# INHERITENCE IN JAVA

## 1.INHERITENCE

### DEFINITION/WHAT

**Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviour of a parent object.

### NEED/WHY

- ✓ Need 1:Code Reusability
- ✓ Need 2: For Method Overriding (so runtime polymorphism can be achieved).
- ✓ Denotes IS- A relation ship(Inherience)

### IMPLEMENTATION/HOW IT WORKS

Uses extends keyword

### REAL TIME EXAMPLE

A Manager(subclass) is a Employee(super class).

### ADDITIONAL INFORMATION

Java doesn't support multiple inheritance due to Ambiguity.Java supports Multi-level inheritance

## 2.POLYMORPHISM

### DEFINITION/WHAT

Java supports **Static Polymorphism** and **Dynamic Polymorphism**.

Static Polymorphism is providing same method name to different functionalities. The method parameters and return type will differ in the order and datatype.

Dynamic polymorphism is a mechanism in which a call to an overridden method is to resolve at runtime rather than compile-time

### IMPLEMENTATION/HOW IT WORKS

Dynamic Polymorphism can be achieved by IS-A relationship using extends keyword

### REAL TIME EXAMPLE

**Static Polymorphism:**

calculateArea(radius) – to calculate the area of a circle

calcuclateArea(side) to calculate the area of square.

**Dynamic Polymorphism:**

WebsiteOrder extends Order.

StoreOrder extends Order.

You can call the processLogic() method based on the type of the order.

## 3. Abstract Keyword:

### DEFINITION/WHAT

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

### NEED/WHY

*Abstract* classes permit providing a *partial* set of default implementations of methods in a class.

Without abstract classes, we have to provide dummy implementations of the methods, which needs to be overridden.

### IMPLEMENTATION/HOW IT WORKS

#### Abstract class

- It can contain have abstract and non-abstract methods.
- It cannot be instantiated.

#### Abstarct Method

- A method, declared as abstract and does not have implementation, is known as an abstract method.

### REAL TIME EXAMPLE

Fruit is an abstract class and subclasses classes like Mango, Apple should inherit fruit.

displayNutrients() can be declared as abstract method in the parent class.

## 4.Instanceof

### DEFINITION/WHAT

It is used to test whether the object is an instance of the specified type (class or subclass or interface).

### NEED/WHY

When we typecast a parent class reference with Child class, it is always a good idea to check if the typecasting is valid or not.

Reference Link :

https://www.geeksforgeeks.org/instanceof-keyword-in-java/

### IMPLEMENTATION/HOW IT WORKS

An Object held in superclass reference variable can check to which subclass type it belongs to using instanceof Operator

The instanceof in java is also known as type *comparison operator* because it compares the instance with type. It returns either true or false.

### REAL TIME EXAMPLE

A super class Employee reference holds one of the subclass Objects (manager/director/Clerk).

Now to check which subclass object it is pointing to can be done by using instanceof.

## 5. Interface

### DEFINITION/WHAT

An interface is **a contract**, which describes the behaviour an implementing class will have.

### NEED/WHY

- ✓ Need 1:to provide abstraction feature of OOP

### IMPLEMENTATION/HOW IT WORKS

An interface must be implemented in class using implements keyword.

It is a collection of abstract methods. All variables declared in interface as static and final.

It is similar to class but cannot have implementation for any method; otherwise, interface is 100% abstract

### REAL TIME EXAMPLE

CreditCard can be considered as an interface.

HDFCBank, ICCIBank will be the implementation classes.

## Alternatives/Differences between abstract class and Interface

| Abstract class | Interface |
|---|---|
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |

## 6.Wrapper Class

### DEFINITION/WHAT

It is a collection of abstract methods. All variables declared in interface as static and final.

It is similar to class but can't have implementation for any method,otherwise interface is 100% abstract

Since J2SE 5.0, autoboxing and unboxing feature convert primitives into objects and objects into primitives automatically.

### NEED/WHY

- ✓ Need 1:we can't add primitive datatypes to Collection Framework.However wrapper class enables to wrap a primitive datatype using its corresponding wrapper Class and add to Collection

### IMPLEMENTATION/HOW IT WORKS

Eg:int i=10;

This variable can be treated as object as 1)Integer k=new Integer(i); or 2)Integer k=I;

### REAL TIME EXAMPLE

Used in Collections framework

### ADDITIONAL INFORMATION

Wrapper Class(Integer,Float,Byte,Long,Double,Boolean,Short,Character) are defined in java.lang package. We can also use the valueOf() method to convert primitive types into corresponding objects.

```
Eg:int a=10;

//Converts to wrapper Object

Integer obj=Integer.valueOf(a);
```

For unboxing

```
Int k=obj.intValue();
```