Міністерство освіти і науки України

Національний Технічний Університет

«Харківський Політехнічний Інститут»

Кафедра «Стратегічного Управління»

Лабораторна робота № 6

«Використання успадкування та поліморфізму в Java»

Перевірила: ас. кафедри СК

Вільхівская О. В.

Виконав:

Харків 2018

Лабораторна робота № 6

Тема: Використання успадкування та поліморфізму в Java

Завдання

1. Індивідуальне завдання

Внести у код і функціональність класів, які були створені в попередній лабораторній роботі, 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.

Таблиця 1.1 - Індивідуальні завдання

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №№ | Перша ознака | Друга ознака | №№ | Перша ознака | Друга ознака |
| 2 | За збільшенням кількості студентів | За збільшенням довжини теми | 17 | За номером групи | За зменшенням кількості студентів |

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.

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.

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.

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 роботи**

1. Індивідуальне завдання

Клас Topic

|  |
| --- |
| package task1;  import java.util.Arrays;  import java.util.StringTokenizer;  public class Topic implements Comparable<Topic> {  private int[] date;  private String subject;  private int students;  public Topic(int[] date, String subject, int students) {  this.date = date;  this.subject = subject;  this.students = students;  }  public int[] getDate() {  return date;  }  public void setDate(int[] date) {  this.date = date;  }  public String getSubject() {  return subject;  }  public void setSubject(String subject) {  this.subject = subject;  }  public int getAmount() {  return students;  }  public void setAmount(int amount) {  this.students = amount;  }  @Override  public boolean equals(Object obj) {  if (this == obj)  return true;  if (obj==null || !(obj instanceof Topic))  return false;  Topic test = (Topic) obj;  return Arrays.equals(test.getDate(), (getDate())) &&  test.getAmount() == getAmount() &&  test.getSubject() == getSubject();    }  @Override  public int compareTo(Topic p) {  return Integer.compare(getAmount(),p.getAmount());  }  @Override  public String toString() { //overloading toString() method  return ("Date: " + date[0] + "." + date[1] + "." + date[2] + " Subject: " + subject + " Students: " + students);  }    public boolean containsWord(String word) {  StringTokenizer st = new StringTokenizer(subject);  String s;  while (st.hasMoreTokens()) {  s = st.nextToken();  if (s.toLowerCase().equals(word.toLowerCase())) {  return true;  }  }  return false;  }  } |

Клас Subject

|  |
| --- |
| package task1;  import java.util.StringTokenizer;  import java.util.Arrays;  import java.util.Comparator;  public class Subject {  private Topic[] Practices;  public Subject(Topic[] practices, String name, boolean exams) {  super();  Practices = practices;  this.name = name;  this.exams = exams;  }  private String name;  private boolean exams;  public double getAverage() {  double average=0;  for (Topic practice: Practices) {  average+=practice.getAmount();  }  return average/=Practices.length;  }  public Topic getMax() {  Topic practice = Practices[0];  for (int i=1; i<Practices.length; i++) {  if(Practices[i].getAmount()>practice.getAmount()) {  practice = Practices[i];  }  }  return practice;  }  public Topic[] getPractices() {  return Practices;  }  public static Topic[] addToArray(Topic[] arr, Topic item) {  Topic[] newArr;  if (arr!=null) {  newArr = new Topic[arr.length+1];  System.arraycopy(arr, 0, newArr, 0, arr.length);  }  else {  newArr = new Topic[1];  }  newArr[newArr.length-1]=item;  return newArr;  }  public void setPractices(Topic[] arr) {  this.Practices = arr;  }  public boolean addPractice(Topic practice) {  if (getPractices()!=null) {  for (Topic p : Practices) {  if (p.equals(practice)) {  return false;  }  }  }  setPractices(addToArray(getPractices(), practice));  return true;  }  public Topic[] findWord(String word) {  System.out.println("Info about practice containing word " + word);  Topic[] arr = null;  for (Topic practice: Practices) {  if (practice.containsWord(word)) {  addToArray(arr, practice);  System.out.println(practice);  }  }  return arr;  }  public void sortByStudents() {  Arrays.sort(Practices);  }  public void sortByTopic(){  Arrays.sort(Practices, new CompareByTopic());  }  class CompareByTopic implements Comparator<Topic>{  public int compare(Topic p1, Topic p2) {  return Integer.compare(p1.getSubject().length(),p2.getSubject().length());  }  }  public int practicesCount() {  return Practices == null ? 0 : Practices.length;  }  public Topic getPractice(int i) {  return Practices[i];  }  @Override  public String toString() { //overloading toString() method  String result = "Name: " + name + " Exams: " + exams + " Amount of practices: " + Practices.length;  for (int i = 0; i < practicesCount(); i++) {  result += "\n" + getPractice(i);  }  return result;  }  public static void main(String[] args) {  Topic[] Practices = {  new Topic(new int []{10,1,2018}, "Deep First Search", 5),  new Topic(new int[]{17,1,2018}, "Bredth First Search", 6),  };  Subject sub = new Subject(Practices, "Algorithms & Data Structures", true);    System.out.println("Students average amount: " + sub.getAverage());  System.out.println("Max amount of students practice: " + sub.getMax().getAmount() + " (" + sub.getMax().getSubject() + " topic)");  sub.findWord("Search");    System.out.println("\nInfo about Subject object: \n" + sub);    System.out.println("\nAdding new practices to Algorithms & Data Structures");  System.out.println(sub.addPractice(new Topic(new int[]{24,1,2018}, "Gift Wrapping Algorithm", 8)));  System.out.println(sub.addPractice(new Topic(new int[]{31,1,2018}, "Pseudorandom generators", 3)));  System.out.println(sub.addPractice(new Topic(new int[]{7,2,2018}, "Dynamic Programming", 8)));  System.out.println(sub.addPractice(new Topic(new int[]{7,2,2018}, "Dynamic Programming", 8)));    sub.sortByStudents();  System.out.println("\nSorting by students amount: \n" + sub);  sub.sortByTopic();  System.out.println("\nSorting by the length of a topic name: \n" + sub);  }  } |

Приклад виконання програми:

|  |
| --- |
| Students average amount: 5.5  Max amount of students practice: 6 (Bredth First Search topic)  Info about practice containing word Search  Date: 10.1.2018 Subject: Deep First Search Students: 5  Date: 17.1.2018 Subject: Bredth First Search Students: 6  Info about Subject object:  Name: Algorithms & Data Structures Exams: true Amount of practices: 2  Date: 10.1.2018 Subject: Deep First Search Students: 5  Date: 17.1.2018 Subject: Bredth First Search Students: 6  Adding new practices to Algorithms & Data Structures  true  true  true  false  Sorting by students amount:  Name: Algorithms & Data Structures Exams: true Amount of practices: 5  Date: 31.1.2018 Subject: Pseudorandom generators Students: 3  Date: 10.1.2018 Subject: Deep First Search Students: 5  Date: 17.1.2018 Subject: Bredth First Search Students: 6  Date: 24.1.2018 Subject: Gift Wrapping Algorithm Students: 8  Date: 7.2.2018 Subject: Dynamic Programming Students: 8  Sorting by the length of a topic name:  Name: Algorithms & Data Structures Exams: true Amount of practices: 5  Date: 10.1.2018 Subject: Deep First Search Students: 5  Date: 17.1.2018 Subject: Bredth First Search Students: 6  Date: 7.2.2018 Subject: Dynamic Programming Students: 8  Date: 31.1.2018 Subject: Pseudorandom generators Students: 3  Date: 24.1.2018 Subject: Gift Wrapping Algorithm Students: 8 |

2. Ієрархія класів

|  |
| --- |
| **package** task2;  **class** Human {  **private** String name;  **private** String surname;  **private** **int** age;    **public** Human(String name, String surname, **int** age) {  **super**();  **this**.name = name;  **this**.surname = surname;  **this**.age = age;  }  **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getSurname() {  **return** surname;  }  **public** **void** setSurname(String surname) {  **this**.surname = surname;  }  **public** **int** getAge() {  **return** age;  }  **public** **void** setAge(**int** age) {  **this**.age = age;  }  @Override  **public** String toString() {  **return** "Human [name=" + name + ", surname=" + surname + ", age=" + age + "]";  }  }  **class** Student **extends** Human{  **private** **int** year;  **public** Student(String name, String surname, **int** age, **int** year) {  **super**(name, surname, age);  **this**.year = year;  }  **public** **int** getYear() {  **return** year;  }  **public** **void** setYear(**int** year) {  **this**.year = year;  }  @Override  **public** String toString() {  **return** "Student [year of study=" + year + ", getName()=" + getName() + ", getSurname()=" + getSurname() + ", getAge()="  + getAge() + "]";  }  }  **class** Coworker **extends** Human{  **private** **int** year;  **public** Coworker(String name, String surname, **int** age, **int** year) {  **super**(name, surname, age);  **this**.year = year;  }  **public** **int** getYear() {  **return** year;  }  **public** **void** setYear(**int** year) {  **this**.year = year;  }  @Override  **public** String toString() {  **return** "Coworker [Coworking years=" + year + ", getName()=" + getName() + ", getSurname()=" + getSurname() + ", getAge()="  + getAge() + "]";  }  }  **class** Citizen **extends** Human{  **private** String city;  **public** Citizen(String name, String surname, **int** age, String city) {  **super**(name, surname, age);  **this**.city = city;  }  **public** String getCity() {  **return** city;  }  **public** **void** setYear(**int** year) {  **this**.city = city;  }  @Override  **public** String toString() {  **return** "Citizen [City=" + city + ", getName()=" + getName() + ", getSurname()=" + getSurname() + ", getAge()="  + getAge() + "]";  }  }    **public** **class** Humans{  **public** **static** **void** main(String[] args) {  Human[] arr = {**new** Student("Steve", "Rogers", 100, 3),  **new** Citizen("Tony", "Stark", 42, "Malibu"),  **new** Coworker("Natasha", "Romanoff", 35, 3)};  **for** (Human human : arr) {  System.***out***.println(human);  }  }  } |

Приклад виконання програми:

|  |
| --- |
| Student [year of study=3, getName()=Steve, getSurname()=Rogers, getAge()=100]  Citizen [City=Malibu, getName()=Tony, getSurname()=Stark, getAge()=42]  Coworker [Coworking years=3, getName()=Natasha, getSurname()=Romanoff, getAge()=35] |

3. Мінімум функції

Частина 1. Використання абстрактного класу.

Вміст класу AbstractEquation

|  |
| --- |
| **package** task3;  **public** **abstract** **class** AbstractEquation {  **abstract** **double** f (**double** x);    **double** solve (**double** a, **double** b, **double** step) {  **double** min = f(a);  **double** i;  **for** (i=a; i<b; i+=step) {  **if** (f(i)<f(min)) min = i;  }  **return** f(min);  }  } |

Вміст класу SpecificEquation

|  |
| --- |
| **package** task3;  **public** **class** SpecificEquation **extends** AbstractEquation{  **double** f(**double** x) {  **return** x\*x;  }  **public** **static** **void** main(String[] args) {  SpecificEquation se = **new** SpecificEquation();  System.***out***.println(se.solve(-5, 5, 1));  }  } |

Приклад виконання програми:

|  |
| --- |
| 0.0 |

Частина 2. Використання інтерфейсу.

Вміст інтерфейсу LeftSide

|  |
| --- |
| **package** task3\_2;  **public** **interface** LeftSide {  **double** f(**double** x);  } |

Вміст класу Solver

|  |
| --- |
| **package** task3\_2;  **public** **class** Solver {  **static** **double** solver(**double** a, **double** b, **double** step, LeftSide ls) {  **double** min = ls.f(a);  **double** i;  **for** (i=a; i<b; i+=step) {  **if** (ls.f(i)<ls.f(min)) min = i;  }  **return** ls.f(min);  }  } |

Вміст класу MyEquation

|  |
| --- |
| **package** task3\_2;  **class** MyEquation **implements** LeftSide {  **public** **double** f(**double** x) {  **return** x \* x;  }    **public** **static** **void** main(String[] args) {  System.***out***.println(Solver.*solver*(-5, 5, 1, **new** MyEquation()));  }  } |

Приклад виконання програми:

|  |
| --- |
| 0.0 |

4. Реалізація масиву точок через двовимірний масив

Вміст абстрактного класу AbstractArrayOfPoints

|  |
| --- |
| **package** task4;  **public** **abstract** **class** AbstractArrayOfPoints {  **public** **abstract** **void** setPoint(**int** i, **double** x, **double** y);  **public** **abstract** **double** getX(**int** i);  **public** **abstract** **double** getY(**int** i);  **public** **abstract** **int** count();  **public** **abstract** **void** addPoint(**double** x, **double** y);  **public** **abstract** **void** removeLast();  **public** **void** sortByX() { //Insertion sort  **for**(**int** i=1;i<count();i++){  **for**(**int** j=i; (j>0) && (getX(j-1) > getX(j)); j--){  **double** x = getX(j);  **double** y = getY(j);  setPoint(j, getX(j-1), getY(j-1));  setPoint(j-1, x, y);  }  }  }  @Override  **public** String toString() {  String s = "";  **for** (**int** i = 0; i < count(); i++) {  s += "x = " + getX(i) + " \ty = " + getY(i) + "\n";  }  **return** s + "\n";  }  **public** **void** test() {  addPoint(22, 45);  addPoint(4, 11);  addPoint(30, 5.5);  addPoint(-2, 48);  System.***out***.println(**this** + "Sorted by X");  sortByX();  System.***out***.println(**this**);  }  } |

Вміст класу TwoDimensional

|  |
| --- |
| **package** task4;  **public** **class** TwoDimensional **extends** AbstractArrayOfPoints{  **private** **double**[][] arr = {};    @Override  **public** **void** setPoint(**int** i, **double** x, **double** y) {  **if** (i<count()) {  arr[i][0]=x;  arr[i][1]=y;  }  }  @Override  **public** **double** getX(**int** i) {  **return** arr[i][0];  }  @Override  **public** **double** getY(**int** i) {  **return** arr[i][1];  }  @Override  **public** **int** count() {  **return** arr.length;  }  @Override  **public** **void** addPoint(**double** x, **double** y) {  **double**[][] arr1 = **new** **double**[arr.length+1][2];  **for** (**int** i=0; i<arr.length; i++) {  System.*arraycopy*(arr[i], 0, arr1[i], 0, arr[i].length);  //System.arraycopy(src, srcPos, dest, destPos, length);  }  arr1[arr.length][0] = x;  arr1[arr.length][1] = y ;  arr = arr1;  }  @Override  **public** **void** removeLast() {  **if** (count() == 0) **return**;  **double**[][] arr1 = **new** **double**[arr.length-1][2];  **for** (**int** i=0; i<arr.length; i++) {  System.*arraycopy*(arr[i], 0, arr1[i], arr.length, arr[i].length);  }  arr=arr1;  }  **public** **static** **void** main(String[] args) {  **new** TwoDimensional().test();  }  } |

Приклад виконання програми:

|  |
| --- |
| x = 22.0 y = 45.0  x = 4.0 y = 11.0  x = 30.0 y = 5.5  x = -2.0 y = 48.0  Sorted by X  x = -2.0 y = 48.0  x = 4.0 y = 11.0  x = 22.0 y = 45.0  x = 30.0 y = 5.5 |

5. Реалізація масиву точок через одновимірний масив дійсних чисел

Вміст класу AbstractArrayOfPoints

|  |
| --- |
| **package** task5;  **public** **abstract** **class** AbstractArrayOfPoints {  **public** **abstract** **void** setPoint(**int** i, **double** x, **double** y);  **public** **abstract** **double** getX(**int** i);  **public** **abstract** **double** getY(**int** i);  **public** **abstract** **int** count();  **public** **abstract** **void** addPoint(**double** x, **double** y);  **public** **abstract** **void** removeLast();  **public** **void** sortByX() {  **for**(**int** i=1;i<count();i++){  **for**(**int** j=i; (j>0) && (getX(j-1) > getX(j)); j--){  **double** x = getX(j);  **double** y = getY(j);  setPoint(j, getX(j-1), getY(j-1));  setPoint(j-1, x, y);  }  }  }  @Override  **public** String toString() {  String s = "";  **for** (**int** i = 0; i < count(); i++) {  s += "x = " + getX(i) + " \ty = " + getY(i) + "\n";  }  **return** s + "\n";  }  **public** **void** test() {  addPoint(22, 45);  addPoint(4, 11);  addPoint(30, 5.5);  addPoint(-2, 48);  System.***out***.println(**this** + "Sorted by X");  sortByX();  System.***out***.println(**this**);  }  } |

Вміст класу OneDimensional

|  |
| --- |
| **package** task5;  **public** **class** OneDimensional **extends** AbstractArrayOfPoints{  **private** **double**[] arr = {};    @Override  **public** **void** setPoint(**int** i, **double** x, **double** y) {  **if** (i<count()) {  arr[i\*2]=x;  arr[i\*2+1]=y;  }  }  @Override  **public** **double** getX(**int** i) {  **return** arr[i\*2];  }  @Override  **public** **double** getY(**int** i) {  **return** arr[i\*2+1];  }  @Override  **public** **int** count() {  **return** arr.length/2;  }  @Override  **public** **void** addPoint(**double** x, **double** y) {  **double**[] arr1 = **new** **double**[arr.length+2];  System.*arraycopy*(arr, 0, arr1, 0, arr.length);  //System.arraycopy(src, srcPos, dest, destPos, length);  arr1[arr.length] = x;  arr1[arr.length+1] = y ;  arr = arr1;  }  @Override  **public** **void** removeLast() {  **if** (count() == 0) **return**;  **double**[] arr1 = **new** **double**[arr.length-2];  System.*arraycopy*(arr, 0, arr1, arr.length, arr.length);  arr=arr1;  }  **public** **static** **void** main(String[] args) {  **new** OneDimensional().test();  }  } |

Приклад виконання програми:

|  |
| --- |
| x = 22.0 y = 45.0  x = 4.0 y = 11.0  x = 30.0 y = 5.5  x = -2.0 y = 48.0  Sorted by X  x = -2.0 y = 48.0  x = 4.0 y = 11.0  x = 22.0 y = 45.0  x = 30.0 y = 5.5 |

Висновок

Для втілення роботи наслідування класів були перевизначені деякі функції із зістосуванням анотації @Override, що дозволяє зробити перевірку відповідності похідної та базової функцій. Довелося визначати ознаку сортування — об'єкт класу CompareByTopic(), що реалізує інтерфейс компаратора. Для виконання четвертого та п'ятого завдання був використан один й той же абстрактний клас. Цей механізм написання коду є зручним, бо дозволяє використовувати різні варіанти представлення структур даних.

Наслідування у Java працюэ за схожою схемою з C++. Щодо поліморфізму, у двох мовах є деякі розбіжності. У Java усі класі поліморфні, кожна функія окрім функцій з можифікаторами final, static, private – є віртуальною.