**Java OOPs Concepts**

**What is OOPs in java?**

OOps in java is to improve code readability and reusability by defining a Java program efficiently. The main principles of object-oriented programming are **abstraction, encapsulation, inheritance, and polymorphism**. These concepts aim to implement real-world entities in programs.

**List of OOPs Concepts in Java**

* Object
* Class
* Abstraction
* Inheritance
* Polymorphism
* Encapsulation

**1.Object**

Objects are always called instances of a class which are created from a class in java or any other language. They have states and behaviour.

These objects always correspond to things found in the real world, i.e., real entities. So, they are also called run-time entities of the world.

These are self–contained which consists of methods and properties which make data useful. Objects can be both physical and logical data. It contains addresses and takes up some space in memory.

Some examples of objects are a dog, chair, tree etc.

When we treat animals as objects, it has states like colour, name, breed etc., and behaviours such as eating, wagging the tail etc.

Suppose, we have created a class called My book, we specify the class name followed by the object name, and we use the keyword new.

**Object Example 1:**

In the above example, a new object is created, and it returns the value of x which may be the number of books.

Public **class** Mybook {

**int** x=10;

Public **static** **void** main (String args []) {

Mybook Myobj= **new** Mybook ();

System.out.println(MyObj.x);

}

}

**Mybook Myobj= new Mybook ();**

 This is the statement used for creating objects.

**System.out.println(Myobj.x);**

3.Abstraction

This statement is used to return the value of x of an object.

**2.Class**

Collection of objects is called class. It is a logical entity.

### A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

**3.Abstraction**

### Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

### **4.Inheritance**

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

There are five types of inheritance single, multilevel, multiple, hybrid and hierarchical.

* **Single level**

In this one class i.e., the derived class inherits properties from its parental class.  This enables code reusability and also adds new features to the code. Example: class b inherits properties from class a.

Class A is the base or parental class and class b is the derived class.

|  |
| --- |
| 2  3  4  5  6 |

**Syntax:**

Class a {

…

}

Class b extends class a {

…

}

* **Multilevel**

This one class is derived from another class which is also derived from another class i.e., this class has more than one parental class, hence it is called multilevel inheritance.

**Syntax:**

|  |  |
| --- | --- |
| Class a {  ….  }  Class b **extends** **class** a {  ….  }  Class c **extends** **class** b {  …  }  **Hierarchical level**  In this one parental class has two or more derived classes or we can say that two or more child classes have one parental class.  **Syntax:**  Class a {  …  }  Class b **extends** **class** a {  ..  }  Class c **extends** **class** a {  ..  }   * **Hybrid inheritance**   This is the combination of multiple and multilevel inheritances and in java, multiple inheritances are not supported as it leads to ambiguity and this type of inheritance can only be achieved through interfaces.  Consider that class a is the parental or base class of class b and class c and in turn, class b and class c are parental or a base class of class d. Class b and class c are derived classes from class a and class d is derived class from class b and class c.  The following program creates a superclass called add and a subclass called sub, using extend keyword to create a subclass add.   |  | | --- | | // a simple example of inheritance  //create a superclass  Class Add {  **int** my;  **int** by;  **void** setmyby (**int** xy, **int** hy) {  my=xy;  by=hy;  }  }  /create a sub **class**  **class** b **extends** add {  **int** total;  **void** sum () {  **public** Static **void** main (String args [ ] ) {  b subOb= **new** b ( );  subOb. Setmyby (10, 12);  subOb. Sum ( ) ;  System.out.println(“total =” + subOb. Total);  }  } |   It gives output as – **total = 22**  **5.Polymorphism** |

If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Polymorphism in java can be classified into two types:

1. Static / Compile-Time Polymorphism
2. Dynamic / Runtime Polymorphism

**What is Compile-Time Polymorphism in Java?**

Compile-Time polymorphism in java is also known as Static Polymorphism. to resolved at compile-time which is achieved through the Method Overloading.

**What is Runtime Polymorphism in Java?**

Runtime polymorphism in java is also known as Dynamic Binding which is used to call an overridden method that is resolved dynamically at runtime rather than at compile time.

**6.Encapsulation**

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

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

**Advantages of OOPs Concept**

* **Re-usability**

When we say re-usability, it means that “write once, use it multiple times” i.e., reusing some facilities rather than building it again and again, which can be achieved by using class. We can use it n number of times whenever required.

* **Data redundancy**

It is one of the greatest advantages in oops. This is the condition which is created at the data storage when the same piece of data is held at two different places. If we want to use similar functionality in multiple classes, we can just write common class definitions for similar functionalities by inheriting them.

* **Code maintenance**

It is easy to modify or maintain existing code as new objects which can be created with small differences from the existing ones. This helps users from doing rework many times and modifying the existing codes by incorporating new changes to it.

* **Security**

Data hiding and abstraction are used to filter out limited exposure which means we are providing only necessary data to view as we maintain security

* **Design benefits**

The designers will have a long and more extensive design phase, which results in better designs. At a point of time when the program has reached critical limits, it will be easier to program all non-oops separately.

* **Easy troubleshooting**

Using encapsulation objects is self-constrained. So, if developers face any problem easily it can be solved. And there will be no possibility of code duplicity.

* Flexibility
* Problem-solving

**Disadvantages of OOPs Concept**

* Effort – A lot of work is put into creating these programs.
* Speed – These programs are slower compared to other programs.
* Size – OOPs programs are bigger when compared to other programs.