 private Features fea;  
 private Opportunities opps;  
 private Walk walk;  
  
 private PrintWriter fout;  
  
 static public int *numOfOb* = 0;  
 */\*\*  
 \* Constructor  
 \** ***@throws*** *FileNotFoundException check file existing  
 \*/* public Dog() throws FileNotFoundException  
 {  
 fea = new Features();  
 opps = new Opportunities();  
 walk = new Walk();  
 fout = new PrintWriter(new File("Dog.txt"));  
 }  
  
 */\*\*  
 \* Method makeDog implements making your own pet  
 \** ***@param*** *hp The health value  
 \** ***@param*** *en The energy value  
 \** ***@param*** *age The age value  
 \*/* public void makeDog(int hp, int en, int age){  
 fea.setHealth(hp);  
 fea.setEnergy(en);  
 fea.setAge(age);  
 fout.print("Dog health: "+fea.getHealth()+"\n");  
 fout.print("Dog energy: "+fea.getEnergy()+"\n");  
 fout.print("Dog age: "+fea.getAge()+"\n");  
 fout.flush();  
 if (age > 3) {  
 ++*numOfOb*;  
 }  
 }  
  
 */\*\*  
 \* Method aboutDog shows information about dog  
 \*/* public void aboutDog(){  
 *out*.print("Dog health: "+fea.getHealth()+"\n");  
 *out*.print("Dog energy: "+fea.getEnergy()+"\n");  
 *out*.print("Dog age: "+fea.getAge()+"\n");  
 }  
  
 */\*\*  
 \* Method makeAgeBigger implements rising age of dog  
 \** ***@param*** *age  
 \*/* public void makeAgeBigger(int age){  
 fea.changeAgeToGrow(age);  
 *out*.print("Dog grows: "+fea.getAge()+"\n");  
 fout.print("Dog grows: "+fea.getAge()+"\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method oppDog implements dog's opportunities  
 \** ***@param*** *hun The hungry value  
 \** ***@param*** *dream The dream value  
 \** ***@param*** *wash The purity value  
 \*/* public void oppDog(int hun, int dream, int wash){  
 opps.setHunger(hun);  
 opps.setDream(dream);  
 opps.setWash(wash);  
 fout.print("Hunger: "+opps.getHunger()+"\n");  
 fout.print("Dream: "+opps.getDream()+"\n");  
 fout.print("Purity: "+opps.getWash()+"\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method dogOpp shows information about dog's opportunities  
 \*/* public void dogOpp(){  
 *out*.print("Hunger: "+opps.getHunger()+"\n");  
 *out*.print("Dream: "+opps.getDream()+"\n");  
 *out*.print("Purity: "+opps.getWash()+"\n");  
 }  
  
 */\*\*  
 \* Method feedDog implements the possibility to feed dog  
 \** ***@param*** *food The food value  
 \*/* public void feedDog(int food){  
 opps.Feed(food);  
 fout.print("Dog's hunger: " + opps.getHunger() + "\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method sleepDog implements the possibility to dog to sleep  
 \** ***@param*** *hour The hour of dream value  
 \*/* public void sleepDog(int hour){  
 opps.Sleep(hour);  
 fout.print("Dog's dream: " + opps.getDream() + "\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method washDog implements the possibility to make dog clear  
 \** ***@param*** *wash The purity value  
 \*/* public void washDog(int wash){  
 opps.Wash(wash);  
 fout.print("Dog's purity: " + opps.getWash() + "\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method moveDog implements dog's moving  
 \** ***@param*** *xPos The X coordinate value  
 \*/* public void moveDog(int xPos){  
 walk.setXCoordinate(xPos);  
 fout.print("New dog position: " + walk.getXCoordinate() + "\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method getDogPosition returns dog's position  
 \** ***@return*** *X coordinate value  
 \*/* public int getDogPosition(){  
 return walk.getXCoordinate();  
 }  
  
 */\*\*  
 \* Method walkLeftWithDog implements the possibility to move dog left  
 \** ***@param*** *xPos The X coordinate value  
 \*/* public void walkLeftWithDog(int xPos){  
 walk.walkLeft(xPos);  
 *out*.print("Walk left position: " + walk.getXCoordinate() + "\n");  
 fout.print("Walk left position: " + walk.getXCoordinate() + "\n");  
 fout.flush();  
 }  
 */\*\*  
 \* Method walkRightWithDog implements the possibility to move dog right  
 \** ***@param*** *xPos The X coordinate value  
 \*/* public void walkRightWithDog(int xPos){  
 walk.walkRight(xPos);  
 *out*.print("Walk right position: " + walk.getXCoordinate() + "\n");  
 fout.print("Walk right position: " + walk.getXCoordinate() + "\n");  
 fout.flush();  
 }  
  
 */\*\*  
 \* Method dogJump implements the possibility for dog to jump  
 \** ***@param*** *xPos The X coordinate value  
 \** ***@param*** *yPos The Y coordinate value  
 \*/* public void dogJump(int xPos, int yPos){  
 walk.jump(xPos,yPos);  
 *out*.print("Dog jump on: " + walk.getXCoordinate() + ", " + walk.getYCoordinate()+"\n");  
 fout.print("Dog jump on: " + walk.getXCoordinate() + ", " + walk.getYCoordinate()+"\n");  
 fout.flush();  
 }  
 */\*\*  
 \* Method close file  
 \*/* public void dis()  
 {  
 fout.close();  
 }  
}

**Features.java**

package KI\_35.Dzoban.lab3;  
*/\*\* Class Features implements the features of the pet  
 \*  
 \** ***@author*** *Mykola Dzoban  
 \** ***@version*** *1.0.1  
 \** ***@since*** *version 1.0  
 \*  
 \*/*public class Features {  
 //  
 private int health, energy, age;  
  
 */\*\*  
 \* Constructor  
 \*/* public Features(){  
 health = 0;  
 energy = 0;  
 age = 0;  
 }  
 */\*\*  
 \* Method returns the health value  
 \** ***@return*** *The health value  
 \*/* public int getHealth(){  
 return health;  
 }  
 */\*\*  
 \* Method returns the energy value  
 \** ***@return*** *The energy value  
 \*/* public int getEnergy(){  
 return energy;  
 }  
  
 */\*\*  
 \* Method returns the age value  
 \** ***@return*** *The age value  
 \*/* public int getAge() {  
 return age;  
 }  
 */\*\*  
 \* Method sets the health value  
 \** ***@param*** *hp The health value  
 \*/* public void setHealth(int hp){  
 this.health = hp;  
 }  
 */\*\*  
 \* Method sets the energy value  
 \** ***@param*** *energy The energy value  
 \*/* public void setEnergy(int energy) {  
 this.energy = energy;  
 }  
  
 */\*\*  
 \* Method sets the age value  
 \** ***@param*** *age The age value  
 \*/* public void setAge(int age) {  
 this.age = age;  
 }  
 */\*\*  
 \* Method sets the age value after growing  
 \** ***@param*** *age The age value  
 \*/* public void changeAgeToGrow(int age){  
 this.age += age;  
 }  
  
}

**Opportunities.java**

package KI\_35.Dzoban.lab3;  
import static java.lang.System.\*;  
*/\*\* Class Opportunities implements what user can do with the pet  
 \*  
 \** ***@author*** *Mykola Dzoban  
 \** ***@version*** *1.0.1  
 \** ***@since*** *version 1.0  
 \*  
 \*/*public class Opportunities {  
 private int dream,hunger,wash;  
 */\*\*  
 \* Constructor  
 \*/* public Opportunities(){  
 dream = 0;  
 hunger = 0;  
 wash = 0;  
 }  
 */\*\*  
 \* Method returns the hunger value  
 \** ***@return*** *The hunger value  
 \*/* public int getHunger() {  
 return hunger;  
 }  
 */\*\*  
 \* Method sets the hunger value  
 \** ***@param*** *hunger The hunger value  
 \*/* public void setHunger(int hunger) {  
 this.hunger = hunger;  
 }  
 */\*\*  
 \* Method sets the hunger value after feed  
 \** ***@param*** *food The food value  
 \*/* public void Feed(int food){  
 if ((this.hunger + food) <= 0){  
 *out*.print("Your dog is very hungry!!!\n");  
 }  
 else {  
 this.hunger += food;  
 }  
 }  
 */\*\*  
 \* Method returns the dream value  
 \** ***@return*** *The dream value  
 \*/* public int getDream() {  
 return dream;  
 }  
 */\*\*  
 \* Method sets the dream value  
 \** ***@param*** *dream The dream value  
 \*/* public void setDream(int dream) {  
 this.dream = dream;  
 }  
 */\*\*  
 \* Method sets the dream value after sleep  
 \** ***@param*** *hour The hour value  
 \*/* public void Sleep(int hour){  
 if (hour <= 0){  
 *out*.print("Your dog is exhausted!!!\n");  
 this.dream = hour;  
 }  
 else {  
 this.dream = hour;  
 }  
 }  
 */\*\*  
 \* Method returns the wash value  
 \** ***@return*** *The wash value  
 \*/* public int getWash() {  
 return wash;  
 }  
 */\*\*  
 \* Method sets the wash value  
 \** ***@param*** *wash The wash value  
 \*/* public void setWash(int wash) {  
 this.wash = wash;  
 }  
 */\*\*  
 \* Method sets the wash value after sleep  
 \** ***@param*** *wash The wash value  
 \*/* public void Wash(int wash){  
 if ((this.wash+wash) <= 0){  
 *out*.print("Your dog is very dirty!!!\n");  
 }  
 else {  
 this.wash += wash;  
 }  
 }  
}

**Walk.java**

package KI\_35.Dzoban.lab3;  
  
*/\*\* Class Walk implements the walk with the pet  
 \*  
 \** ***@author*** *Mykola Dzoban  
 \** ***@version*** *1.0.1  
 \** ***@since*** *version 1.0  
 \*  
 \*/*public class Walk {  
 private int x, y;  
 */\*\*  
 \* Constructor  
 \*/* public Walk(){  
 this.x = 0;  
 this.y = 0;  
 }  
 */\*\*  
 \* Method gets the X coordinate value  
 \** ***@return*** *The X coordinate value  
 \*/* public int getXCoordinate() {  
 return this.x;  
 }  
 */\*\*  
 \* Method sets the X coordinate value  
 \** ***@param*** *x The X coordinate value  
 \*/* public void setXCoordinate(int x) {  
 this.x = x;  
 }  
 */\*\*  
 \* Method gets the Y coordinate value  
 \** ***@return*** *The Y coordinate value  
 \*/* public int getYCoordinate() {  
 return this.y;  
 }  
 */\*\*  
 \* Method sets the X coordinate value after walk left  
 \** ***@param*** *X The X coordinate value  
 \*/* public void walkLeft(int X){  
 this.x -= X;  
 }  
 */\*\*  
 \* Method sets the X coordinate value after walk right  
 \** ***@param*** *X The X coordinate value  
 \*/* public void walkRight(int X){  
 this.x += X;  
 }  
 */\*\*  
 \* Method return the X and Y coordinates value to jump  
 \** ***@param*** *x The X coordinate value  
 \** ***@param*** *y The Y coordinate value  
 \*/* public void jump(int x, int y){  
 this.x=x;  
 this.y=y;  
 }  
}