МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

ЛАБОРОТОРНА РОБОТА №5

з дисципліни «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему "Поліморфізм. Висхідне перетворення"

Студента 2 курсу групи АД-181
Гежа Н. І.
Перевіряв
доцент Рудніченко Н. Д.

Одеса - 2019

ЗМІСТ

Вступ

Ціль роботи: ознайомитися з механікою поліморфізма в ООП. Розібратись зі статичним та динамічним зв’язуванням. Навчитись використовувати висхідне перетворення типів.

Теоретична частина

Поліморфізм є третьою важливою особливістю об'єктно-орієнтованих мов, разом з абстракцією і спадкуванням. Він являє ще одну ступінь відділення інтерфейсу від реалізації, роз'єднання що від як. Поліморфізм покращує організацію коду і його читаність, а також сприяє створенню розширюваних програм, які можуть «рости» не тільки в процесі початкової розробки проекту, але і при додаванні нових можливостей.

Зробимо порівняння із наслідуванням. Найважливіша особливість наслідування полягає не в тому, що воно дозволяє новому класу використовувати поля і методи базового, а в тому, що спадкування виражає відношення між новим і базовим класом. Це відношення можна виразити як «Новий клас є різновидом базового класу», це ж справедливо і для об'єктів цих класів. Наприклад, клас Student є різновидом Person, клас Bear є різновидом Animal і т.д.

Дане відношення підтримується мовою програмування. Наприклад, розглянемо базовий клас Instrument, який представляє музичні інструменти, і клас Guitar, який представляє музичний інструмент гітару. Так як спадкування означає, що всі методи базового класу також доступні в похідному класі, будь-яке повідомлення, яке ми можемо відправити базового класу, можна відправити і похідному класу. Якщо в класі Instrument є метод play (), то він буде присутній і в класі Guitar.

Завдання 1

Створіть клас Person, який повинен містити такі поля і геттери \ сеттери для цих полів:

- Прізвище;
- Ім'я;
- Вік.

Створіть метод printInfo (), який буде повертати строку з цими полями.

Створіть клас Student, який повинен успадковуватися від класу Person. Додайте додаткові поля, і геттери \ сеттери для цих полів:

- Група;
- Номер студентського квитка.

Перевизначите метод printInfo (), який буде повертати строку з цими полями.

Створіть клас Lecturer, який повинен успадковуватися від класу Person. Додайте додаткові поля, і геттери \ сеттери для цих полів:

- Кафедра;
- Зарплата.

Перевизначите метод printInfo (), який буде повертати строку з цими полями.

Використовуючи висхідне перетворення, створіть в класі Main кілька об'єктів класів Student і Lecturer, після чого створіть масив, який би міг включати об'єкти класів Person, Student, Lecturer. Заповніть масив об'єктами цих класів.

Використовуючи цикл, зверніться до елементів масиву і виведіть в консоль, за допомогою методу printInfo (), інформацію від кожного об'єкта.

Реалізація завдання 1

Код програми у файлі Person:

```
package AD181.Gezha;

public class Person {
  private String surname = "";
  private String name = "";
  private int age = 0;

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

  public String printInfo() {
    return "Person " + this.getSurname() + " " +
           this.getName() + ", age: " + this.getAge();
  }

  public void setSurname(String surname) {
    this.surname = surname;
  }

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

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

  public String getSurname() {
    return surname;
  }

  public String getName() {
    return name;
  }

  public int getAge() {
    return age;
  }
}
```

Код програми у файлі Student:

```
package AD181.Gezha;

public class Student extends Person {
  private String group = "";
  private String studentID = "";

  public Student(String surname, String name, int age, String group, String
studentID) {
    super(surname, name, age);
    this.group = group;
    this.studentID = studentID;
  }

  @Override
  public String printInfo() {
    return "Student of group " + this.getGroup() + " " + this.getSurname() + " "
+
           this.getName() + ", age: " + this.getAge() +
           ". Student ID: " + this.getStudentID();
  }

  public String getGroup() {
    return group;
  }

  public void setGroup(String group) {
    this.group = group;
  }

  public String getStudentID() {
    return studentID;
  }

  public void setStudentID(String studentID) {
    this.studentID = studentID;
  }
}
```

Код програми у файлі Lecturer:

```
package AD181.Gezha;

public class Lecturer extends Person {
  private String department = "";
  private double salary = 0.0;

  public Lecturer(String surname, String name, int age, String department,
double salary) {
    super(surname, name, age);
    this.department = department;
    this.salary = salary;
  }

  @Override
  public String printInfo() {
    return "Lecturer of " + this.getDepartment() + " department " +
this.getSurname() + " " +
           this.getName() + ", age: " + this.getAge() +
           ". Salary: " + this.getSalary();
  }

  public String getDepartment() {
    return department;
  }

  public void setDepartment(String department) {
    this.department = department;
  }

  public double getSalary() {
    return salary;
  }

  public void setSalary(double salary) {
    this.salary = salary;
  }
}
```

Код програми у файлі Main:

```
package AD181.Gezha;

public class Main {
  public static void main(String[] args) {
    Student student1 = new Student("Ross", "Bob", 18, "BS-181", "11037");
    Student student2 = new Student("Cheese", "William", 20, "AA-113", "1498");
    Lecturer lecturer = new Lecturer("Smith", "John", 42, "physics", 2500.0);

    Person[] humans = {student1, student2, lecturer};
    for (Person human : humans) {
      System.out.println(human.printInfo());
    }
  }
}
```

Програма була запущена та виконала свої задачі як планувалося (Рис. 1)

```
Student of group BS-181 Ross Bob, age: 18. Student ID: 11037
Student of group AA-113 Cheese William, age: 20. Student ID: 1498
Lecturer of physics department Smith John, age: 42. Salary: 2500.0
```

Рис. 1: результат виконання програми

Завдання 2

В архіві з лабораторної роботою знаходиться директорія з проектом.

Дана програма являє собою простий малювач фігур.

Завдання полягає в наступному:

1. Відкрити проект, імпортувати код в Idea і розібратися - як працює програма.
2. При здачі лабораторної програми ви повинні бути готовим показати, де і для яких цілей може використовуватися поліморфізм і висхідний перетворення.
3. Використовуючи механізм поліморфізму, змінюйте програму, додавши можливість вибрати і малювати нову фігуру (наприклад трапецію, ромб або іншу).

Реалізація завдання 2

До програми був написаний клас MyRhombus, який реалізує інтерфейс Class2D, задля можливості малювання ромба. Фігура задається двома точками, які будуть кінцями більшої діагоналі ромба, координати кінців малої діагоналі розраховуються таким чином, щоб коротша діагональ відносилась до більшої як задано у змінній "`sideProportion`". Задля збереження стилю існуючого коду, корисних коментарів не було написано. Код наведений нижче:

```
package paint;

import java.awt.Color;
import java.awt.Point;
import java.awt.Polygon;
import java.io.BufferedWriter;
import java.io.IOException;

public class MyRhombus implements Class2D {
  private Polygon rectangle;
  private Color lineColor;
  private Color fillColor = null;

  public void makeObject(Point startDrag, Point endDrag) {
    setLineColor(GUI.selectColor);
    double startX = startDrag.x;
    double startY = startDrag.y;
    double endX = endDrag.x;
    double endY = endDrag.y;
    double distanceX = endX - startX;
    double distanceY = (endY - startY);
    double length = Math.hypot(distanceX, distanceY);
    double centerX = startX + distanceX / 2;
    double centerY = startY + distanceY / 2;
    double asine = Math.asin(distanceY / length);
    double acosine = Math.acos(distanceX / length);

    double sideProportion = 0.5;

    // pls send help
    int[] xs = {
        (int) startX,
        (int) (centerX + length * sideProportion * Math.sin(asine + 1.41) / 2),
        (int) endX,
        (int) (centerX + length * sideProportion * Math.sin(asine - 1.41) / 2)
    };
    int[] ys = {
        (int) startY,
        (int) (centerY + length * sideProportion * Math.cos(acosine + 1.41) /
2),
        (int) endY,
        (int) (centerY + length * sideProportion * Math.cos(acosine - 1.41) / 2)
    };

    rectangle = new Polygon(xs, ys, 4);
    this.setRectangle(rectangle);
```

```
  }

  public void setRectangle(Polygon rectangle) {
    this.rectangle = rectangle;
  }

  public Polygon getRectangle() {
    return this.rectangle;
  }

  public void fill(Color c) {
    this.setColor(c);
  }

  public Color getColor() {
    return this.fillColor;
  }

  public void setColor(Color color) {
    this.fillColor = color;
  }

  public Color getLineColor() {
    return this.lineColor;
  }

  public void setLineColor(Color lineColor) {
    this.lineColor = lineColor;
  }

  @Override
  public boolean contains(Point p) {
    return getRectangle().contains(p);
  }

  @Override
  public void move(Point startDrag, Point endDrag) {
    int[] xs = {0, 0, 0, 0};
    int[] ys = {0, 0, 0, 0};
    for (int k = 0; k < 4; k++) {
      xs[k] = this.getRectangle().xpoints[k] + endDrag.x - startDrag.x;
      ys[k] = this.getRectangle().ypoints[k] + endDrag.y - startDrag.y;
    }
    this.setRectangle(new Polygon(xs, ys, 4));
  }

  @Override
  public void writetoFile(BufferedWriter b) {
    try {
      b.write(getClass().getSimpleName() + ";");
      b.write(getRectangle().xpoints[0] + ";" + getRectangle().ypoints[0] +
";");
      b.write(getRectangle().xpoints[2] + ";" + getRectangle().ypoints[2] +
";");
      b.write(getLineColor().getRed() + ";" + getLineColor().getGreen() + ";" +
getLineColor().getBlue() + ";");
      if (getColor() == null) {
        b.write("null" + ";" + "null" + ";" + "null");
      } else {
        b.write(getColor().getRed() + ";" + getColor().getGreen() + ";" +
getColor().getBlue());
```

```
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  @Override
  public void draw(GraphicAdapter g) {
    if (getColor() == null) {
      g.getGraphicAdapter().setColor(getLineColor());
      g.getGraphicAdapter().drawPolygon(getRectangle());
    } else {
      g.getGraphicAdapter().setColor(getColor());
      g.getGraphicAdapter().fillPolygon(getRectangle());
    }
  }
}
```

Задля створення кнопки для визначення мальованої фігури як ромба, у конструктор класу GUI був додан слідуючий код:

```
JButton btnRhomb = new JButton("Rhombus");
btnRhomb.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    selectShap = "Rhombus";
  }
});
panel.add(btnRhomb);
```

Задля можливості завантажувати фігуру ромба з файлу, до конструктору класу OpenFile був додан наступний код:

```
} else if (a[0].equals("MyRhombus")) {
       MyRhombus r = new MyRhombus();
       r.makeObject(new Point(Integer.parseInt(a[1]), Integer.parseInt(a[2])),
                                          new Point(Integer.parseInt(a[3]),
Integer.parseInt(a[4])));
       r.setLineColor(new Color(Integer.parseInt(a[5]), Integer.parseInt(a[6]),
Integer.parseInt(a[7])));
       if (a[8].equals("null")) {
            r.setColor(null);
       } else {
            r.setColor(new Color(Integer.parseInt(a[8]), Integer.parseInt(a[9]),
Integer.parseInt(a[10])));
       }
       GUI.paint.add(r);
       GUI.frame.repaint();
```

Задля можливості створювати ромб, до функції paint() класу Paint_App було додано наступний код:

```
} else if (GUI.selectShap == "Rhombus") {
  MyRhombus obj = new MyRhombus();
  obj.makeObject(startDrag, endDrag);
  obj.draw(g2);
```

Задля можливості мати візуальний орієнтир при переміщенні фігури ромбу, до функції paint() класу Paint_App було додано наступний код:

```
} else if (ptemp instanceof MyRhombus) {
  MyRhombus rhombus = (MyRhombus) ptemp;
  if (rhombus.contains(startDrag)) {
    int[] newXs = {0, 0, 0, 0};
    int[] newYs = {0, 0, 0, 0};
    for (int i = 0; i < 4; ++i) {
      newXs[i] = rhombus.getRectangle().xpoints[i] + endDrag.x - startDrag.x;
      newYs[i] = rhombus.getRectangle().ypoints[i] + endDrag.y - startDrag.y;
    }

    if (rhombus.getColor() == null) {
      g2.getGraphicAdapter().setColor(rhombus.getLineColor());
      g2.getGraphicAdapter().drawPolygon(newXs, newYs, 4);
    } else {
      g2.getGraphicAdapter().setColor(rhombus.getColor());
      g2.getGraphicAdapter().fillPolygon(newXs, newYs, 4);
      g2.getGraphicAdapter().drawPolygon(newXs, newYs, 4);
    }
  }
```

Програма була запущена та перевірена (Рис. 2). Увесь функціонал стосовно програмної логіки роботи програми (створення фігури, розмалювання фігури, збереження та завантаження фігури з диску, пересування та видалення фігури) працює належним чином. Але через погано реалізовані геометричні формули розташування точок ромбу, фігура є ромбом тільки у деяких окремих випадках.

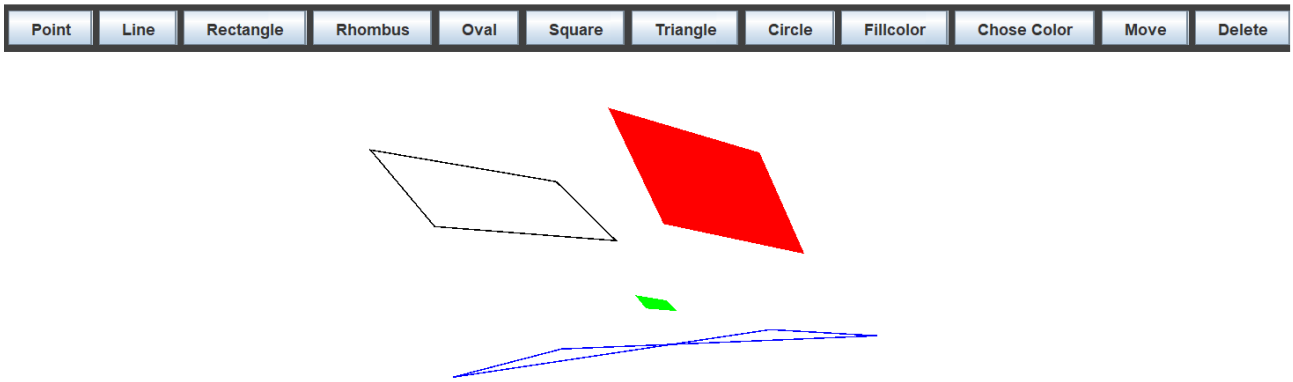

Рис. 2: намальовані в програмі ромби

Висновок

Було проведено ознайомлення з поліморфізмом у мові програмування Java, було розібрано статичне та динамічне зв’язування та приведення типів. Прикладений до лабораторної роботи проект було модифіковано з додаванням нової фігури і перевірено, критичних помилок не було помічено, але через некоректні розрахунки нова фігура не завжди є правильною.

Список літератури

1. Сайт https://www.tutorialspoint.com
2. Приклади до лабораторної роботи