Java OOP

Rev 1.2 (21/07/2019

There are four main OOP concepts in Java: *abstraction*, *encapsulation*, *inheritance* and *polymorphism*.

Abstraction - using simple things to represent complexity, Java uses objects, classes and attributes to represent more complex data and code.
Abstract classes and interfaces are provided as an access point for more complex implementations.

Encapsulation - hiding implementation details by keeping data and code references within the class. Can be acomplished with java acess modifers for method and attributes (private, public) and setter and getter methods.
Encapsulation enables functionality reuse without jeopoarding security or backward incompatiblity.

Inheritance - allows a class (subclass) to adopt the properties of a another (super class). One of the simplest way to reuse code in Java close related with Polymorphism.

Polymorphism - concept where the same word can have different meanings in different contexts. A polymorphic class inherits attributes from more than one class, all Java classes are polymorphic because they inherit from class Object and their own type.

Polymorphis can be achieved with:

- method **overriding**, the subclasses overides a method of the parent class.
- method **overloading**, a single method may perform functions depending on the context in which it’s called.

Best practices for OOP

DRY (don’t repeat yourself) - avoid code replication (e.g. support refactorization).

Single responsability - a class should always have one only functionality.

Open closed design - make all methods closed for modification but open for extension.

(extra):

> Differentiate between data object classes that can be reused and functionality classes that perform specific operations.
>
> Avoid overcomplexity, too many layers of indirection levels when a functionalities relies on multiple desing patters or a cascade of functions.
>
> Avoid code fragmentation, use of very small classes, which functionaly could be easily grouped in just one class.
>
> Keep auxiliar or utility classes separted from busines logic.