Classes and Objects - Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities.

Class - 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.

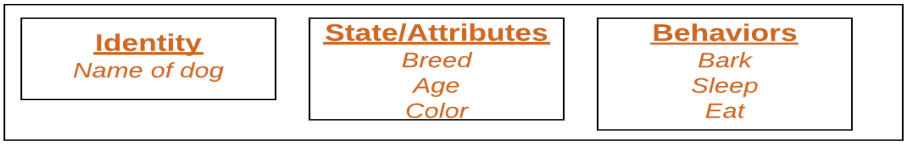
- In general, class declarations can include these components, in order:
  1. Modifiers: A class can be **public** or has **default** access.
  2. Class name: The name should begin with a initial letter (capitalized by convention).
  3. Superclass(if any): The name of the class's parent (superclass), 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 surrounded by braces, **{ }**.
- **Constructors** are used for **initializing** new objects.
- **Fields** are variables that **provides the state** of the class and its objects.
- **Methods** are used to **implement the behavior** of the class and its objects.
- There are various types of classes that are used in real time applications such as **nested classes**, **anonymous classes**, **lambda expressions**.

Object - It is a **basic unit** of Object Oriented Programming and represents the **real life entities**.

- An object consists of -

1. State: It is represented by **attributes** of an object. It also reflects the **properties** of an object.
2. Behavior: It is represented by **methods** of an object. It also reflects the **response** of an object with other objects.
3. Identity: It gives a **unique name** to an object and enables one object to interact with other objects.

- Example of an object : **dog**



Declaring Objects (Also called instantiating a class) - When an object of a class is created, the class is said to be **instantiated**.

- All the **instances** share the **attributes** and the **behavior** of the class.
- The **state** (**value** of those attributes) are **unique** for each object.
- A single class may have any number of instances.
- Example -

Dog dog_name; ***// This is just the referance variable. Object is NOT created (Memory is Not allocated***).

Initializing an object -

- The **new** operator **instantiates** a class by **allocating memory** for a new object and returning a reference to that memory.
- The new operator also **invokes** the class constructor.
- Example - Dog dog_name = new Dog(); **//** Now **memory is allocated** and the object is created.

Constructor -

- All classes have **at least one** constructor.
- If a class does not explicitly declare any, the **Java compiler** automatically provides a **no-argument** constructor, also called the **default constructor**.
- This default constructor calls the **class parent's no-argument** constructor (as it contain only one statement i.e super();), or the Object class constructor if the class has no other parent (as Object class is parent of all classes either directly or indirectly).
- If Parent Class doesn't have any no-argument constructor, we need to explicitally call its parametrized constructor inside Child Class constructor.

Anonymous objects -Anonymous objects are the objects that are **instantiated** but are **not stored** in a **reference** variable.

- They are used for **immediate method calling**.
- They will be **destroyed** after method calling.
- Example: FileInputStream file = new FileInputStream(**new File(filename)**); // **File** object is Anonymous.

Methods in Java - A method is a **collection of statements** that perform **some specific task** and **return** the **result** to the caller. They allow us to reuse the code without retyping the code.

- In general, method declarations has **six** components :

  1. Access Modifiers 2. Return type
  3. Method name 4. Parameter's list
  5. Exception list 6. Method body

- Method signature: It consists of the **method name** and a **parameter list** (**number** of parameters, **type** of the parameters and **order** of the parameters). The **return type** and **exceptions** are **not** considered as part of it.

- A method **returns** to the code that invoked it when:

  1. It **completes** all the statements in the method. 2. It reaches a **return** statement.
  3. **Throws** an exception.

Memory allocation for methods calls - Methods calls are implemented through **stack**. Whenever a method is called a stack frame is created within the stack area and after that the arguments passed to and the local variables and value to be returned by this called method are stored in this stack frame and when execution of the called method is finished, the allocated stack frame would be deleted. There is a stack pointer register that tracks the top of the stack which is adjusted accordingly.

Access Modifiers – Defines access type of the method i.e. from where it can be accessed in your application.

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

- **Default** type has **more restriction** than **protected** type.