# Introduction to Java

By Sanjana Bandara

# Content

- Object Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- OOPs Misc

# Object Class

# 01. Java OOP Concepts

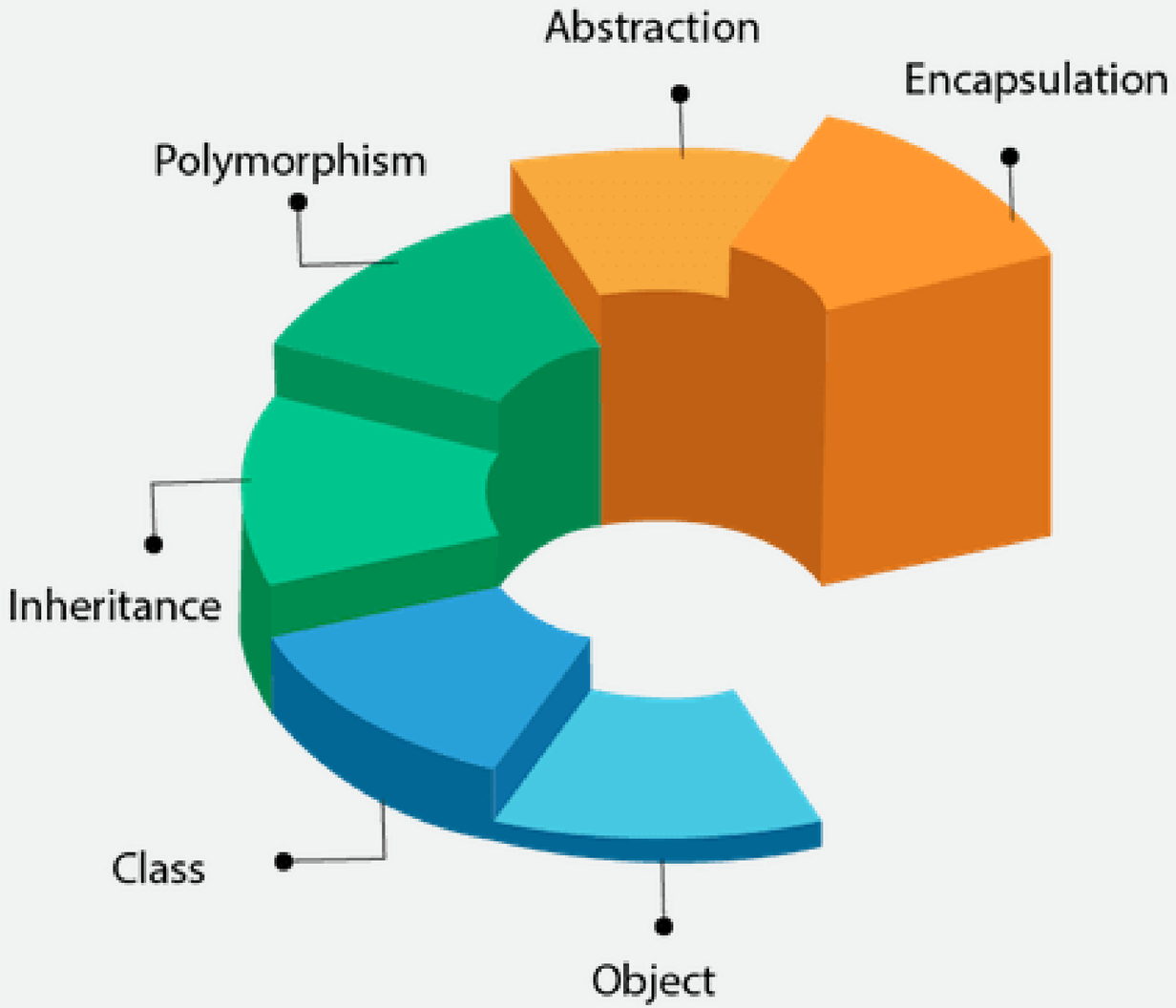
# Java OOPs Concepts

- Object means a real-world entity such as a pen, chair, table, computer, watch, etc.
- **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.
- Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.
- The popular object-oriented languages are Java, C#, PHP, Python, C++, etc.

- Concepts
  - Object
  - Class
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation

- Terms
  - Coupling
  - Cohesion
  - Association
  - Aggregation
  - Composition

# OOPs (Object-Oriented Programming System)



# Object

- Any entity that has a state and behavior is known as an Object
- Can be physical or logical
- An object is an instance of a class
- **Example:** A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

# Class

- The collection of objects is called a class
- Logical entity.
- Defined as a blueprint from which you can create an individual object

# Inheritance

- When one object acquires all the properties and behaviors of a parent object, it is known as inheritance.
- Provides code reusability

# Polymorphism


- If one task is performed in different ways, it is known as polymorphism.
- Method overloading and method overriding to achieve polymorphism.
- Speak something; for example, a cat speaks meow, a dog barks woof, etc.

# Abstraction

- Hiding internal details and showing functionality is known as abstraction.
- In Java, we use abstract class and interface to achieve abstraction.

# Encapsulation

- Binding (or wrapping) code and data together into a single unit are known as encapsulation.
- For example, a capsule is wrapped with different medicines.

Capsule

# Coupling

- Coupling refers to the knowledge or information or dependency of another class.
- If a class has the details information of another class, there is strong coupling.

# Cohesion

- Cohesion refers to the level of a component which performs a single well-defined task.
- A single well-defined task is done by a highly cohesive method.
- The weakly cohesive method will split the task into separate parts.

# Association

- The relationship between the objects.
- One object can be associated with one object or many objects.
- Types:
  - One to One
  - One to Many
  - Many to One
  - Many to Many

# Aaareaation

- Aggregation is a way to achieve Association.
- It represents the weak relationship between objects.
- It is also termed a **has-a** relationship in Java.
- It is another way to reuse objects.

# Composition

- A way to achieve Association.
- There is a strong relationship between the containing object and the dependent object.
- It is the state where containing objects do not have an independent existence.
- If you delete the parent object, all the child objects will be deleted automatically.

# Advantaaes of OOPs over Procedure-oriented proarammina lanauaae

- OOPs makes development and maintenance easier, whereas, in a procedure-oriented programming language, it is not easy to manage if code grows as project size increases.
- OOPs provides data hiding, whereas, in a procedure-oriented programming language, global data can be accessed from anywhere.
- OOPs provides the ability to simulate real-world event much more effectively. We can provide the solution to real word problems if we use the Object-Oriented Programming language.

# 02. Java Namina Convention

# Class

- It should start with the uppercase letter.
- It should be a noun such as Color, Button, System, Thread, etc.
- Use appropriate words, instead of acronyms.
- Example: public class Employee{}

# Interface

- It should start with the uppercase letter.
- It should be an adjective such as Runnable, Remote, ActionListener.
- Use appropriate words, instead of acronyms.
- Example: interface Printable {}

# Method

- It should start with a lowercase letter.
- It should be a verb such as main(), print(), println().
- If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().
- Example: public class Employee{ void draw(){} }

# Variable

- It should start with a lowercase letter such as id, or name.
- It should not start with special characters like & (ampersand), $ (dollar), _ (underscore).
- If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, or lastName.
- Avoid using one-character variables such as x, y, z.

# Packaae

- It should be a lowercase letter such as java, lang.
- If the name contains multiple words, it should be separated by dots (.)
- such as java.util, java.lang.

# Constant

- It should be in uppercase letters such as RED, YELLOW.
- If the name contains multiple words, it should be separated by an underscore(_) such as MAX_PRIORITY.
- It may contain digits but not as the first letter.

# 03. Objects and Classes in Java

# What is an Object

- Has three characteristics
  - **State:** represents the data (value) of an object.
  - **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
  - **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.
- Object Definitions
  - Real-world entity
  - Runtime entity
  - is an entity which has state and behavior
  - is an instance of a class

# What is a Class

- A class in Java can contain:
  - Fields
  - Methods
  - Constructors
  - Blocks
  - Nested class and interface
- Syntax to declare a class:

```
class<className>{
field;
method;
}
```

- 3 Ways to initialize objects.
  - By reference variable
  - By method
  - By constructor

# Practicles

# 04. Constructor in Java

# Constructors in Java

- In Java, a constructor is a block of codes similar to the method.
- It is called when an instance of the class is created. At the time of calling the constructor, memory for the object is allocated in the memory.
- It is a special type of method which is used to initialize the object.
- Rules for creating Java constructor
  - The constructor name must be the same as its class name
  - A Constructor must have no explicit return type
  - A Java constructor cannot be abstract, static, final, and synchronized
- Types of java constructors
  - Default constructor
  - Parameterized constructor

# Practicles

# 05. Static keyword

# Java static keyword

- The static keyword in Java is used for memory management mainly.
- We can apply static keywords with variables, methods, blocks, and nested classes.
- The static keyword belongs to the class than an instance of the class.
- The static can be:
  - Variable (also known as a class variable)
  - Method (also known as a class method)
  - Block
  - Nested class

# 1) Java static variable

- If you declare any variable as static, it is known as a static variable.
- The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
- The static variable gets memory only once in the class area at the time of class loading.
- If any object changes the value of the static variable, it will retain its value.
- It makes your program memory efficient (i.e., it saves memory).

# 2) Java static Method

- If you apply static keywords with any method, it is known as a static method.
  - A static method belongs to the class rather than the object of a class.
  - A static method can be invoked without the need for creating an instance of a class.
  - A static method can access static data members and can change their value of it.
- Restrictions
  - The static method can not use non-static data members or call a non-static method directly.
  - this and super cannot be used in a static context.

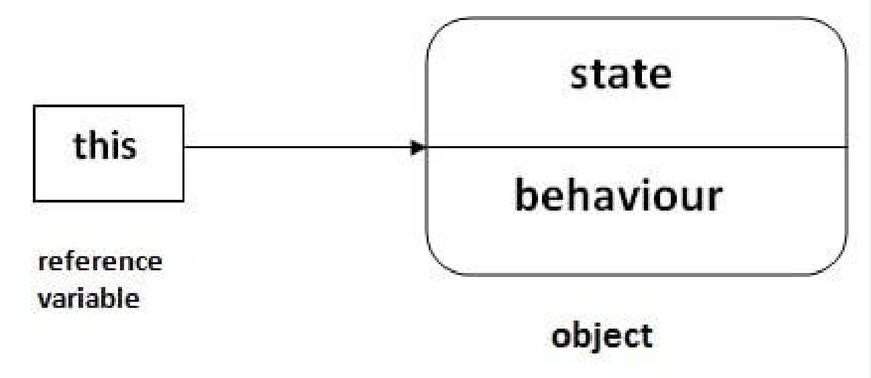
# 3) Java static block

- Is used to initialize the static data member.
- It is executed before the main method at the time of classloading.

# 06. this keyword

# this keyword in Java

- In Java, this is a reference variable that refers to the current object.
- Usage:
  - **this** can be used to refer current class instance variable. (Beginner)
  - **this** can be used to invoke the current class method (implicitly) (Beginner)
  - **this()** can be used to invoke the current class constructor. (Beginner)
  - **this** can be passed as an argument in the method call.
  - **this** can be passed as an argument in the constructor call.
  - **this** can be used to return the current class instance from the method.



# Practicles

# Inheritance

# Inheritance in Java

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object.
- Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

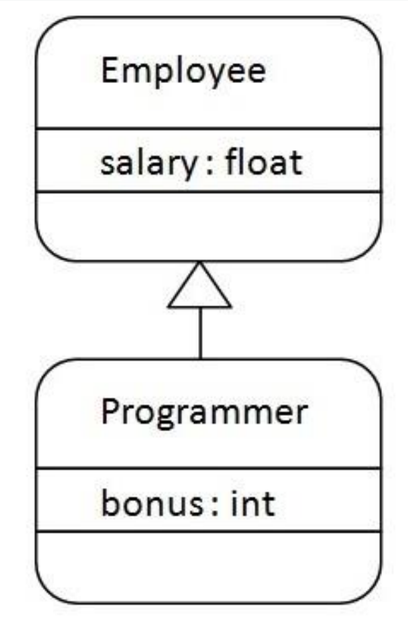
- Why use inheritance in java
  - For Method Overriding (so runtime polymorphism can be achieved).
  - For Code Reusability.

- Terms used in Inheritance
  - **Class**
  - **Sub Class/Child Class**
  - **Super Class/Parent Class**
  - **Reusability**

# Inheritance in Java

- The syntax of Java Inheritance

```
class Subclass-name extends Superclass-name
{
//methods and fields
}
```




```
class Employee{
 float salary=40000;
}

class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary
is:"+p.salary);
   System.out.println("Bonus of Programmer
is:"+p.bonus);
}
}
```

# Practicles