Encapsulation:

<https://www.enjoyalgorithms.com/blog/encapsulation-in-oops>

Abstraction:

<https://www.enjoyalgorithms.com/blog/abstraction-in-oops>

ABSTRACTION and ENCAPSULATION

Yes, abstraction is often achieved with the help of encapsulation in object-oriented programming.

Abstraction is a concept that focuses on hiding complex implementation details while exposing only relevant and necessary information to the outside world. It allows you to represent real-world entities in a simplified manner by defining their essential characteristics and behaviors.

Encapsulation, on the other hand, is the practice of bundling data (attributes) and methods (functions) that operate on that data into a single unit called a class. The idea is to restrict direct access to the internal details of an object and provide controlled access through well-defined interfaces.

Abstraction is supported by encapsulation because encapsulation enables you to control how data and methods are accessed from outside the class. By providing well-defined public interfaces (methods) and controlling access to internal data, you can present an abstract and simplified view of an object's behavior without exposing its internal complexities.

In summary, while abstraction focuses on presenting essential characteristics and behaviors while hiding implementation details, encapsulation enables you to achieve this abstraction by controlling access to the internal structure and behavior of objects. Both concepts work together to create more manageable and maintainable code in object-oriented programming.Top of Form

Inheritance

<https://www.enjoyalgorithms.com/blog/inheritance-in-java>

multiple interitence with interface:

interface AnimalEat {

void eat();

}

interface AnimalTravel {

void travel();

void eat();

}

class Animal implements AnimalEat, AnimalTravel {

public void eat() {

System.out.println("Animal is eating");

}

public void travel() {

System.out.println("Animal is travelling");

}

}

public class Main

{

public static void main(String[] args) {

Animal a = new Animal();

a.eat();

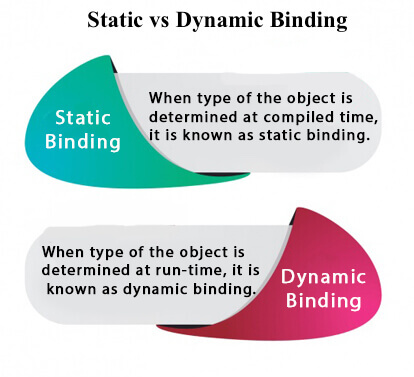
a.travel();

}

}

**Polymorphism:**

[**https://www.enjoyalgorithms.com/blog/difference-between-compile-time-and-runtime-polymorphism**](https://www.enjoyalgorithms.com/blog/difference-between-compile-time-and-runtime-polymorphism)



Dynamic method dispatch is done in runtime polymorphism

Message passing :

<https://www.enjoyalgorithms.com/blog/message-passing-oops>

static : <https://www.javatpoint.com/static-keyword-in-java>

No, you cannot override static methods in Java. Static methods belong to the class itself and are not tied to any specific instance of the class. This means that they are called on the class itself rather than on instances of the class. As a result, static methods cannot be overridden in the same way that instance methods can be.

**THIS** keyword :There can be a lot of usage of **Java this keyword**. In Java, this is a **reference variable** that refers to the current object.