МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ» Кафедра «Програмна інженерія та інформаційні технології управління» Звіт з лабораторної роботи №6

З предмету «Алгоритмізація та програмування»

Виконав

Студент групи КН-36а

Рубан Ю.Д.

Перевірив:

Ст. в. Смолін П. О.

Харків 2017

**Використання успадкування та поліморфізму в Java**

**1 Завдання на лабораторну роботу**

**1.1 Індивідуальне завдання**

Внести у код і функціональність класів, які були створені в [попередній лабораторній роботі](http://iwanoff.inf.ua/algorithmization_2/LabTraining05.html), такі зміни:

* додати перевизначення функцій toString() і використовувати їх для виведення даних про об'єкти
* додати перевизначення методів equals() для перевірки еквівалентності об'єктів
* реалізувати функцію додавання об'єкта до масиву з перевіркою, чи такий елемент вже присутній у масиві
* змінити функції пошуку таким чином, щоб вони повертали масиви об'єктів (або **null**, якщо пошук не дав результатів), замість того, щоб безпосередньо виводити ці результати
* додати функції сортування за визначеними ознаками.

Тестування програми повинно включати виконання завдання попередньої лабораторної роботи, а також сортування за визначеними ознаками. Для сортування слід використовувати метод sort() класу Arrays. Ознаки сортування визначаються у залежності від номеру студента у списку групи. Одне з сортувань повинне бути забезпечене реалізацією інтерфейсу Comparable для сутності, об'єкти якої зберігаються в масиві. Друге сортування забезпечується створенням окремого класу, який реалізує інтерфейс Comparator.



**1.2 Ієрархія класів**

Реалізувати класи "Людина", "Громадянин", "Студент", "Співробітник". Створити масив посилань на різні об'єкти ієрархії. Для кожного об'єкта вивести на екран рядок даних про нього.

**1.3 Мінімум функції**

Реалізувати програму, що дозволяє знайти мінімум деякої функції на заданому інтервалі. Алгоритм знаходження мінімуму полягає в послідовному переборі з певним кроком точок інтервалу і порівнянні значень функції в поточній точці з раніше знайденим мінімумом.

Реалізувати два підходи - через використання абстрактних класів і через використання інтерфейсів.

**1.4 Реалізація масиву точок через двовимірний масив**

Реалізувати функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі 3.2, через використання двовимірного масиву дійсних чисел. Кожен рядок масиву має відповідати точці. Здійснити тестування класу.

**1.5 Реалізація масиву точок через одновимірний масив дійсних чисел**

Реалізувати функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі 3.2, через використання одновимірного масиву дійсних чисел. Кожна пара чисел у масиві має відповідати точці.

**Хід виконання роботи**

Введено у код і функціональність класів, які були створені в [попередній лабораторній роботі](http://iwanoff.inf.ua/algorithmization_2/LabTraining05.html), такі зміни:

* додано перевизначення функцій toString() і використано їх для виведення даних про об'єкти
* додано перевизначення методів equals() для перевірки еквівалентності об'єктів
* реалізовано функцію додавання об'єкта до масиву з перевіркою, чи такий елемент вже присутній у масиві
* змінено функції пошуку таким чином, щоб вони повертали масиви об'єктів (або **null**, якщо пошук не дав результатів), замість того, щоб безпосередньо виводити ці результати
* додано функції сортування за визначеними ознаками.

**Код програми:**

Клас Main

import java.util.\*;  
public class Main {  
 public static void main(String[] args)  
 {  
 ArrayList<Day> days =new ArrayList<Day>();  
 Day d1=new Day(30,"день 1");  
 days.add(d1);  
 Day d2=new Day(25 ,"на след день получился большой коментарий");  
 Day d3=new Day(26 ,"ААААА");  
 //days.add(d2);  
 Show s=new Show("выставка","автор",days);  
 s.pushDay(d2);  
 s.pushDay(d3);  
 Collections.*sort*(s.days(),new Comp());  
 System.*out*.println(s);  
 }  
}

Клас Show

import java.util.\*;  
public class Show  
{  
 Show(){};  
 Show(String t,String n,ArrayList<Day> d)  
 {  
 this.title = t;  
 this.name = n;  
 this.D=d;  
 }  
 void pushDay(Day d)  
 {  
 boolean check = false;  
 for (Day t:D)  
 {  
 if(t.equals(d))  
 {  
 check = true;  
 }  
 }  
 if(!check)  
 {  
 D.add(D.size(), d);  
 }  
 }  
 @Override  
 public String toString() {  
 return "Show{" +  
 "Days=" + D +  
 ", title='" + title + '\'' +  
 ", name='" + name + '\'' +  
 '}';  
 }  
 boolean equals(Show obj)  
 {  
 if(name == obj.name && title == obj.title)  
 {  
 return true;  
 }  
 else  
 {  
 return false;  
 }  
 }  
  
 ArrayList<Day> days()  
 {  
 return D;  
 }  
 ArrayList<Integer> visitors(Day o)  
 {  
 ArrayList<Integer>res = new ArrayList<Integer>();  
 for (Day d:D)  
 {  
 if(o.equals(d))  
 {  
 res.add(res.size(), d.getCountOfvisitors());  
 }  
 }  
 return res;  
 }  
 ArrayList<String> comments(Day o)  
 {  
 ArrayList<String>S = new ArrayList<String>();  
 for (Day d:D)  
 {  
 if(o.equals(d))  
 {  
 S.add(S.size(), d.getComment());  
 }  
 }  
 return S;  
 }  
 void setName(String n){name = n;}  
 String getName(){return name;}  
 void setTitle(String t){title = t;}  
 String getTitle(){return title;}  
 private String title;  
 private String name;  
 private ArrayList <Day> D = new ArrayList <Day>();  
}

Клас Day

public class Day implements Comparable<Day>  
{  
 Day(int count,String c)  
 {  
 this.comment=c;  
 this.countOfvisitors=count;  
 }  
  
 @Override  
 public String toString() {  
 return "Day{" +  
 "countOfvisitors=" + countOfvisitors +  
 ", comment='" + comment + '\'' +  
 '}';  
 }  
 boolean equals(Day obj)  
 {  
 if(countOfvisitors == obj.countOfvisitors && comment == obj.comment)  
 {  
 return true;  
 }  
 else  
 {  
 return false;  
 }  
 }  
  
 public void setComment(String c) {  
 this.comment = c;  
 }  
  
 public void setCountOfvisitors(int count) {  
 this.countOfvisitors = count;  
 }  
  
 public int getCountOfvisitors() {  
 return this.countOfvisitors;  
 }  
  
 public String getComment() {  
 return this.comment;  
 }  
  
 private int countOfvisitors;  
 private String comment;  
  
 @Override  
 public int compareTo(Day o) {  
 return Integer.*compare*(o.getCountOfvisitors(),this.getCountOfvisitors());  
 }  
}

Клас Comp

import java.util.ArrayList;  
import java.util.Comparator;  
  
public class Comp implements Comparator<Day>  
{  
 @Override  
 public int compare(Day o1, Day o2) {  
 return o1.getComment().compareTo(o2.getComment());  
 }  
}

Результат виконання програми:

Show{Days=[Day{countOfvisitors=26, comment='ААААА'}, Day{countOfvisitors=30, comment='день 1'}, Day{countOfvisitors=25, comment='на след день получился большой коментарий'}], title='выставка', name='автор'}

### 1.2 Ієрархія класів

Реалізовано класи "Людина", "Громадянин", "Студент", "Співробітник". Створено масив посилань на різні об'єкти ієрархії. Для кожного об'єкта виведено на екран рядок даних про нього.

Код програми:

Клас Main

public class Main {  
 public static void main(String[] args) {  
 Human[] h = new Human[]  
 {  
 new Student("lev","vel",154325,321),  
 new Citizen("ambal", "the killer", 007),  
 new Coworker("fiona", "sherowna",999, "ambal"),  
 new Student("pavlo", "zibrow",777,3432)  
 };  
 for (Human d:h)  
 {  
 System.*out*.println(d);  
 }  
 }  
}

Клас Human

public class Human {  
 private String name;  
 private String surname;  
 public Human(String name, String surname)  
 {  
 this.name = name;  
 this.surname = surname;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 @Override  
 public String toString() {  
 return "Human{" +  
 "name='" + name + '\'' +  
 ", surname='" + surname + '\'' +  
 '}';  
 }  
}

Клас Citizen

public class Citizen extends Human  
{  
 private int passport;  
 public Citizen(String name, String surname, int passport) {  
 super(name, surname);  
 this.passport = passport;  
 }  
  
 public int getPassport() {  
 return passport;  
 }  
  
 @Override  
 public String toString() {  
 return "Citizen{" +"name='" + this.getName() + '\'' +  
 ", surname='"+this.getSurname()+'\''+  
 ", passport='"+this.getPassport()+'\''+  
 "}";  
 }  
}

Клас Student

public class Student extends Citizen  
{  
 private int book;  
 public Student(String name, String surname, int passport, int book) {  
 super(name, surname, passport);  
 this.book = book;  
 }  
  
 @Override  
 public String toString() {  
 return "Student{" +"name='" + this.getName() + '\'' +  
 ", surname='"+this.getSurname()+'\''+  
 ", passport='"+this.getPassport()+'\''+  
 ", book='"+this.book+'\''+"}";  
 }  
}

Клас Coworker

public class Coworker extends Citizen  
{  
 private String profession;  
 public Coworker(String name, String surname, int passport, String profession) {  
 super(name, surname, passport);  
 this.profession = profession;  
 }  
  
 @Override  
 public String toString() {  
 return "Coworker{" +  
 "name='" + this.getName() + '\'' +  
 ", surname='"+this.getSurname()+'\''+  
 ", passport='"+this.getPassport()+'\''+  
 ", profession='"+this.profession+'\''+"}";  
 }  
}

Результат виконання програми:

Student{name='lev', surname='vel', passport='154325', book='321'}

Citizen{name='ambal', surname='the killer', passport='7'}

Coworker{name='fiona', surname='sherowna', passport='999', profession='ambal'}

Student{name='pavlo', surname='zibrow', passport='777', book='3432'}

### 1.3 Мінімум функції

Реалізовано програму, що дозволяє знайти мінімум деякої функції на заданому інтервалі. Алгоритм знаходження мінімуму полягає в послідовному переборі з певним кроком точок інтервалу і порівнянні значень функції в поточній точці з раніше знайденим мінімумом.

Реалізувано два підходи - через використання абстрактних класів і через використання інтерфейсів.

Код програми:

Клас Main

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Funk obj = new Funk();  
 Scanner sc = new Scanner(System.*in*);  
 double a =sc.nextDouble();  
 double b = sc.nextDouble();  
 double min=obj.f(a);  
 for(double i = a;i<=b;i+=0.001)  
 {  
 if(min>obj.f(i))  
 {  
 min = obj.f(i);  
 }  
 }  
 System.*out*.println("minimum interface funk="+min);  
 a = sc.nextDouble();  
 b = sc.nextDouble();  
 FunkFromAbs y = new FunkFromAbs();  
 min = y.f(a);  
 for(double i = a;i<=b;i+=0.001)  
 {  
 if(min>y.f(i))  
 {  
 min=y.f(i);  
 }  
 }  
 System.*out*.println("minimum abs funk="+min);  
 }  
}

Інтерфейс InterfaceFunk

public interface InterfaceFunk  
{  
 double f(double x);  
}

Клас Funk

public class Funk implements InterfaceFunk{  
 public double f(double x){return Math.*tan*(x)\*x\*x\*x/3\*Math.*log*(x)\*Math.*sinh*(x);}  
}

Абстрактний клас AbsFunk

public abstract class AbsFunk {  
 public abstract double f(double x);  
}

Клас FunkFromAbs

public class FunkFromAbs extends AbsFunk {  
  
 @Override  
 public double f(double x) {  
 return x\*x - 2;  
 }  
}

Результат виконання програми:

3 5

minimum interface funk=-4930641.211303169

2 6

minimum abs funk=2.0

### 1.4 Реалізація масиву точок через двовимірний масив

Реалізувано функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі 3.2, через використання двовимірного масиву дійсних чисел. Кожен рядок масиву відповідає точці

### 1.5 Реалізація масиву точок через одновимірний масив дійсних чисел

Реалізувано функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі 3.2, через використання одновимірного масиву дійсних чисел. Кожна пара чисел у масиві відповідає точці.

Код програми:

Абстрактний клас

public abstract class AbstractArrayOfPoints {  
 // Запис нових координат точки:  
 public abstract void setPoint(int i, double x, double y);  
  
 // Отримання X точки i:  
 public abstract double getX(int i);  
  
 // Отримання Y точки i:  
 public abstract double getY(int i);  
  
 // Отримання кількості точок:  
 public abstract int count();  
  
 // Додавання точки в кінець масиву:  
 public abstract void addPoint(double x, double y);  
  
 // Видалення останньої точки:  
 public abstract void removeLast();  
  
 // Сортування за значеннями X:  
 public void sortByX() {  
 boolean mustSort; // Повторюємо доти,  
 // доки mustSort дорівнює true  
 do {  
 mustSort = false;  
 for (int i = 0; i < count() - 1; i++) {  
 if (getX(i) > getX(i + 1)) {  
 // обмінюємо елементи місцями  
 double x = getX(i);  
 double y = getY(i);  
 setPoint(i, getX(i + 1), getY(i + 1));  
 setPoint(i + 1, x, y);  
 mustSort = true;  
 }  
 }  
 }  
 while (mustSort);  
 }  
}

Реалізация за допомогою двохмірного масиву

import java.util.ArrayList;  
public class ArrayOfPoints extends AbstractArrayOfPoints {  
 private int size;  
 private ArrayList<ArrayList<Double>>array;  
 ArrayOfPoints()  
 {  
 array = new ArrayList<ArrayList<Double>>(size);  
 for(int i =0;i<size;i++)  
 {  
 array.add(i,new ArrayList<Double>(2));  
 }  
 }  
 @Override  
 public void setPoint(int i, double x, double y) {  
 array.get(i).set(0,x);  
 array.get(i).set(1,y);  
 }  
  
 @Override  
 public double getX(int i) {  
 return array.get(i).get(0);  
 }  
  
 @Override  
 public double getY(int i) {  
 return array.get(i).get(1);  
 }  
  
 @Override  
 public int count() {  
 return size;  
 }  
  
  
 @Override  
 public void addPoint(double x, double y)  
 {  
 ArrayList<Double>temp = new ArrayList<Double>(2);  
 temp.add(0,x);  
 temp.add(1,y);  
 array.add(temp);  
 size++;  
 }  
  
 public String toString(int i) {  
 return "("+array.get(i).get(0)+", "+array.get(i).get(1)+")";  
 }  
  
 @Override  
 public void removeLast()  
 {  
 array.remove(size-1);  
 size--;  
 }  
}

Реалізація за допомогою одновимірного масиву

import java.util.ArrayList;  
  
public class ArrayOfPoints extends AbstractArrayOfPoints{  
 private ArrayList<Double>array;  
 ArrayOfPoints()  
 {  
 array = new ArrayList<Double>();  
 }  
 @Override  
 public void setPoint(int i, double x, double y)  
 {  
 array.set(i\*2,x);  
 array.set(i\*2+1,y);  
 }  
 public String toString(int i)  
 {  
 return "("+array.get(i\*2)+", "+array.get(i\*2+1)+")";  
 }  
 @Override  
 public double getX(int i) {  
 return array.get(i\*2);  
 }  
  
 @Override  
 public double getY(int i) {  
 return array.get(i\*2+1);  
 }  
  
 @Override  
 public int count() {  
 return array.size()/2;  
 }  
  
 @Override  
 public void addPoint(double x, double y) {  
 array.add(x);  
 array.add(y);  
 }  
  
 @Override  
 public void removeLast() {  
 array.remove(array.size()-1);  
 array.remove(array.size()-1);  
  
 }  
}

**Висновки**

У даній лабораторній роботі я навчився використовувати поліморфізм у java. Дізнався про абстрактні класи, інтерфейси та наслідування.