МЕТОДИЧНА РОЗРОБКА

для проведення практичного заняття з студентами (курсантами) 1-го курсу спеціальності 122 «Комп'ютерні науки» з дисципліни «Основи програмування»

Лабораторне заняття 6.6: «Застосування поліморфізму»

Мета: закріпити теоретичні знання та практичні навики щодо перевизначення методів та конструкторів класів, а також написання рекурсивних методів при створенні прикладних програм за об'єктно-орієнтованим принципом. Закріплення навиків щодо застосування принципів поліморфізму в межах одного класу та між класами одного пакету.

Час: 2 годин

Місце заняття: комп'ютерна лабораторія

Матеріальне забезпечення:
- ПК з відповідним ПЗ та доступом до мережі Internet;
- Методичні рекомендації щодо виконання лабораторної роботи.

Після лабораторного заняття студенти (курсанти) повинні:

вміти: здійснювати перевизначення методів та конструкторів класу, створювати рекурсивні методи.

знати: принципи застосування поліморфізму та рекурсії в обє'ктно-орієнтованому програмуванні.

Література:

1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з : http://virt.ldubgd.edu.ua

План заняття:

1. Перевизначення методів
2. Перевизначення конструкторів
3. Видача індивідуальний практичних завдань

1. Перевизначення методів

Як зазначено в попередній лекції, перевизначення методів в межах одного класу можливо реалізувати шляхом заміни типу даних та кількості параметрів методу. Кількість параметрів, що передаються у метод можливо збільшувати, зменшувати, вводити, за їх відсутності, або видаляти у разі потреби. Відповідно під час виклику такого методу Java надаватиме можливість обрати необхідний варіант для виконання. Не слід забувати і про можливість перевизначення методу шляхом зміни типу даних, які повертає цей метод.

Розглянемо процедуру перевизначення методів детальніше. Для наочності обрано приклад розрахунку параметрів прямокутної пожежі (розглянутий на попередніх заняттях):

```
public class Firerectangular {
	double length;
	double width;

	public Firerectangular(double length, double width) {
		this.length = length;
		this.width = width;
	}

	double square() {
		double s;
		s = length * width;
		return s;
	}

	void perimeter() {
		double p = 2 * (length + width);
		System.out.println(p);
	}

	void front(double f, double n) {
		if (f == length && n == 1) {
			System.out.println(length);
		} else if (f == length && n == 2) {
			System.out.println(length * 2);
		} else if (f == width && n == 1) {
			System.out.println(width);
		} else if (f == width && n == 2) {
			System.out.println(width * 2);
		} else {
			System.out.println("Ви ввели не коректне значен-
ня");
		}
	}
}
```

В наведеному прикладі представлено клас `Firerectangular` із двома змінними екземпляра класу та конструктором для їх ініціалізації. В якості основних методів написані: типізований метод `square()` для визначення площі прямокутної пожежі, **`void`** -метод `perimeter()` – для визначення периметру та **`void`** -метод, що приймає параметри і визначає фронт пожежі `front(double f, double n)`.

Зважаючи на мету заняття проведемо перевизначення означених методів в межах класу шляхом зміни типу даних з якими працюють методи та введенням параметрів у відповідні методи. Спочатку проведемо перевизначення методу `square()`. В якості параметрів методу передаються **локальні** змінні **`int`** `length` та **`int`** `width`, які **не залежать** від змінних екземпляра класу (локальні змінні перекривають змінні екземплярів класу з однаковими іменами):

```
public class Firerectangular {
	double length;
	double width;
```

```
    public Firerectangular(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double square() {
        double s;
        s = length * width;
        return s;
    }

    // перевизначений метод, що повертає значення площі пожежі
    int square(int length, int width) {
        int s = length * width;
        return s;
    }
```

З наведеного прикладу спостерігається, що один і той самий метод `square()` може викликатись у двох різних варіантах виконання, а саме без прийняття параметрів із результатом роботи типу **`double`**, або із прийняття параметрів у вигляді локальних змінних **`int`** `length` і **`int`** `width` та результатом своєї роботи типу **`int`**.

Далі проведемо перевизначення методу `perimeter()` шляхом введенням параметру **`double`** `n`, який буде використано під час роботи методу:

```
public class Firerectangular {
    double length; // змінні екземплярів класу
    double width;

public Firerectangular(double length, double width) {
    this.length = length;
    this.width = width;
}

    double square() {
        double s;
        s = length * width;
        return s;
    }

    // перевизначений метод, що повертає значення площі пожежі
    int square(int length, int width) {
        int s = length * width;
        return s;
    }

    void perimeter() {
        double p = 2 * (length + width);
        System.out.println(p);
    }

    // перевизначений метод, що повертає знач. периметру пожежі
    void perimeter(double n) {
        double p = n * (length + width);
```

```
            System.out.println(p);
        }
    }
```

За умови перевизначення методу `perimeter()` користувач отримає можливість викликати його з `main()`-методу за умови введення аргументу **`double`** `n`, або без передачі аргументів (в такому випадку n = 2).

На черзі останній метод `front()` класу `Firerectangular`. Оригінальний метод відповідає за визначення фронту прямокутної пожежі залежно від однієї з чотирьох умов. Проведемо його перевизначення шляхом скорочення кількості параметрів, які передаються методу. Крім того логікою методу передбачено умову, що змінні екземпляра класу **`double`** `length` та **`double`** `width` рівні:

```
public class Firerectangular {
    double length; // змінні екземплярів класу
    double width;

    public Firerectangular(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double square() {
        double s;
        s = length * width;
        return s;
    }

    // перевизначений метод, що повертає значення площі пожежі
    int square(int length, int width) {
        int s = length * width;
        return s;
    }

    void perimeter() {
        double p = 2 * (length + width);
        System.out.println(p);
    }

    // перевизначений метод, що повертає знач. периметру пожежі
    void perimeter(double n) {
        double p = n * (length + width);
        System.out.println(p);
    }

    void front(double f, double n) {
        if (f == length && n == 1) {
            System.out.println(length);
        } else if (f == length && n == 2) {
            System.out.println(length * 2);
        } else if (f == width && n == 1) {
            System.out.println(width);
        } else if (f == width && n == 2) {
```

```
                System.out.println(width * 2);
            } else {
                System.out.println("Ви ввели не коректне значен-
ня");
            }
        }

        // перевизначений метод, що повертає значення фронту пожежі
        // за умови рівності усіх сторін
        void front(double n) {
            if (n == 1) {
                System.out.println(length);
            } else if (n == 2) {
                System.out.println(length * 2);
            } else if (n == 3) {
                System.out.println(length * 3);
            } else {
                System.out.println(length * 4);
            }
        }
}
```

В результаті перевизначення методу `front()` буде отримано можливість визначення фронту пожежі за умови різних, або однакових значень змінних `double length` та `double width`.

Далі для тестування працездатності попередньо написаних та перевизначених методів проведемо їх виклик в головному класі з `main()`-методом:

```
public class MainFire {
    public static void main(String[] args) {
        Firerectangular fr = new Firerectangular(10, 10.5);

        System.out.println(fr.square());
        System.out.println(fr.square(5, 6));

        fr.perimeter();
        fr.perimeter(1.85);

        fr.front(10, 1);
        fr.front(3);
    }
}

Результат роботи програми:
105.0
30
41.0
37.925000000000004
10.0
30.0
```

Доступний також варіант перевантаження одного і того ж методу в різних класах. Суть такого перевизначення полягає в тому, що для різних класів виклик ме-

тоду з одним і тим самим іменем може виконувати різну логіку (спільний інтерфейс, логіка різна). Для наочності застосування такого інструменту створимо додатковий клас `FireCircular`, метою якого є визначення параметрів кругової пожежі:

```
public class FireCircular {
	double radius;

	public FireCircular(double radius) {
		this.radius = radius;
	}
}
```

Далі в означеному класі проведемо перевизначення методів `square()` та `perimeter()` таким чином, щоб назва, тип даних та параметри методів залишались не змінними. Наповнення методів (їх логіка) має визначати параметри пожеж різного виду, залежно від того до якого екземпляра класу їх буде викликано (параметри прямокутної або кругової пожежі):

```
public class FireCircular {
	double radius;

	public FireCircular(double radius) {
		this.radius = radius;
	}

	double square() {
		double s;
		s = Math.PI * Math.pow(radius, 2);
		return s;
	}

	void perimeter() {
		double p = 2 * Math.PI * radius;
		System.out.println(p);
	}
}
```

Для того, щоб переконатись в працездатності перевизначених методів `square()` та `perimeter()` в класі `FireCircular`, а також в тому, що перевизначення не вплинуло на відповідні методи класу `Firerectangular` проведемо їх виклик в класі `MainFire` із `main()`-метедом:

```
public class MainFire {
	public static void main(String[] args) {
		Firerectangular fr = new Firerectangular(10, 10.5);
		FireCircular fc = new FireCircular(15);

		System.out.println(fr.square());
		fr.perimeter();

		System.out.println(fc.square());
```

```
            fc.perimeter();
        }
}

Результат роботи програми:
105.0
41.0
706.8583470577034
94.24777960769379
```

Як видно з наведеного прикладу виклик одного і того ж методу до різних екземплярів класу зумовлює до одержання різних результатів виконання. Це відбувається за рахунок виклику відповідного методу саме з того класу, до екземпляра якого застосовано виклик. Цей приклад яскраво відтворює принцип поліморфізму в об'єктно-орієнтованому програмуванні.

2. Перевизначення конструкторів

Зважаючи на те, що в процесі написання програм часто виникає необхідність ініціалізації змінних екземплярів класів по різному, в об'єктно-орієнтованих мовах програмування передбачено можливість перевизначення конструкторів. Перевизначення конструкторів також є однією із форм прояву поліморфізму. Для наочності використано попередній приклад класу `Firerectangular` із двома змінними екземплярів класу. В наведеному прикладі для ініціалізації змінних екземплярів класу було визначено стандартний конструктор, який надає можливість передати значення змінним під час створення екземпляра класу. Проте така форма не передбачає можливості ініціалізації змінних іншими способами у разі необхідності. Наприклад, якщо змінні `double length` та `double width` є рівними, то їх ініціалізація може проводитись з використанням одного параметру. Інший приклад це створення екземпляра класу без ініціалізації змінних (порожнього об'єкта), або передача в конструктор посилання на інший екземпляр класу з метою створення «клону» вже існуючого об'єкта.

В наступному прикладі проведемо процедуру перевизначення конструктора трьома різними способами. Кінцевий вигляд класу `Firerectangular` із перевизначеним конструктором матиме такий вигляд:

```
public class Firerectangular {
    double length; // змінні екземплярів класу
    double width;

    // генерований конструктор
    public Firerectangular(double length, double width) {
        this.length = length;
        this.width = width;
    }

    // перевизначений конструктор, яким передбачено
    // встановлення рівності усіх сторін
    public Firerectangular(double i) {
        length = width = i;
    }
```

```
        // перевизначений конструктор, яким передбачено
        // створення клону існуючого екземпляра класу
        public Firerectangular(Firerectangular fr) {
             length = fr.length;
             width = fr.width;
        }

        // перевизначений конструктор, що не
        // приймає жодного параметру
        public Firerectangular() {
        }
}
```

Далі в головному класі з `main()`-методом проведемо ініціалізацію змінних екземплярів класу із застосуванням усіх перевизначених конструкторів:

```
public class MainFire {
     public static void main(String[] args) {
          Firerectangular fr = new Firerectangular(10, 10.5);
          Firerectangular fr1 = new Firerectangular(10);
          Firerectangular fr2 = new Firerectangular(fr);
          Firerectangular fr3 = new Firerectangular();
          fr3.length = 11;
          fr3.width = 12.5;
     }
}
```

Як видно з представленого прикладу за умови виклику одного із перевизначених конструкторів користувачеві надається можливість ініціалізовувати змінні різними способами. В першому випадку проводиться стандартна ініціалізація, де за допомогою конструктора відбувається присвоєння значень зміним `length` та `width`. За умови застосування другого варіанту конструктора відбувається присвоєння однакового значення у дві змінні `length` та `width`. Третій конструктор надає можливість створити клон вже існуючого екземпляра класу ініціалізовуючи змінні ідентично до об’єкту, який передано в якості аргументу. Останній варіант конструктора надає можливість створювати об’єкти без ініціалізації змінних, а в разі виникнення такої потреби ініціалізація проводиться окремо.

3. Видача індивідуальний практичних завдань

У відповідності до тематики заняття для виконання практичного завдання необхідно розширити написану раніше об’єктно-орієнтовану програму (за результатами заняття №6.4) дотримуючись таких вказівок:

1. В кожному із 3-х існуючих класів програми (крім головного класу) необхідно:

1.1. Перевизначити щонайменше однин метод, що прийматиме в якості параметрів дані іншого типу із збереженням загальної логіки методу (наприклад: якщо дочірній метод працює з даними типу **`int`** то перевизначений метод має приймати в якості параметра та працювати з даними типу **`double`**);

1.2. Написати один метод з статичним ім’ям та динамічною логікою (назва, тип даних та параметри однакові, зміст різний) і перевизначити його таким чином, щоб за умови виклику цього методу до екземплярів різних класів в `main()`-методі результат виконання був унікальним (різним).

3. В головному класі з `main()`-методом (4-й клас) викликати усі перевизначені методи до відповідних екземплярів класів.

4. В одному із існуючих класів перевизначити конструктор 4-ма довільними варіантами таким чином, щоб в `main()`-методі, при створенні екземплярів цього класу, користувачеві надавалась можливість обирати 4 варіанти ініціалізації змінних. В головному класі з `main()`-методом ініціалізувати змінні усіма можливими варіантами при створенні екземплярів відповідних класів.

Результати виконання практичного завдання являються вихідними даними для виконання чергового індивідуального завдання у відповідності до програми курсу.

Звіт подається чотирма файлами із розширенням *.java. Назви файлів мають відповідати назвам класів. Зміст файлів має відповідати поставленій умові в п.4. Програмний код рекомендовано супроводжувати коментарями щодо його змісту та логіки.

Звіт за результатами виконання лабораторної роботи (4 файли) завантажується у відповідну категорію Віртуального університету.

Методичну розробку склав:
Начальник кафедри УПІТтаТ
підполковник служби цивільного захисту  Олександр ПРИДАТКО

Методична розробка обговорена на засіданні кафедри УПІТтаТ
Протокол № _*1*_ від "_*27*_" *серпня*_ 20_*20*_ р.