OOPs Concepts Java (Object-Oriented Programming)

Resumming By William Campos

OOP concepts include abstraction, encapsulation, inheritance and polymorphism.

Java defines OOP concepts as follows:

- Abstraction.
  Using simple things to represent complexity. We all know how to turn the TV on, but we don't need to know how it works in order to enjoy it. In Java, abstraction means simple things like objects, classes and variables represent more complex underlying code and data. This is important because it lets you avoid repeating the same work multiple times.

- Encapsulation.
  The practice of keeping fields within a class private, then providing access to those fields via public methods. Encapsulation is a protective barrier that keeps the data and code safe within the class itself. We can then reuse objects like code components or variables without allowing open access to the data system-wide.

- Inheritance.
  A special feature of Object-Oriented Programming in Java, Inheritance lets programmers create new classes that share some of the attributes of existing classes. Using Inheritance lets us build on previous work without reinventing the wheel.

- Polymorphism. Allows programmers to use the same word in Java to mean different things in different contexts. One form of polymorphism is method overloading. That's when the code itself implies different meanings. The other form is method overriding. That's when the values of the supplied variables imply different meanings. Let's delve a little further.

How Abstraction Works

Abstraction lets programmers create useful and reusable tools. For example, a programmer can create several different types of objects, which can be variables, functions or data structures. Programmers can also create different classes of objects as ways to define the objects.For instance, a class of variable might be an address. The class might specify that each address object shall have a name, street, city and zip code. The objects, in this case, might be employee addresses, customer addresses or supplier addresses.

How Encapsulation Works

Encapsulation lets us reuse functionality without jeopardizing security. It's a powerful, time-saving OOP concept in Java. For example, we may create a piece of code that calls specific data from a database. It may be useful to reuse that code with other databases or processes. Encapsulation lets us do that while keeping our original data private. It also lets us alter our original code without breaking it for others who have adopted it in the meantime.

How Inheritance Works

Inheritance is another labor-saving Java OOP concept that works by letting a new class adopt the properties of another. We call the inheriting class a subclass or a child class. The original class is often called the parent. We use the keyword extends to define a new class that inherits properties from an old class.

How Polymorphism Works

Polymorphism in Java works by using a reference to a parent class to affect an object in the child class. We might create a class called “horse” by extending the “animal” class. That class might also implement the “professional racing” class. The “horse” class is “polymorphic,” since it inherits attributes of both the “animal” and “professional racing” class.Two more examples of polymorphism in Java are method overriding and method overloading.In method overriding, the child class can use the OOP polymorphism concept to override a method of its parent class.

That allows a programmer to use one method in different ways depending on whether it’s invoked by an object of the parent class or an object of the child class.

In method overloading, a single method may perform different functions depending on the context in which it’s called.

This means a single method name might work in different ways depending on what arguments are passed to it.

Best Practices for OOP Concepts in Java

The goal of OOP concepts in Java is to save time without sacrificing security and ease of use. The following best practices are all oriented toward advancing that main goal.

- DRY (Don’t Repeat Yourself). A core concept in Java, DRY simply means you should never have two blocks of identical code in two different places. Instead, have one method you use for different applications.If you expect your Java code to change in the future, encapsulate it by making all variables and methods private at the outset. As the code changes, increase access to “protected” as needed, but not too public.

- Single Responsibility. This best practice principle for OOP concepts in Java states that a class should always have only one functionality. That way, the class can be called and/or extended on its own when new uses arise for it, without causing coupling between different functionalities.

- Open Closed Design. Make all methods and classes Closed for modification but Open for an extension. That way, tried and tested code can remain static but can be modified to perform new tasks as needed.

For more:

OOPs Concepts Java: https://stackify.com/oops-concepts-in-java/#:~:text=The%20main%20ideas%20behind%20Java%27s,of%20them%20without%20compromising%20security.

OOPs Concepts Java 2: https://www.javatpoint.com/java-oops-concepts

Java inheritance: https://www.tutorialspoint.com/java/java_inheritance.htm

Java Polymorphism: http://www.sitesbay.com/java/java-polymorphism

10 Object-Oriented (OOP) Design Principles Java Programmers Should Know: https://javarevisited.blogspot.com/2018/07/10-object-oriented-design-principles.html#ixzz7hoHs09Sz: