
Applied Java

Daniel Lewandowski, Andrzej Kałuża

Object Oriented Features

Key features

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Abstraction

Abstraction is the modelling process

- hide implementation details
- show only essential information to the user
- abstract class is a restricted class that cannot be used to create objects
- abstract method: can only be used in an abstract class
- interface: only methods prototypes (also default implementations)



Encapsulation

Technique of hiding data from users

- object data (fields) should not be directly visible
- object data should be protected from direct modification
- object should expose methods allowing modification
- scope modification
 - private - method and field is visible only by an object
 - protected - method and field is visible also by child object
 - public - method and field is visible by all



Inheritance

In the Java language, classes can be derived from other classes, thereby inheriting fields and methods from those classes

- A class that is derived from another class is called a subclass (also a derived class, extended class, or child class)
- The class from which the subclass is derived is called a superclass (also a base class or a parent class).



Polymorphism

Biology:

- an organism or species can have many different forms or stages

Java

- subclasses of a class can define their own unique behaviors
- and yet share some of the same functionality of the parent class.

An exercise





SOLID principles

SOLID principles were formulated in a response to arising problems with maintaining and developing complex programs

- Single Responsibility
- Open/Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion

Single Responsibility

- class should have a single responsibility
- class should have only one reason to change.
- natural tendency to mix responsibilities (aka features)
- finding and separating those responsibilities is a challenging

Open for Extension, Closed for Modification

- class open for extension (i.e. through inheritance)
- closed for modification
- new functionality without changing the existing code

Liskov Substitution

- substitutability states that if S is a subtype of T, then objects of type T may be replaced with objects of type S
- subclass must not break superclass contract
- *equals* and *hashCode*
-

Interface Segregation

- client should not be forced to depend on methods it does not use
- larger interfaces should be split into smaller ones
-

Dependency Inversion

- decoupling of software modules
- must not depend on implementation
- must be based on abstractions

An exercise



References

- <https://en.wikipedia.org/>
- <https://docs.oracle.com/en/java/>
-