OOPs Concept

08 October 2023 22:41

Class

Object

This keywords

Constructor
- parameterized
- non-parameterized

Constructor Overloading

Static
- Static keyword
- Static block
- Static variable
- Diff between static and non-static function

Encapsulation(Car example)
- Hiding data member of class
- How to encapsulate Data member of class
- Access Modifier in java(public ,private ,protected ,default)
- Getter and setter method
  - Needs of getter and setter
  - Uses of getter and setter method
- Price cannot be negative handled using getter and setter

Inheritance:-
- What is inheritance?
- Why inheritance
- Keywords:-
- Super/parent class
- Sub/child class
- Extends
- 4 case of inheritance

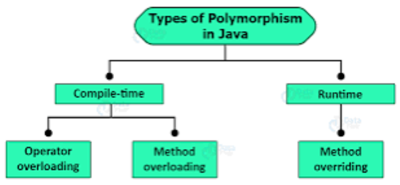
```
 /*
     * case1:- P p1=new P();

     * Case2:- P p2=new C();
     * Without Inheritance  Method-Overriding is not possible
It happens on runtime
     * Case3:- C c1=new P();

     * Case4:- C c2=new C();
 */
```

1. Single level Inheritance
2. Multilevel Inheritance

`Multiple Inheritance(not supported in java)`

`@override annotations`
`Can static variable override`

Polymorphism:-

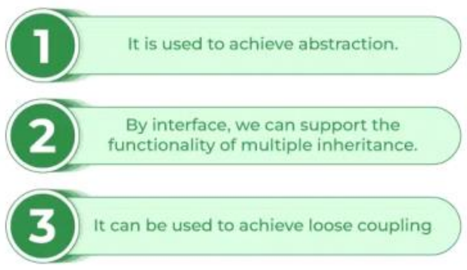


Abstraction:-

- What and Why we need abstraction?
- How we achieve abstraction
- abstract class
- abstract methods

Interface:-

The interface in Java is *a* mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritances in Java using Interface. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also **represents the IS-A relationship**.

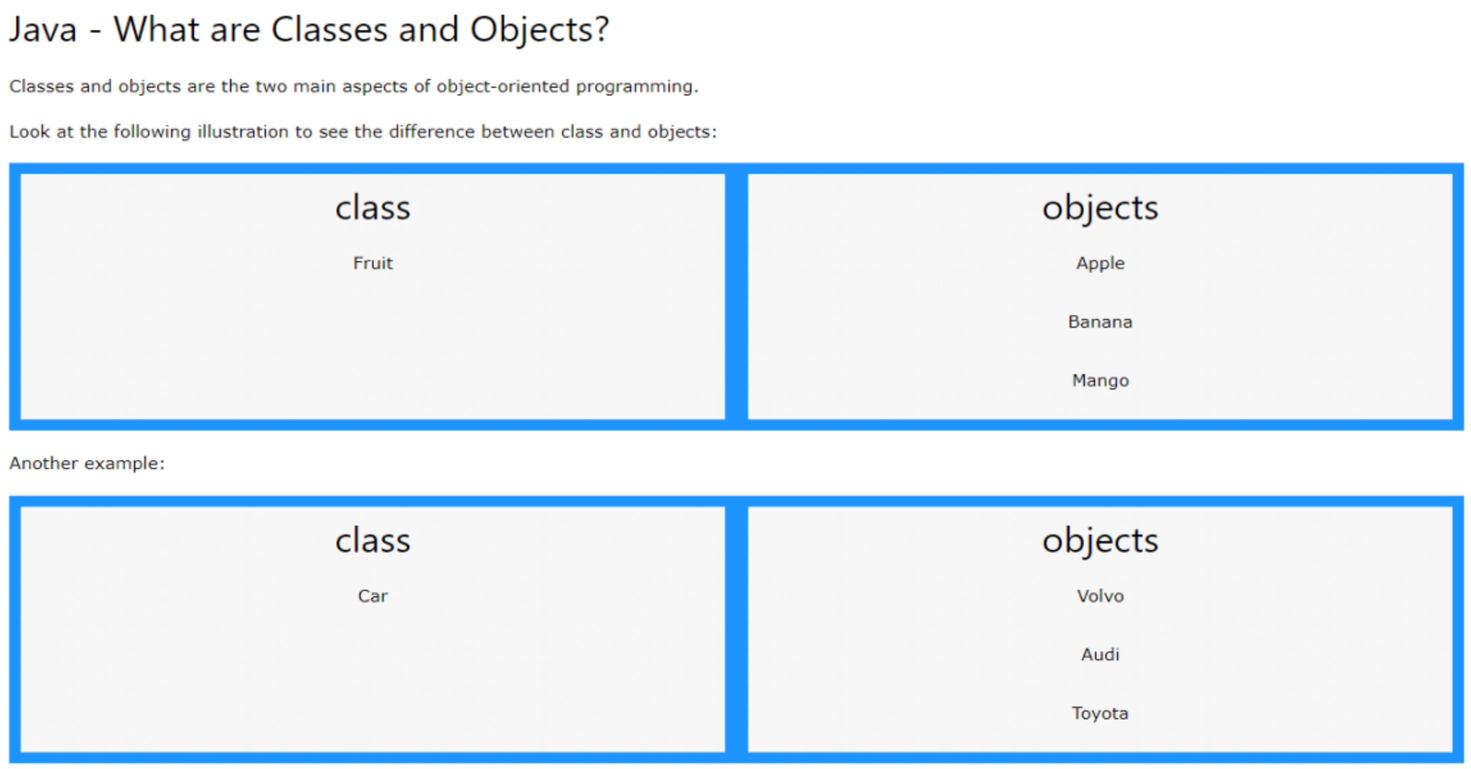

1
It is used to achieve abstraction.
2
By interface, we can support the functionality of multiple inheritance.
3
It can be used to achieve loose coupling

Syntax for Java Interfaces

```
interface {

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

To declare an interface, use the interface keyword. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static, and final by default. A class that implements an interface must implement all the methods declared in the interface.

Exception class
  - Generate exception inside function
  - Throw exception
- Exception Handling
  - Why Exception Handling?
  - Uses of Exception Handling
  - How to use try catch blocks
- Final Keyword (variable name must be CAPITAL)
  - Final variable
  - Final method
- Garbage Collector



Java - What are Classes and Objects?

Classes and objects are the two main aspects of object-oriented programming.

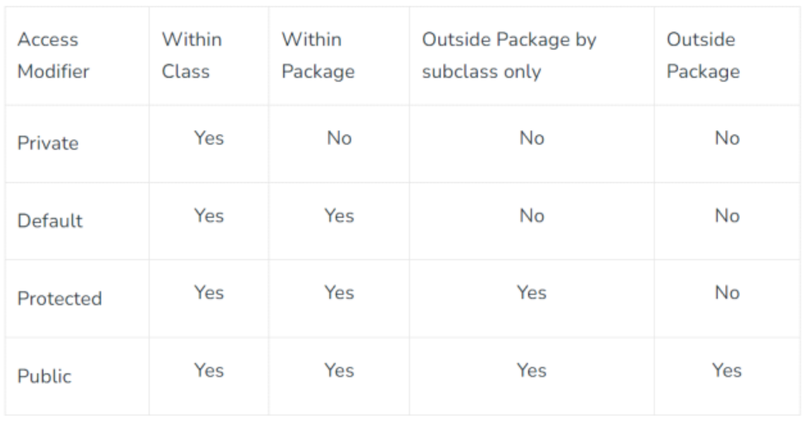
Look at the following illustration to see the difference between class and objects:

class	objects
Fruit	Apple
	Banana
	Mango

Another example:

class	objects
Car	Volvo
	Audi
	Toyota

Understanding Access Modifier



Access Modifier	Within Class	Within Package	Outside Package by subclass only	Outside Package
Private	Yes	No	No	No
Default	Yes	Yes	No	No
Protected	Yes	Yes	Yes	No
Public	Yes	Yes	Yes	Yes

Access Modifier: Defines the **access type** of the method i.e. from where it can be accessed in your application. In Java, there are 4 types of access specifiers:

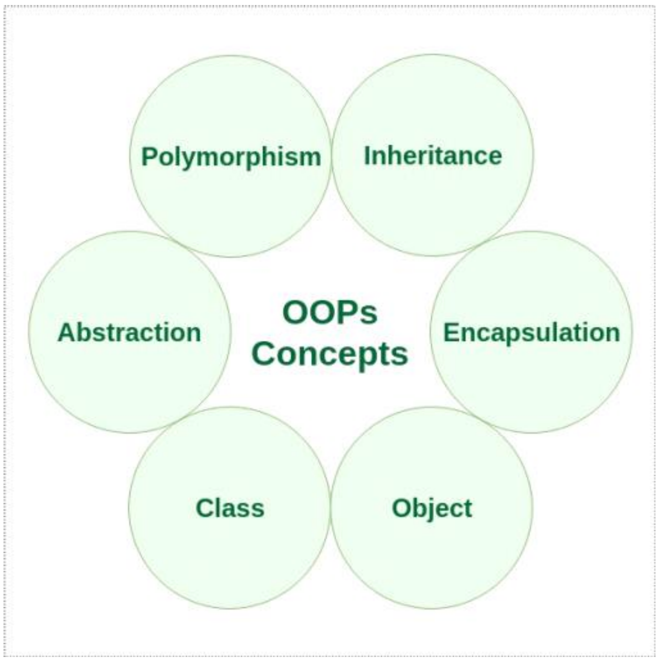
- **public:** Accessible in all classes in your application.
- **protected:** Accessible within the package in which it is defined and in its **subclass(es) (including subclasses declared outside the package)**.
- **private:** Accessible only within the class in which it is defined.
- **default (declared/defined without using any modifier):** Accessible within the same class and package within which its class is defined.

The return type: The data type of the value returned by the method or void if it does not return a value.
Method Name: The rules for field names apply to method names as well, but the convention is a little different.
Parameter list: Comma-separated list of the input parameters that are defined, preceded by their data type, within the enclosed parentheses. If there are no parameters, you must use empty parentheses ().
Exception list: The exceptions you expect the method to throw. You can

Now that we have covered the basic prerequisites, we will move on to the 4 pillars of OOPs which are as follows. But, let us start by learning about the different characteristics of an Object-Oriented Programming Language.

OOPS concepts are as follows:

1. Class
2. Object
3. Method and method passing
4. Pillars of OOPs
   - Abstraction
   - Encapsulation
   - Inheritance
   - Polymorphism
     - Compile-time polymorphism
     - Runtime polymorphism



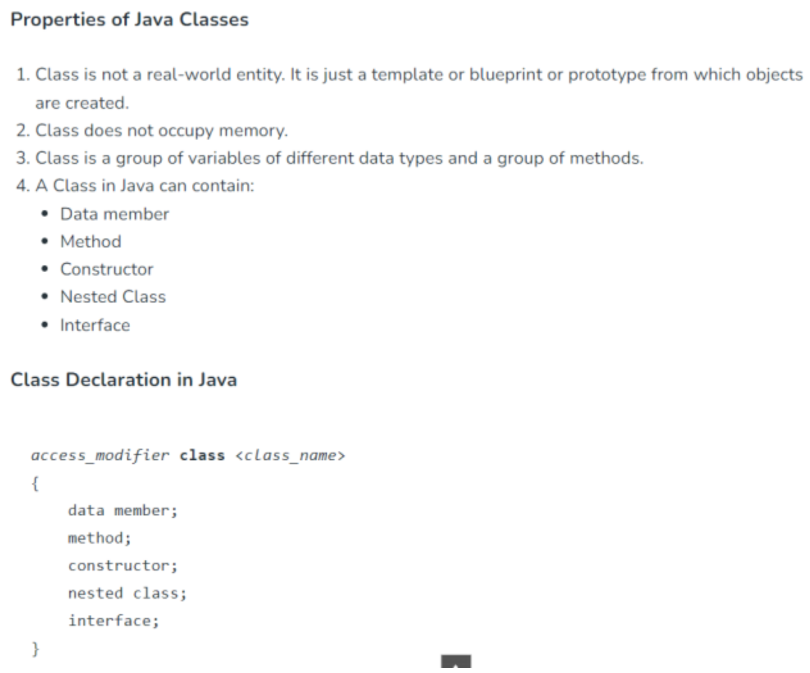
A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. Using classes, you can create multiple objects with the same behavior instead of writing their code multiple times. This includes classes for objects occurring more than once in your code. In general, class declarations can include these components in order:

1. **Modifiers**: A class can be public or have default access (Refer to this for details).
2. **Class name:** The class name should begin with the initial letter capitalized by convention.
3. **Superclass (if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
4. **Interfaces (if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
5. **Body:** The class body is surrounded by braces, { }.

class and objects one simple java program :

Java

```java
public class GFG {

    static String Employee_name;
    static float Employee_salary;

    static void set(String n, float p) {
        Employee_name  = n;
        Employee_salary  = p;
    }

    static void get() {
        System.out.println("Employee name is: " +Employee_name );
        System.out.println("Employee CTC is: " + Employee_salary);
    }

    public static void main(String args[]) {
        GFG.set("Rathod Avinash", 10000.0f);
        GFG.get();
    }
}
```

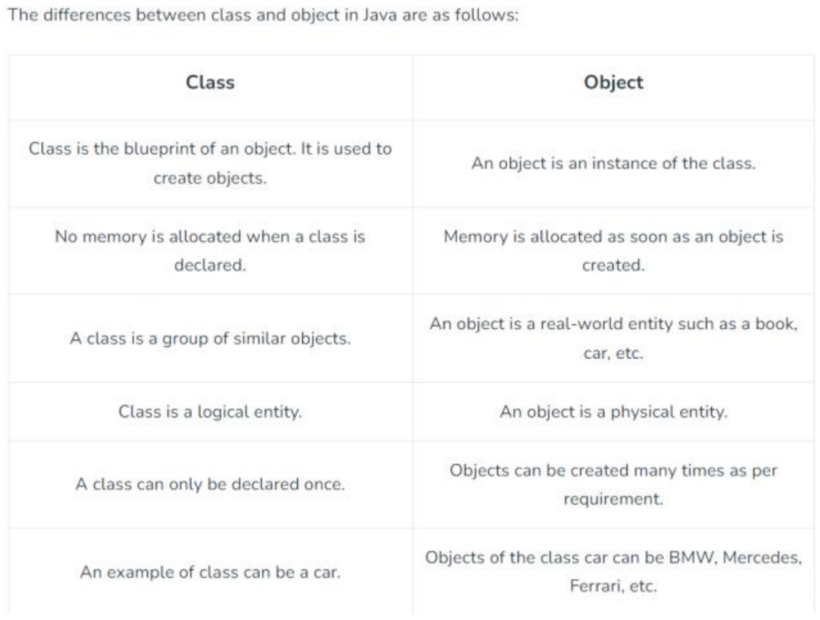


Properties of Java Classes

1. Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
2. Class does not occupy memory.
3. Class is a group of variables of different data types and a group of methods.
4. A Class in Java can contain:
   - Data member
   - Method
   - Constructor
   - Nested Class
   - Interface

Class Declaration in Java

```
access_modifier class <class_name>
{
    data member;
    method;
    constructor;
    nested class;
    interface;
}
```

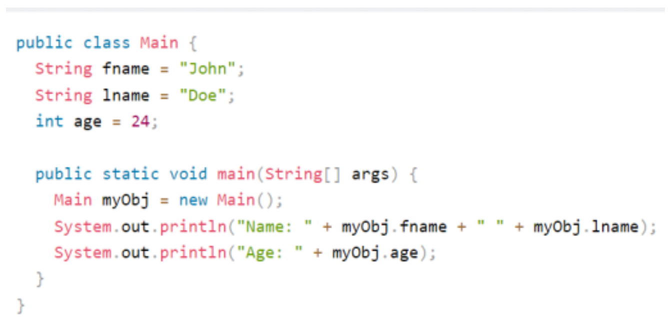


The differences between class and object in Java are as follows:

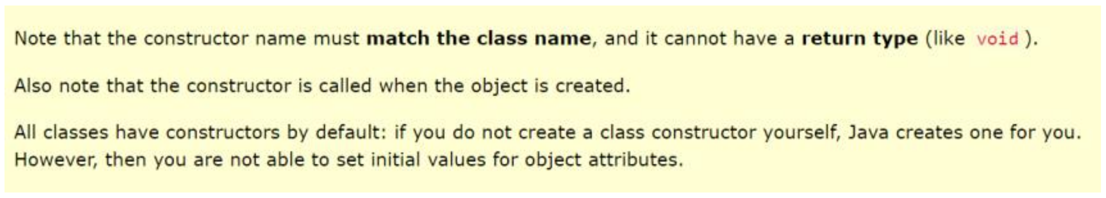
Class	Object
Class is the blueprint of an object. It is used to create objects.	An object is an instance of the class.
No memory is allocated when a class is declared.	Memory is allocated as soon as an object is created.
A class is a group of similar objects.	An object is a real-world entity such as a book, car, etc.
Class is a logical entity.	An object is a physical entity.
A class can only be declared once.	Objects can be created many times as per requirement.
An example of class can be a car.	Objects of the class car can be BMW, Mercedes, Ferrari, etc.

Final keyword

The `final` keyword is useful when you want a variable to always store the same value, like PI (3.14159...).



```
public class Main {
  String fname = "John";
  String lname = "Doe";
  int age = 24;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Age: " + myObj.age);
  }
}
```



Note that the constructor name must **match the class name**, and it cannot have a **return type** (like `void`).

Also note that the constructor is called when the object is created.

All classes have constructors by default: if you do not create a class constructor yourself, Java creates one for you. However, then you are not able to set initial values for object attributes.