Lesson 11

27.02.2025

```
public class Task1 {
    public static void main(String[] args) {
        long thatNumber = 5 ⩾ 5 ? 1+2 : 1*1;
        if(++thatNumber < 4)
            thatNumber += 1;
        System.out.println(thatNumber);
    }
}
```

```java
public class Task2 {
    public static void main(String[] args) {
        int meal = 5;
        int tip = 2;
        int total = meal + (meal > 6 ? ++tip : --tip);
        System.out.println(total);
    }
}
```

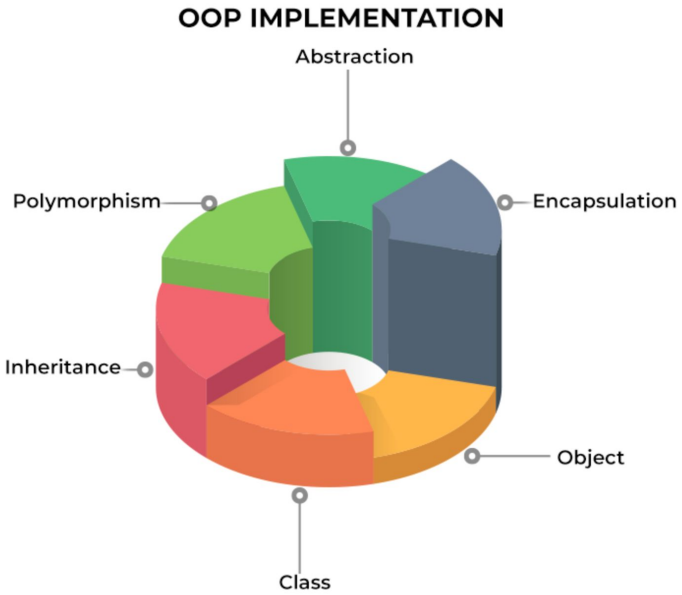
```
public class Task3 {
    public static String travel(int distance) {
        return distance<1000 ? "train" : 10;
    }
    public static void main(String[] answer) {
        System.out.print(travel(500));
    }
}
```

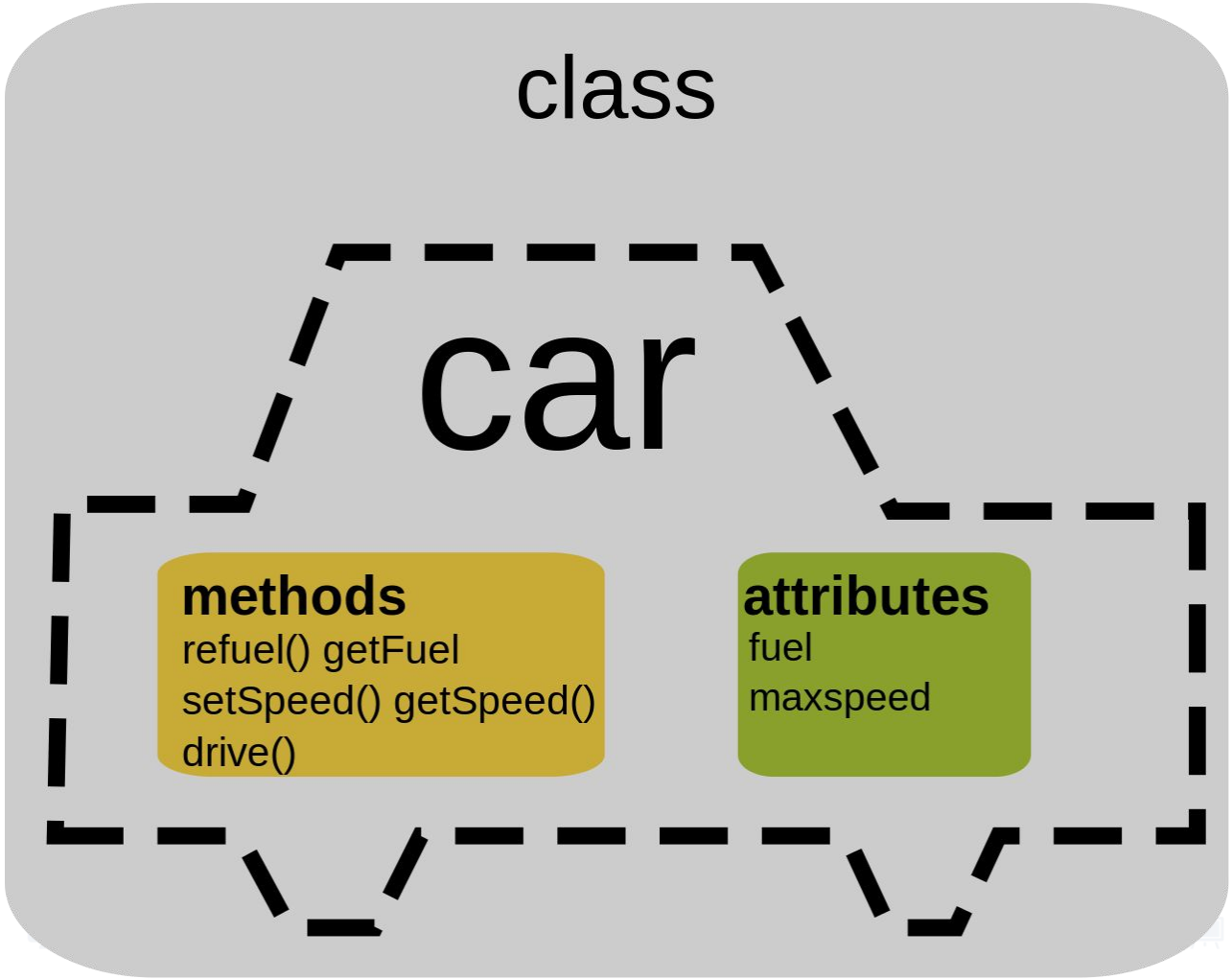
```
public class Task4 {
    public static void main(String[] args) {
        String[] days = new String[]{"Sunday", "Monday", "Tuesday",
                "Wednesday", "Thursday", "Friday", "Saturday"};
        for (int i = 0; i < days.size; i++)
            System.out.println(days[i]);
    }
}
```

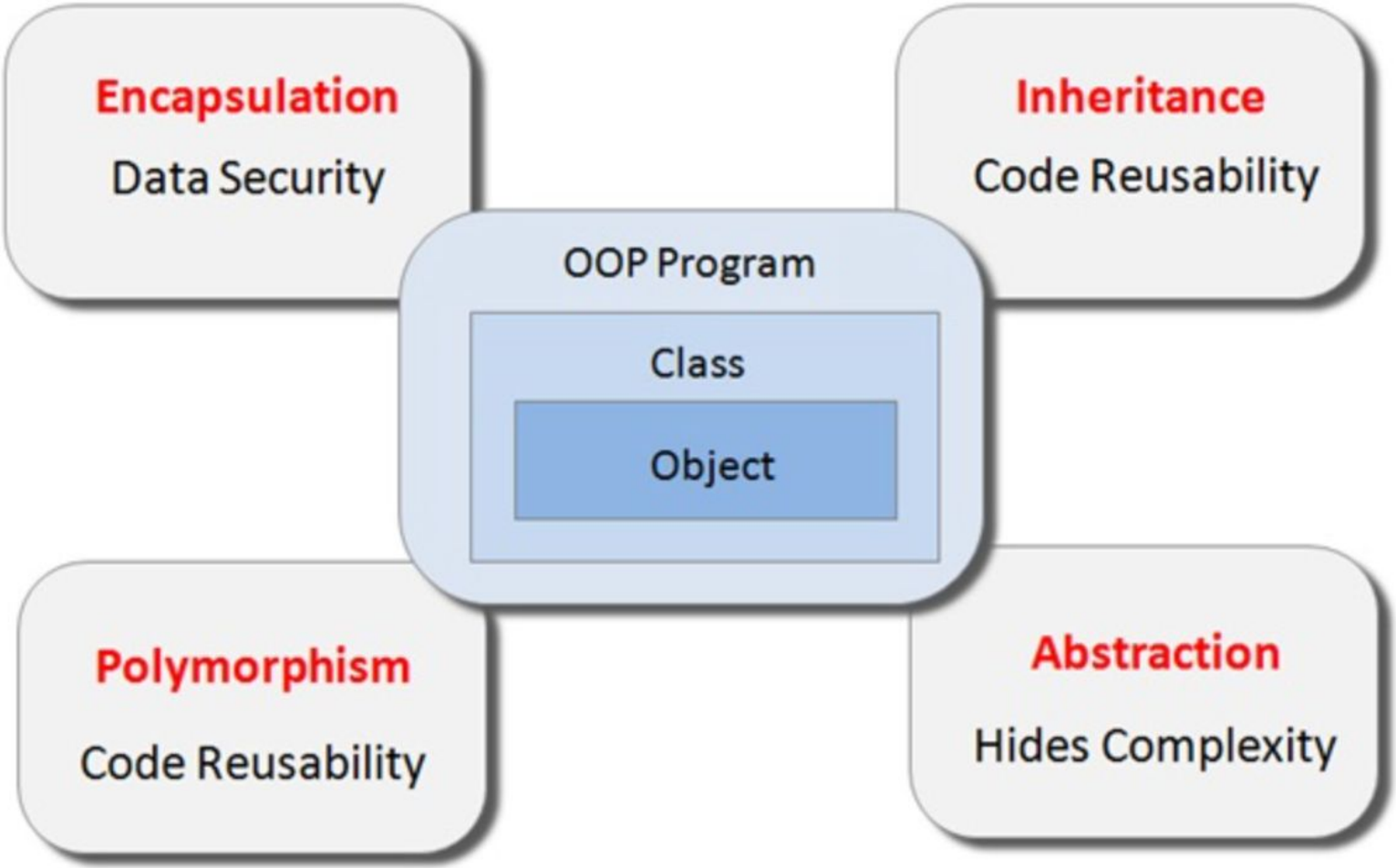
```
public class Task5 {
    public static void main(String[] args) {
        String[] os = new String[] { "Mac", "Windows", "Linux"};
        System.out.println(Arrays.binarySearch(os, "Linux"));
    }
}
```

ООР

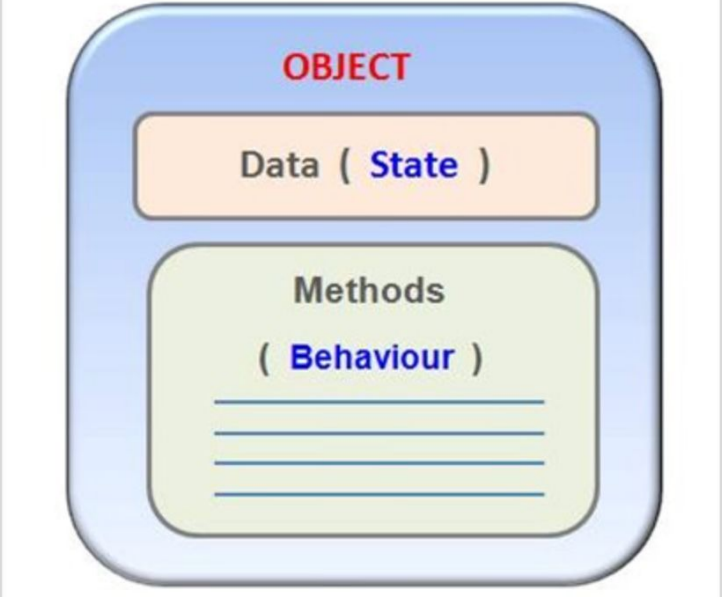
У цій парадигмі програмування програма розбивається на об'єкти – структури даних, що складаються з полів, що описують стан, та методів – підпрограм, що застосовуються до об'єктів для зміни чи запиту їх стану. Більшість об'єктно орієнтованих парадигм для опису об'єктів використовуються класи, об'єкти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



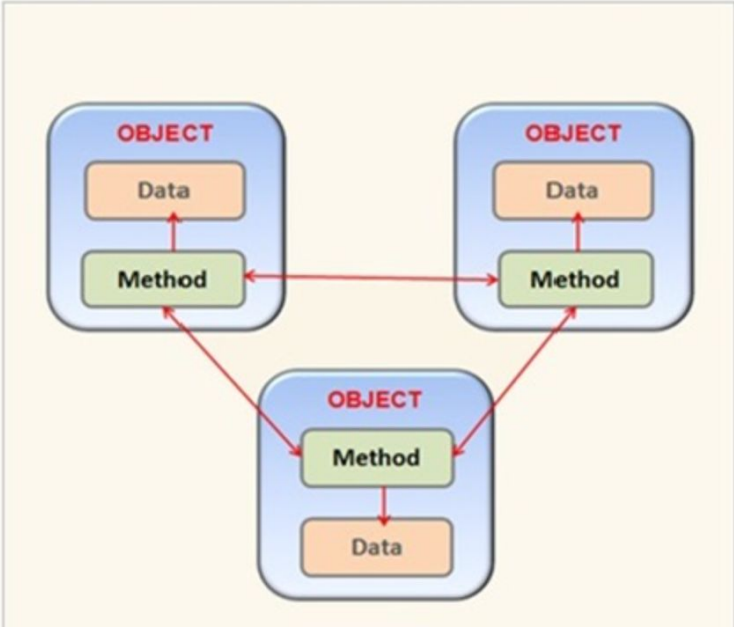

class
car
methods
refuel() getFuel
setSpeed() getSpeed()
drive()
attributes
fuel
maxspeed


Encapsulation
Data Security
Inheritance
Code Reusability
OOP Program
Class
Object
Polymorphism
Code Reusability
Abstraction
Hides Complexity

OOP - Concept Of Object



OPP - Program Organization



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.

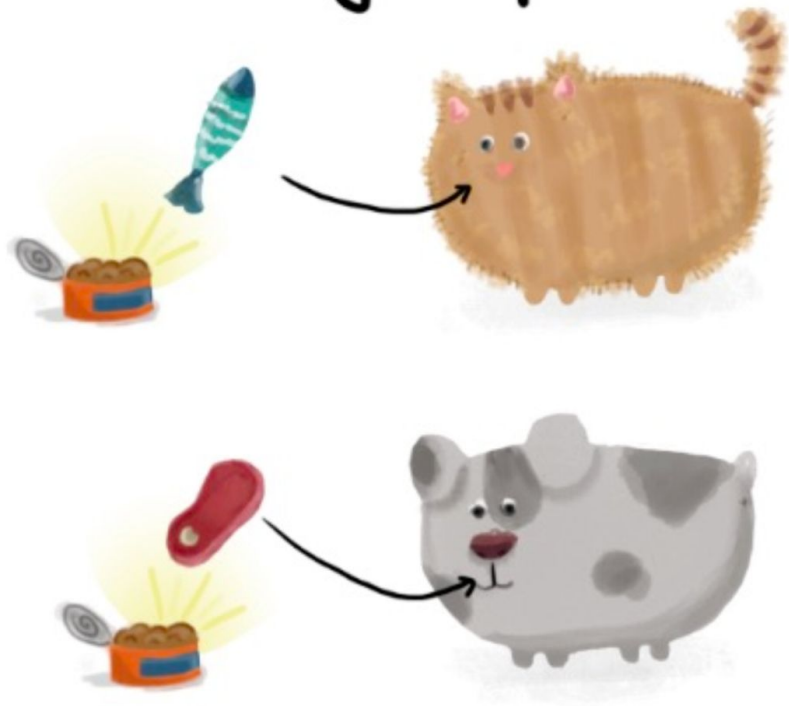
Inheritance

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 ofa 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 allocated 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

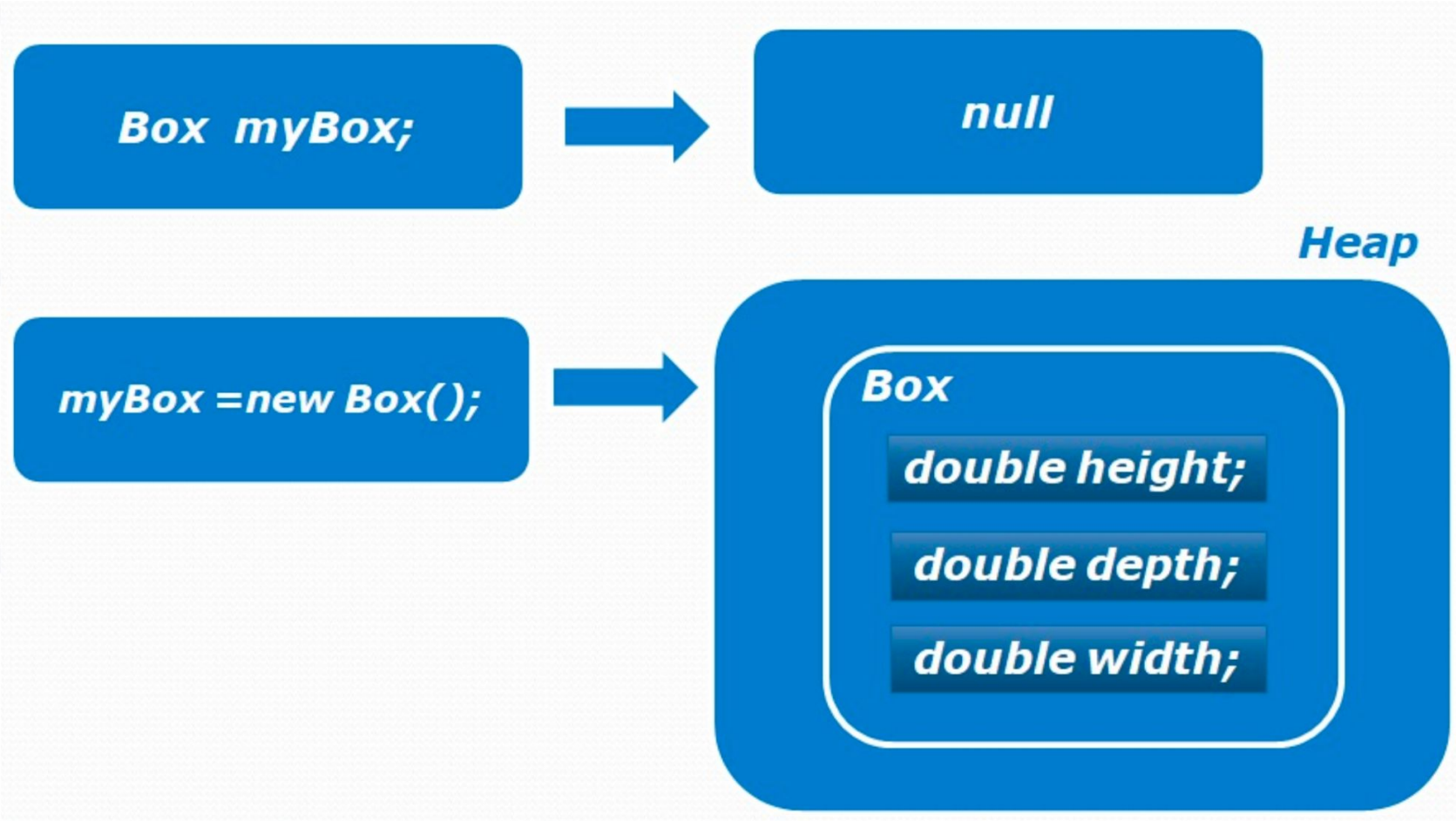
Breed: German Shepherd
Size: large
Colour: white & orange
Age: 6 years

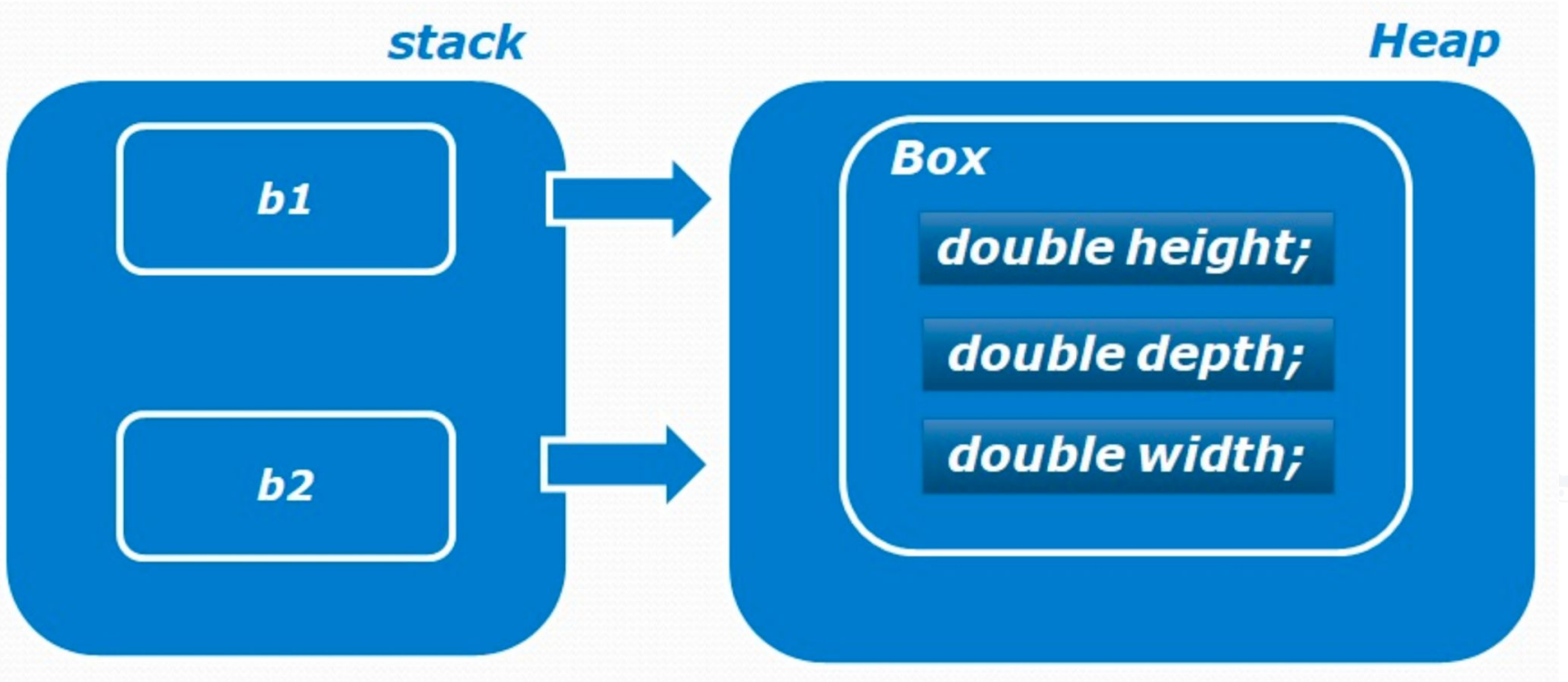
Dog3Object

Dog

Fields
Breed
Size
Colour
Age

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


Box myBox;
null
Heap
myBox =new Box();
Box
double height;
double depth;
double width;


stack
Heap
b1
b2
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

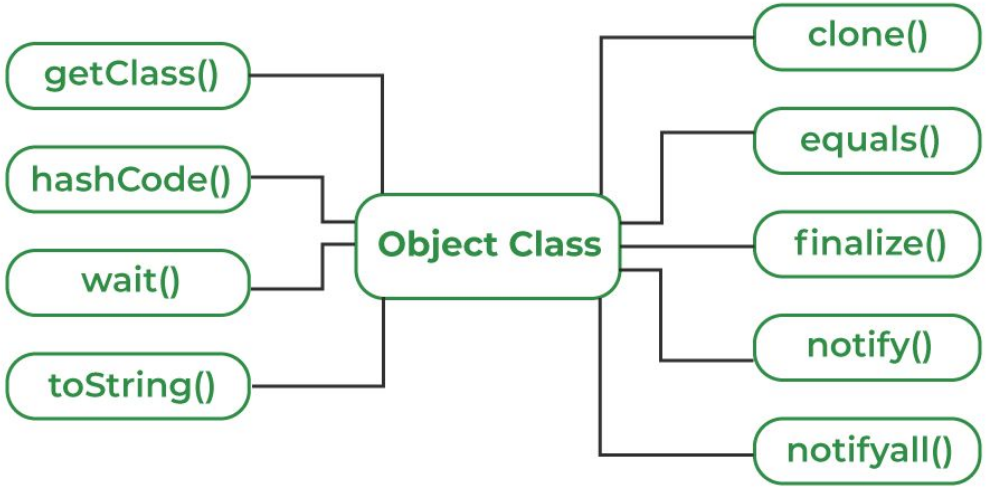
Клас Object

Є суперкласом для всіх класів (включаючи масиви)

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

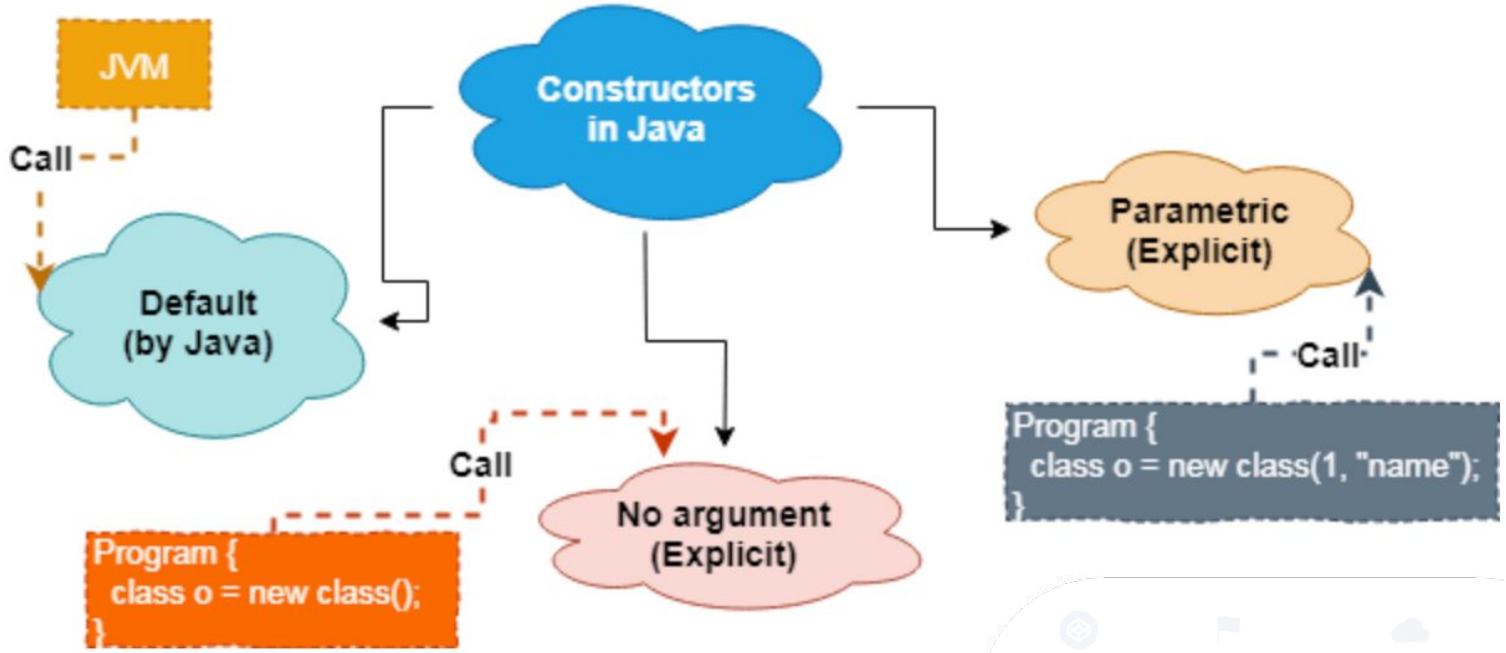
Його методи успадковуються усіма класами

Реалізує базові операції з об'єктами

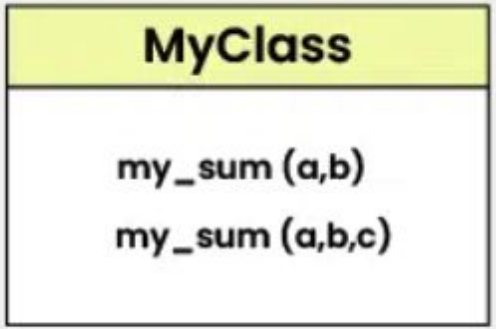


Method	Purpose
Object clone( )	Creates a new object that is the same as the object being cloned.
boolean equals(Object *object*)	Determines whether one object is equal to another.
void finalize( )	Called before an unused object is recycled.
Class<?> getClass( )	Obtains the class of an object at run time.
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 *milliseconds*) void wait(long *milliseconds*, int *nanoseconds*)	Waits on another thread of execution.

Конструктор – це метод, призначення якого полягає у створенні екземпляра класу.



Function Overloading in Programming



Function Overriding in Programming