Особливості використання класів

- Повернення значень з методів
- Ключове слово this
- Статичні члени класу
- Ключове слово final

Повернення значень з методів

- Метод повертається в код, що викликав його, коли:
  - Виконано всі оператори методу
  - Зустрівся оператор return
  - Викинуто виключення
- Завершення методу відбувається за першою подією

Ключове слово return

- Ви оголошуєте тип значення, що повертається, в декларації методу.
  - У тілі методу, використовується оператор **return** для повернення значення.
- Будь-який метод що містить в декларацій **void** не повертає значення. Він не повинен містити оператор **return**, але з нього можна повернутися наступним чином:
  - return;
- Якщо ви спробуєте повернути значення з методу, який має **void**, ви отримаєте помилку компілятора.
- Будь-який метод, з вказаним значенням, що повертається, повинен містити оператор повернення:
  - return ReturnValue;

Приклад використання return

```
public void print(boolean shouldPrint){
    if (!shouldPrint){
        return;
    }
    System.out.println("Test");
}

public double getVolume(double a, double b, double c){
    double volume = a*b*c;
    return volume;
}
```

Особливості передачі параметрів

- У Java існує всього один тип передачі параметрів - передача за значенням.
- При передачі параметра виділяється необхідна область пам'яті, куди копіюється значення параметра, і всередині методу робота йде з цією копією.
- Копія буде знищена при виході з методу.

Ключове слово this

- Ключове слово **this** використовується коли у методу виникає необхідність звертатися до об'єкта, який його викликав.
- **this** - це завжди посилання на об'єкт, метод якого був викликаний.
- Ви можете використовувати this всюди, де дозволяється посилання на об'єкт поточного класу.

this в конструкторі

- this в конструкторі може бути використано для виклику іншого конструктора того ж класу
  - Такий виклик називається явним викликом конструктора
- Якщо явний виклик конструктора присутній то він повинен бути першим рядком конструктора.

```
public Box(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
}

public Box(){
        this(0,0,0);
}
```

Вирішення “проблеми” затінення

Повторне використання конструктора

Статичні члени класу

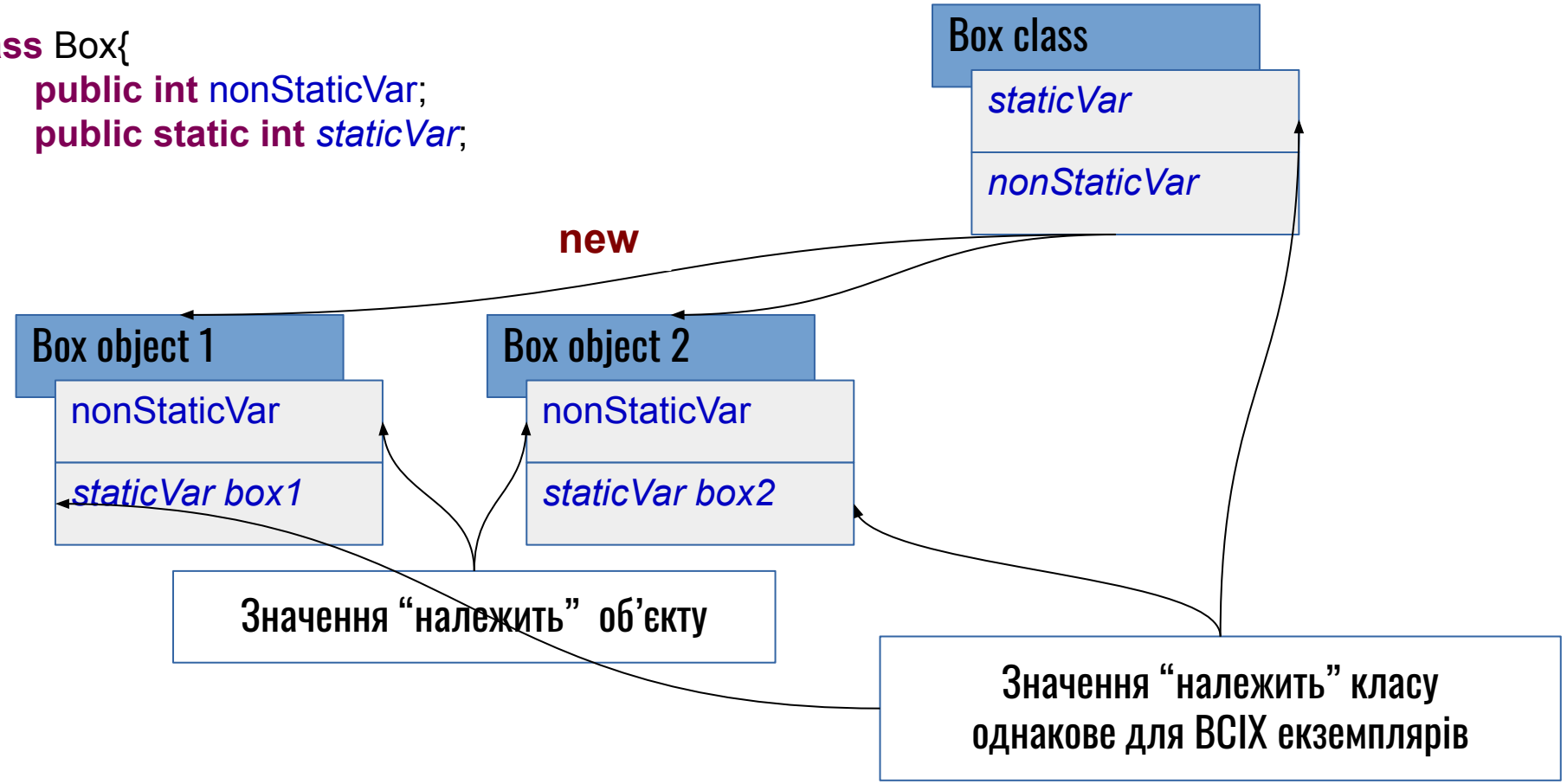
- У Java поля і методи можуть бути описані як static.
  - static int someStaticVar = 0;
- Поля, описані як static зберігаються в єдиній копії для всіх об'єктів даного типу і навіть без існування якого-небудь об'єкту.
- Є два шляхи звернення до статичних полях:
  - Через посилання на будь-який екземпляр класу
    - SomeClass someRef = new someClass ();
    - someRef.someStaticVar = 10;
  - Через ім'я класу
    - SomeClass.someStaticVar = 10
- **Рекомендується звертатися до статичних полях тільки використовуючи ім'я класу**

Статичні методи

- Java підтримує статичні методи в той же спосіб що і статичні поля
- Звернення до статичних методів аналогічне до статичних полів:
    - SomeClass.someMethod ();

Об’єкти, класи яких мають static

```
class Box{
        public int nonStaticVar;
        public static int staticVar;
}
```



Іменовані константи

- Якщо статичне поле класу, або змінна проініціалізовані константним виразом, вони розглядаються компілятором, як іменована константа
- Їх значення може бути використане в операторах switch, а також для умовної компіляції (для констант типу boolean) при використанні з оператором if.
-

```
private static final double PI = 3.14159;
```

Приклад використання статичних полів

```
public class Bicycle {
    ...
    private static int numberOfBicycles = 0;
    private int id;


    public Bicycle(i){
        // increment number of Bicycles
        // and assign ID number
        id = ++numberOfBicycles;
    }

    // method to return the ID instance variable
    public int getID() {
        return id;
    }
    ...
}
```

Особливості використання статичних методів

- Методи примірника класу можуть мати доступ до полів і методів екземпляра безпосередньо.
- Методи примірника можуть безпосередньо звертатися до статичних змінних.
- Статичні методи можуть мати прямий доступ до статичних методів і полів.
- Статичні методи **НЕ** мають прямого доступу до методів і полів екземпляра класу. Для цього необхідне посилання на об'єкт класу.
- Статичні методи **НЕ** можуть використовувати this

Ключове слово final

- Ключове слово final (фінальний) може бути застосовано до опису змінної, методу або класу.
- Фінальне поле класу повинно бути ініційоване при описі, або в конструкторі класу (а статичне поле - в статичному блоці ініціалізації) і далі його значення не може бути змінено.
- Значення локальних змінних, а також параметрів методу, позначених ключовим словом final, не можуть бути змінені після присвоєння.
- Метод класу, відзначений словом final, не може бути перевизначений при спадкуванні.
- Фінальний клас не може мати спадкоємців взагалі.