Delegation Principle

Overview

- delegation is the concept of one class “delegating” its behavior to another class
  - don't do all stuff by yourself, delegate it to a respective class
  - when you delegate, you are simply calling up some class which knows what must be done
    - you do not really care how it does it, all you care about is that the class you are calling knows what it needs to do

- delegation can be viewed as a relationship between objects where one object forwards certain method calls to another object, called its delegate

- delegation is an extreme example of object composition
  - shows that you can always replace inheritance with object composition as a mechanism for code reuse
  - delegation means that you use an object of another class as an instance variable, and forward messages to the instance

- it is better than inheritance for many cases
  - it makes you to think about each message you forward
    - the instance is of a known class, rather than a new class
  - it does not force you to accept all the methods of the super class
    - you can provide only the methods that really make sense

Advantages

- the primary advantage of delegation is run-time flexibility
  - makes it easy to compose behaviors at run-time and to change the way they are composed

- delegation is a good design choice only when it simplifies more than it complicates
  - how effective it will be depends on the context and on how much experience you have with it
  - delegation works best when it is used in design patterns

- several design patterns use delegation

  - **State** - an object delegates requests to a State object that represents its current state

  - **Strategy** - an object delegates a specific request to an object that represents a strategy for carrying out the request

  - **Visitor** - the operation that gets performed on each element of an object structure is always delegated to the Visitor object

Examples

- assume your class is called B and the delegated class is called A
  - use delegation if you want to enhance A, but A is final

- the equals() and hashCode() method in Java is a classic example of delegation
  - in order to compare two objects for equality, we ask the class itself to do comparison instead of client class doing that check

- event delegation is another example of delegation
  - an event is delegated to handlers for handling