



# Java Essential

Основи об'єктно-орієнтованого програмування



# Java Essential

Після уроку обов'язково



Повторіть цей урок у форматі відео на [ITVDN.com](http://itvdn.com)

Доступ можна отримати через керівництво вашого навчального центру



Перевірте, як Ви засвоїли цей матеріал на [TestProvider.com](http://testprovider.com)

## Основи об'єктно-орієнтованого програмування

# Об'єктно-орієнтований підхід

## Парадигми ООП

ІНКАПСУЛЯЦІЯ

УСПАДКУВАННЯ

ПОЛІМОРФІЗМ

АБСТРАКЦІЯ

# Парадигми ООП

## Інкапсуляція

**Інкапсуляція** — механізм мови, що дозволяє обмежити доступ одних компонентів програми до інших.

У Java наявні такі модифікатори доступу :

- **private** — члени класу доступні лише всередині класу;
- **default** (package-private, модифікатор за замовчуванням) — членів класу видно всередині пакета;
- **protected** — члени класу доступні всередині пакета та в спадкоємцях;
- **public** — члени класу доступні всім.

```
class A {  
    private int field1;  
    int field2;  
    protected int field3;  
    public int field4;  
}
```

# Парадигми ООП

## Успадкування

**Успадкування** – механізм об'єктно-орієнтованого програмування (поряд з інкапсуляцією, поліморфізмом та абстракцією), що дає змогу описати новий клас на основі вже наявного (батьківського), водночас властивості та функціональність батьківського класу запозичуються новим класом.

```
class A {  
    public int field1;  
    public void method() {  
        /* ... */  
    }  
}
```

```
class B extends A {  
    public int field2;  
}
```

```
public static void main(String[] args) {  
    B b = new B();  
    b.field1 = 5;  
    b.field2 = 8;  
    b.method();  
}
```

# Парадигми ООП

## Поліморфізм

**Поліморфізм** – можливість об'єктів з однаковою специфікацією мати різну реалізацію.

Форми поліморфізму:

1. Ad-hoc поліморфізм.

2. Класичний (примусовий) поліморфізм:

- використання перевизначених членів(@Override);
- приведення типів.

У разі одночасного використання двох форм класичного поліморфізму перша форма нейтралізує другу (домінує над другою).

# Парадигми ООП

## Абстракція

**Абстракція** в ООП – це надання об'єктові характеристик, які відрізняють його від усіх інших об'єктів, чітко визначаючи його концептуальні межі.

**Абстрагування** в ООП – це спосіб виділити набір значущих показників об'єкта, виключаючи з розгляду незначущі. Відповідно, абстракція – це набір усіх таких характеристик.



# Оголошення класу

## Зміст класу

```
public class Car {
```

```
    public String model;  
    private int maxSpeed;  
    private int speed;
```

```
    public Car(String model, int maxSpeed, int speed) {  
        this.model = model;  
        this.maxSpeed = maxSpeed;  
        this.speed = speed;  
    }
```

```
    public int getMaxSpeed() {  
        return maxSpeed;  
    }
```

```
    public void setMaxSpeed(int maxSpeed) {  
        this.maxSpeed = maxSpeed;  
    }
```

```
}
```

Оголошення класу з ім'ям Car

Оголошення полів класу Car

Оголошення конструктора класу Car

Оголошення методів класу Car

# Члени класу

## Поля та методи

**Поле класу**, або **Атрибут** (class field) – змінна, яка пов'язана з класом чи об'єктом. Усі дані об'єкта зберігаються у його полях. Доступ до полів здійснюється за їхнім ім'ям.

```
private int speed;  
speed = 100;
```

**Метод класу** (class method) – процедури та функції, які пов'язані з класом. Визначають дії, які можна виконувати над об'єктом такого типу та які сам об'єкт може виконувати.

```
public int getSpeed(){  
    return speed;  
}
```

# Члени класу

## Конструктори

**Конструктор** – спеціальний метод створення екземпляра класу (ініціалізації об'єкта).

Ім'я конструктора відповідає імені класу.

```
public Car(String model, int maxSpeed, int speed) {  
    this.model = model;  
    this.maxSpeed = maxSpeed;  
    this.speed = speed;  
}
```

Якщо програміст явно не задав конструктор у класі, JVM під час компіляції створить **конструктор за замовчуванням**.

```
public Car() { }
```

# Об'єкт

## Створення об'єкта

Щоб створити об'єкт, використовується ключове слово **new**.

```
new Car();
```

Водночас JVM викликає конструктор класу Car і збирає екземпляр класу Car. Щоб використати об'єкт надалі, створимо покажчик на нього.

```
Car car = new Car();
```

Тепер через посилання ми можемо отримати доступ до полів і методів об'єкта.

```
car.model;  
car.setMaxSpeed(100);
```

Значення полів характеризують **СТАН** екземпляра класу (об'єкта).

Набір методів характеризує **ПОВЕДІНКУ** об'єкта.

# Java Essential

Q&A

# Інформаційний відеосервіс для розробників програмного забезпечення

