



Lesson 5

01.05.2023

```
public class Ex1 {
    static int a = 1111;

    static {
        System.out.println("static");
        a = a-- - --a;
    }

    {
        System.out.println("non static");
        a = a++ + ++a;
    }

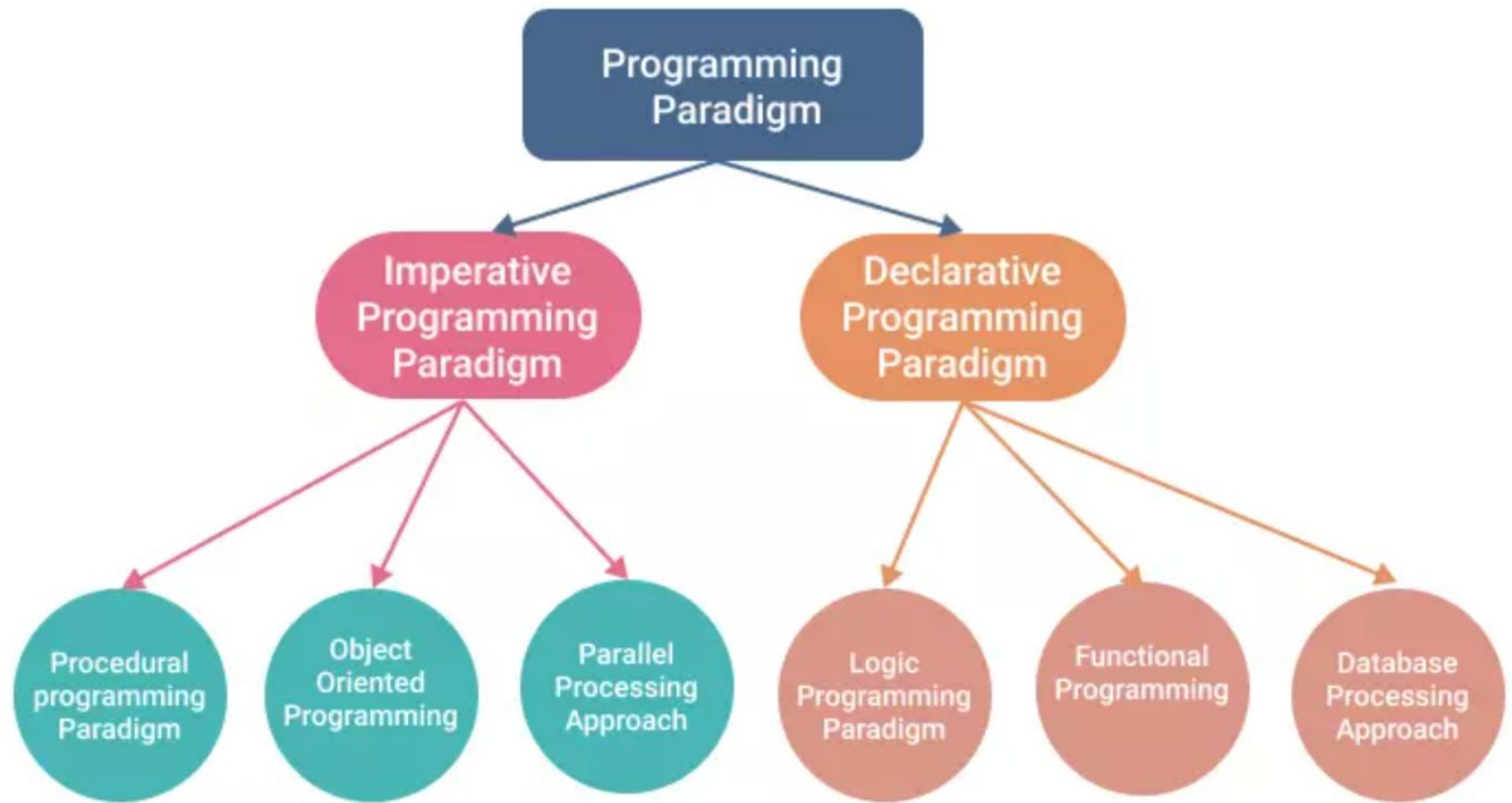
    public static void main(String[] args) {
        System.out.println(a);
    }
}
```

```
public class Ex2 {  
    public static void main(String[] args) {  
        Integer i1 = 128;  
        Integer i2 = 128;  
        System.out.println(i1 == i2);  
  
        Integer i3 = 127;  
        Integer i4 = 127;  
        System.out.println(i3 == i4);  
    }  
}
```



```
public class Ex3 {  
    public static void show() {  
        System.out.println("Static method called");  
    }  
  
    public static void main(String[] args) {  
        Ex3 obj = null;  
        obj.show();  
    }  
}
```

```
public class Ex4 {  
    static int method1(int i) {  
        return method2(i *= 11);  
    }  
    static int method2(int i) {  
        return method3(i /= 11);  
    }  
    static int method3(int i) {  
        return method4(i -= 11);  
    }  
    static int method4(int i) {  
        return i += 11;  
    }  
    public static void main(String[] args) {  
        System.out.println(method1(11));  
    }  
}
```



IMPERATIVE vs DECLARATIVE

WHAT DOES IT MEAN?



Follow steps

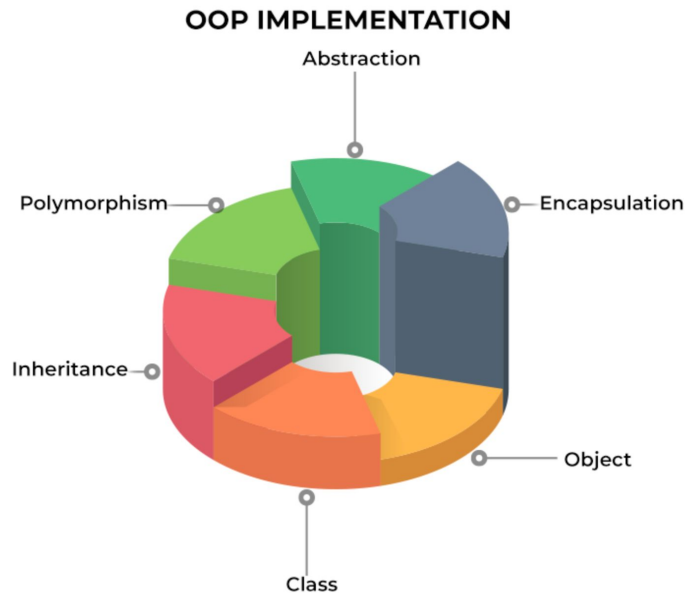
- ① Get a pot
- ② Pour water in pot
- ③ Heat pot over stove
- ④ Wait 5 minutes

Define what we need

- ① Fill kettle so it contains 1L water
- ② Heat water in kettle until temperature reaches 100°C

OOP

У цій парадигмі програмування програма розбивається на об'єкти – структури даних, що складаються з полів, що описують стан, та методів – підпрограм, що застосовуються до об'єктів для зміни чи запиту їх стану. Більшість об'єктно орієнтованих парадигм для опису об'єктів використовуються класи, об'єкти вищого порядку, що описують структуру та операції, пов'язані з об'єктами



class

car

methods

refuel() getFuel
setSpeed() getSpeed()
drive()

attributes

fuel
maxspeed

Encapsulation

Data Security

Inheritance

Code Reusability

OOP Program

Class

Object

Polymorphism

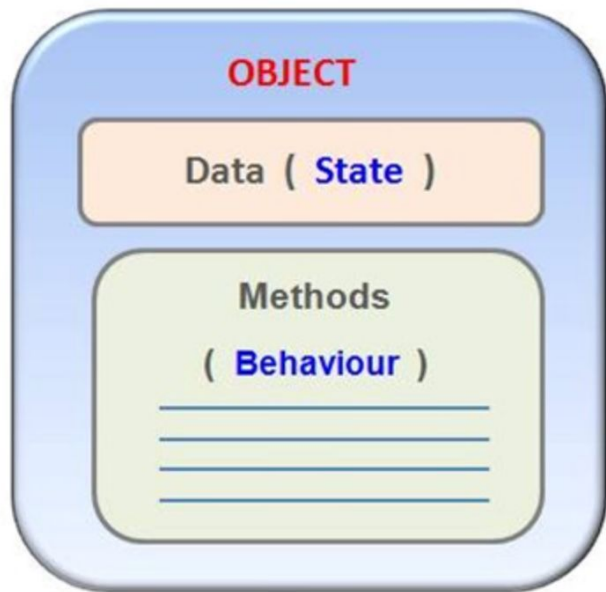
Code Reusability

Abstraction

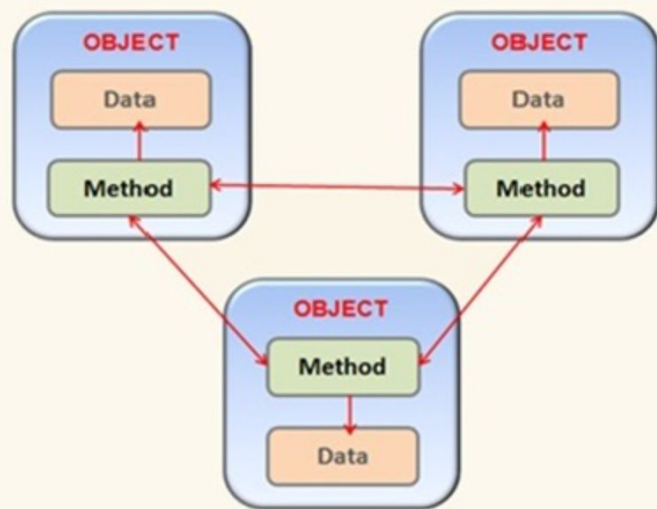
Hides Complexity



OOP - Concept Of Object



OPP - Program Organization





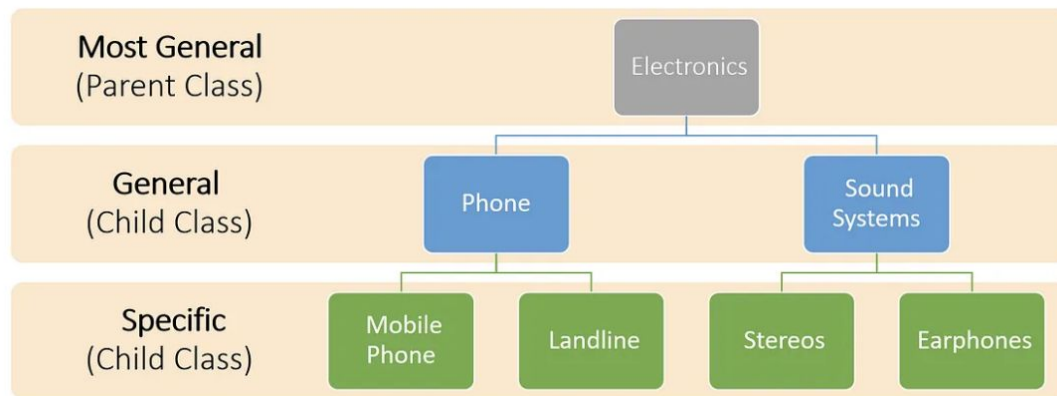
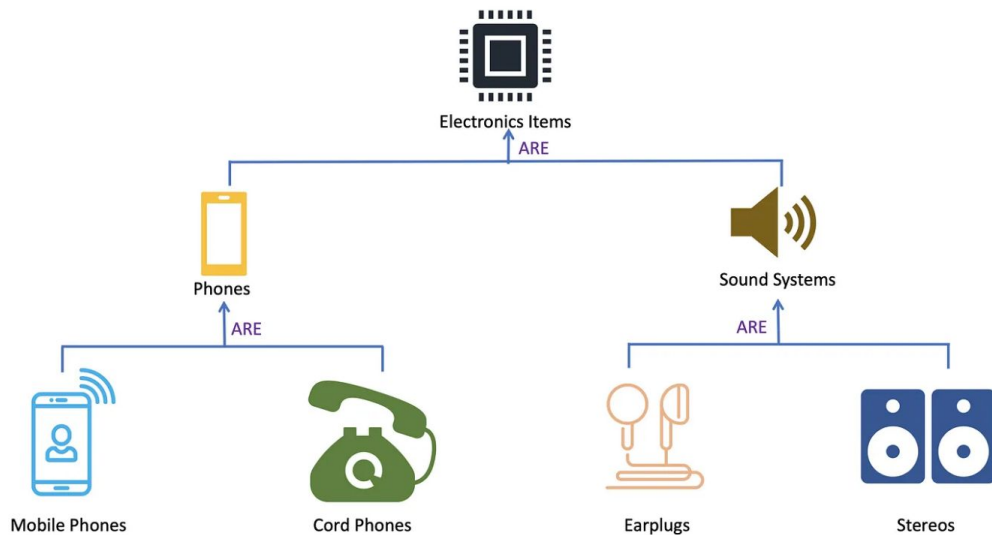
Три принципи ООП

Інкапсуляція це властивість системи, що дозволяє об'єднати дані та методи в класі, та приховати деталі реалізації від користувача.

Наслідування це властивість системи, що дозволяє описати новий клас на основі вже існуючого з частково або повністю запозиченої функціональністю

Поліморфізм – один інтерфейс, безліч методів”. Реалізації поліморфізму в мові Java - це навантаження та перевизначення методів, інтерфейси

Абстракція даних це спосіб виділити набір значних показників об'єкта, крім розгляду не значущі. Відповідно, абстракція – це набір всіх таких показників.

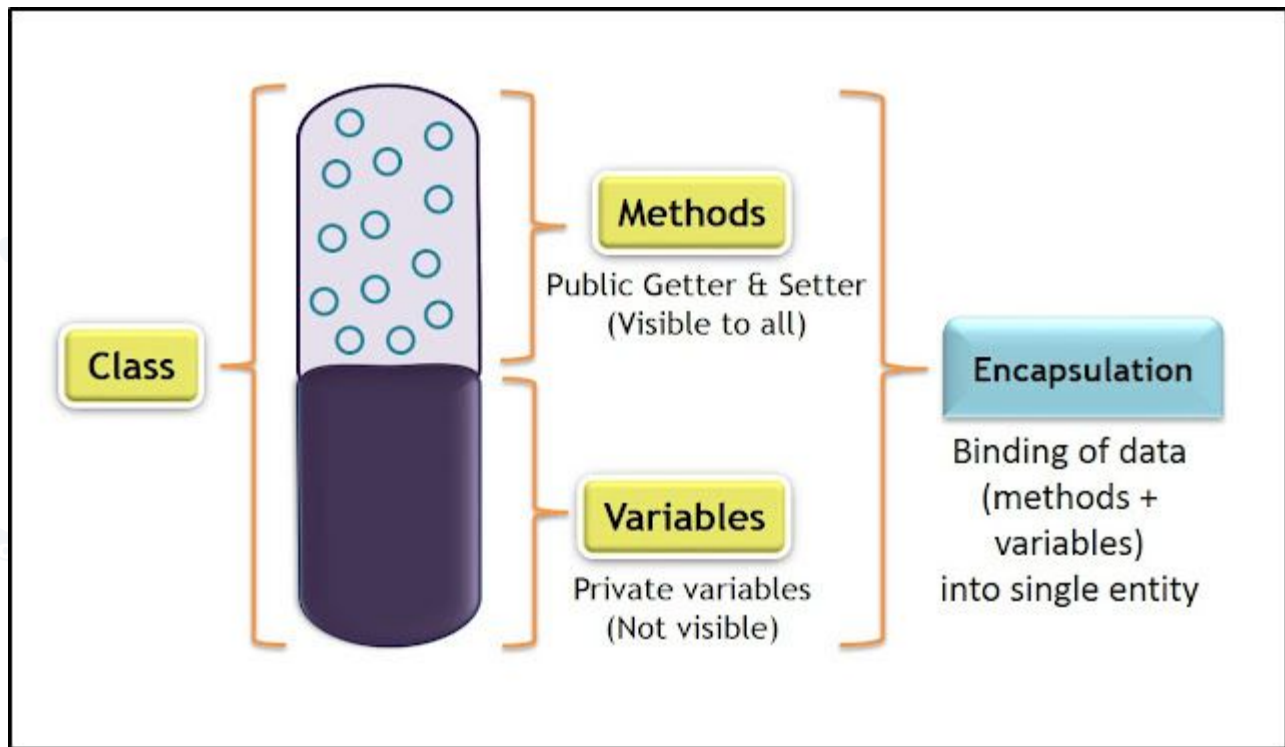


Inheritance



you can create new type of animal
changing or adding properties





Incapsulation



every animal eats
and then poop

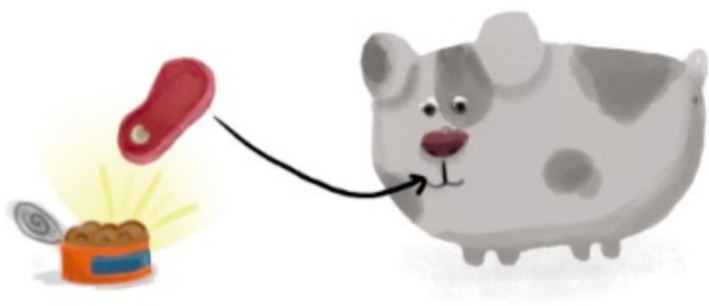




Polymorphism



each animal can eat
its own type of food





Class vs. Object



OBJECT	CLASS
Object is an instance of a class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen, table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocate memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newInstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.



Breed: Bulldog
Size: large
Colour: light gray
Age: 5 years

Dog1Object



Breed: Beagle
Size: large
Colour: orange
Age: 6 years

Dog2Object



Breed: German Shepherd
Size: large
Colour: white & orange
Age: 6 years

Dog3Object

Dog

Fields

Breed
Size
Colour
Age

Methods

Eat()
Run()
Sleep()
Name()



```
Box myBox;
```



null

```
myBox = new Box();
```



Box

double height;

double depth;

double width;

Heap

stack

b1

b2

Heap

Box

double height;

double depth;

double width;



У мові Java під час проектування класів прийнято обмежувати рівень доступу до змінним за допомогою модифікатора **private** і звертатися до них через гетери та сетери.

Існують правила оголошення таких методів, розглянемо їх:

- Якщо властивість НЕ типу `boolean`, префікс геттера має бути `get`. Наприклад: **getName()** - це коректне ім'я геттера для змінної `name`.
- Якщо властивість типу `boolean`, префікс імені геттера може бути `get` або `is`. Наприклад, **getPrinted()** або **isPrinted()** обидва є коректними іменами для змінних типу `boolean`.
- Ім'я сеттера повинне починатися з префікса `set`. Наприклад, **setName()** коректне ім'я для змінної `name`.
- Для створення імені геттера або сеттера, перша літера властивості має бути змінена на велику та додана до відповідного префіксу (`set`, `get` або `is`).
- Сеттер має бути `public`, повертати `void` тип і мати параметр відповідний типу змінної.
- Геттер метод повинен бути `public`, не мати параметрів методу, і повертати значення відповідне типу властивості



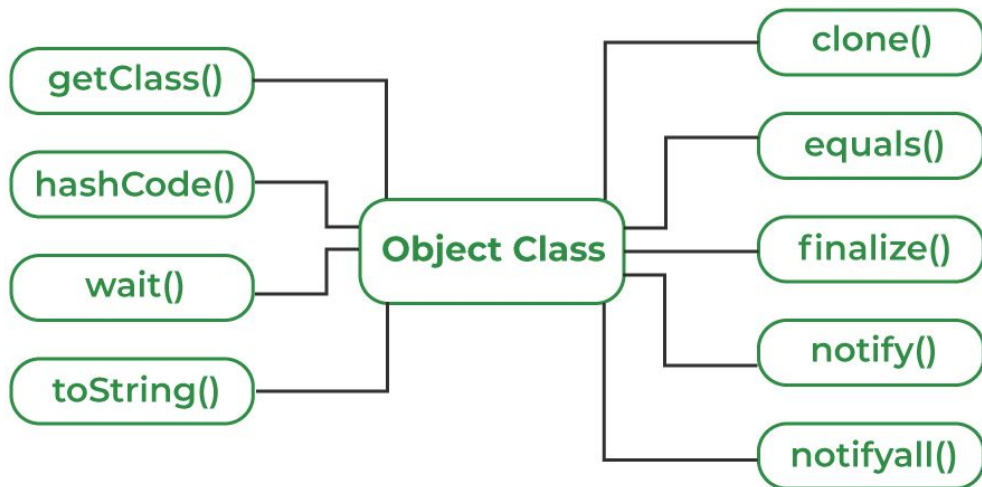
Клас Object

Є суперкласом для всіх класів (включаючи масиви)

Змінна цього типу може посилатися будь-який об'єкт (але не змінну примітивного типу)

Його методи успадковуються усіма класами

Реалізує базові операції з об'єктами



Method	Purpose
Object clone()	Creates a new object that is the same as the object being cloned.
boolean equals(Object <i>object</i>)	Determines whether one object is equal to another.
void finalize()	Called before an unused object is recycled.
Class<?> getClass()	Obtains the class of an object at run time.
int hashCode()	Returns the hash code associated with the invoking object.
void notify()	Resumes execution of a thread waiting on the invoking object.
void notifyAll()	Resumes execution of all threads waiting on the invoking object.
String toString()	Returns a string that describes the object.
void wait() void wait(long <i>milliseconds</i>) void wait(long <i>milliseconds</i> , int <i>nanoseconds</i>)	Waits on another thread of execution.