ЗАТВЕРДЖУЮ
Завідувач кафедри управління проектами, інформаційних технологій та телекомунікацій, д.т.н., професор
Євген МАРТИН
«_ _ __»_____20___ р.

МЕТОДИЧНА РОЗРОБКА

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

Практичне заняття 6.2: «Створення власних класів. Застосування конструкторів»

Мета: здобути навики щодо створення класів, екземплярів класів, змінних екземплярів класів, а також ініціалізації змінних екземплярів класів з використанням конструкторів.

Час: 2 годин

Місце заняття: комп'ютерна лабораторія

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

Після практичного заняття студенти (курсанти) повинні:
вміти: створювати класи та їх екземпляри, проводити ініціалізацію змінних екземплярів класів різними способами (у т.ч. з використанням конструктора).
знати: порядок створення власних класів, а також порядок взаємозв'язку між класами, об'єктами та змінними екземплярів класів.

Література:
1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з : http://virt.ldubgd.edu.ua/course/view.php?id=1778

План заняття:
1. Створення класів, екземплярів класів та змінних екземплярів класів
2. Створення конструкторів та ініціалізація змінних екземплярів класу з їх використанням
3. Видача індивідуальних практичних завдань

1. Створення класів, екземплярів класів та змінних екземплярів класів

Розпочнемо з основного – створення класу. Якщо приймати конструкцію класу абстрактно, то його можна порівняти з контейнером (оболонкою), в якій здійснюють написання програмного коду (усієї програмної логіки).

Зважаючи на специфіку навчального закладу, в якості наочного прикладу, розглянемо процедуру написання класу із визначення параметрів прямокутної пожежі. Створення класу розпочинається із ключового слова `class` та назви `Firerectangular` (модифікатор доступу `public` поки що не обов'язковий):

```
public class Firerectangular {

}
```

Після створення класу у відповідному каталозі комп'ютера (який вказано під час запуску середовища розробки) буде створено файл Firerectangular.java.

Наступним компонентом в ієрархії класу є його поля – змінні екземпляру класу. При оголошенні змінних зазначається тип даних та назва (аналогічно звичайним змінним):

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

Наступним кроком є створення екземпляра касу `Firerectangular` із довільним ім'ям. Екземпляри класів створюють в головному класі із main()-методом:

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

Враховуючи, що в основному класі `Firerectangular` оголошено дві змінні, то в класі `MainFire`, після створення екземпляра класу `fr`, необхідно їх ініціалізувати:

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

В результаті виконаних маніпуляцій змінні екземпляра класу `fr` (`length` та `width`) ініціалізовані та отримали значення 10 і 10,5 відповідно.

Один клас може мати довільну кількість екземплярів. І для кожного екземпляра класу, його змінні можуть ініціалізовуватись різними значеннями. Розглянемо приклад:

```
public class MainFire {
	public static void main(String[] args) {
		Firerectangular fr = new Firerectangular();
		Firerectangular fr1 = new Firerectangular();
		fr.length = 10;
		fr.width = 10.5;
		fr1.length = 20;
		fr1.width = 21;
}
```

Далі, для визначення, наприклад, площі прямокутних пожеж для окремих екземплярів класів (`fr` та `fr1`), необхідно відтворити відповідні вирази із записом результатів розрахунків у новостворені змінні. Для виводу результатів у консоль скористаємось методом `println()`:

```
public class MainFire {
	public static void main(String[] args) {
		Firerectangular fr = new Firerectangular();
		Firerectangular fr1 = new Firerectangular();
		fr.length = 10;
		fr.width = 10.5;
		fr1.length = 20;
		fr1.width = 21;

		double s = fr.length * fr.width;
		double s1 = fr1.length * fr1.width;

		System.out.println("Площа першої пожежі: " + s);
		System.out.println("Площа другої пожежі: " + s1);
}
```

Недоліком подібного представлення коду є те, що за необхідності визначення площі пожежі для кожного екземпляру класу необхідно створювати свою підпрограму для розрахунку, а це значно нагромаджуватиме main()-метод, що є небажаним при об'єктно-орієнтованому підході. Якщо кількість екземплярів класів складатиме 20, то це зумовить до написання 20 окремих стрічок для розрахунку із оголошенням 20 нових змінних для запису результатів розрахунків. Це не зовсім практично та не відповідає парадигмі об'єктно-орієнтованого програмування. Методи, для виконання подібних розрахунків, мають володіти уніфікованим характером (написав 1 раз, використав n разів) та розміщуватись в основному класі. А main()-метод має забезпечувати лише їх виклик для конкретного екземпляру класу. Власне це питання буде більш детально розглянуто у наступних темах.

2. Створення конструкторів та ініціалізація змінних екземплярів класу з їх використанням

На даний момент ми навчились проводити ініціалізацію змінних екземплярів класу безпосередньо в main()-методі. За умови застосування такого підходу, кожну змінну потрібно інціалізовувати окремо із прив'язкою до визначеного екземпляру класу. Така процедура значно нагромаджує програмний код, наприклад:

```
fr.length = 10;
fr.width = 10.5;
fr1.length = 20;
fr1.width = 21;
```

Для вирішення цієї проблеми в програмуванні використовують **конструктори**. Згідно ієрархії програмного коду, конструктори в класі розміщуються після змінних екземплярів класу (хоча це не принципово). Розглянемо приклад:

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

	public Firerectangular(double l, double w) {
		length = l;
		width = w;
	}
}
```

Конструктор за формою схожий на типізований метод (буде розглянуто в наступній темі), який приймає параметри. Тільки в якості параметрів передаються значення змінних екземплярів класу, які присвоюються цим змінним в тілі конструктора за допомогою оператора « = ». Наявність конструктора значно спрощує ініціалізацію змінних екземплярів класу та дозволяє здійснити цю процедуру при першій згадці про екземпляр:

```
public class MainFire {
	public static void main(String[] args) {
		Firerectangular fr = new Firerectangular(10, 10.5);
		Firerectangular fr1 = new Firerectangular(20, 21);
}
```

Таким чином програмний код набуває значно естетичнішого вигляду та не нагромаджує main()-метод.

Більшість середовищ розробки володіють властивістю автоматичної генерації конструктора. До прикладу в IDE Eclipse, для доступу до цієї опції слід скористатись комбінацією клавіш Shift+Alt+S, після чого користувачеві надається можливість обрати перелік змінних, для яких генерується конструктор та місце розташування конструктора. Після автоматичної генерації конструктор набуватиме такого вигляду:

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

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

```
    }
}
```

За умови автоматичного генерування конструктора, спосіб ініціалізації змінних в головному класі залишається без змін. Як видно з представлених прикладів наявність конструктора значно спрощує процес ініціалізації змінних екземплярів класу, особливо за умови їх великої кількості.

3. Видача індивідуальних практичних завдань.

У відповідності до тематики заняття для виконання практичного завдання необхідно написати об'єктно-орієнтовану програму із довільною логікою дотримуючись таких вказівок:

1. Створити три класи в межах одного пакету (Package) (наприклад: `class One, class Two, class Three`). Створення класів реалізується з допомогою меню File/New/Class в середовищі розробки Eclipse. При створенні класів **не потрібно** проводити автоматичне генерування `main()`-методу.

2. В кожному із створених класів оголосити по три змінні екземплярів класу із довільним типом даних **без їх ініціалізації** (рекомендовано використовувати типи `double` або `int`). Приклади оголошення змінних екземплярів класу наведено в методичній розробці, а також в лекції.

3. В кожному із трьох класів написати або згенерувати конструктор для спрощення процедури ініціалізації усіх змінних екземплярів класу.

4. Створити головний клас із `main()`-методом (4-й клас). В головному класі в межах `main()`-методу створити екземпляри (об'єкти) оголошених раніше класів (`class One, class Two, class Three`) з довільними іменами та довільною ініціалізацією змінних. Приклади створення екземплярів класу наведено в методичній розробці, а також в лекції.

Результатом виконання практичного завдання є базова конструкція об'єктно-орієнтованої програми з довільною логікою, яка міститиме три класи із трьома змінними екземплярів класу та конструктором (в кожному класі). Результати виконання практичного завдання являються вихідними даними для виконання чергового завдання наступної теми у відповідності до програми курсу.

Звіт до лабораторної роботи подається чотирма файлами із розширенням *.java. Назви файлів мають відповідати назвам класів. Зміст файлів має відповідати поставленій умові в п.3. Програмний код рекомендовано супроводжувати коментарями щодо його змісту та логіки, наприклад:

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

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

```
}
```

Звіт за результатами виконання практичного завдання (4 файли) завантажується у відповідну категорію Віртуального університету.

Методичну розробку розробив:
Заступник начальника кафедри УПІТтаТ
підполковник служби цивільного захисту Олександр ПРИДАТКО

Методична розробка обговорена на засіданні кафедри УПІТтаТ
Протокол № ____ від "___" _____ 20__ р.