The Single Responsibility Principle

Overview

- states that every class should have responsibility over a single part of the functionality provided by the software
  - the responsibility should be entirely encapsulated by the class
  - all its methods should be narrowly aligned with that responsibility
  - a class should have only one job
- "a class should have a single responsibility, where a responsibility is nothing but a reason to change"
- should make sure that one class at the most is responsible for doing one task or functionality among the whole set of responsibilities that it has
  - only when there is a change needed in that specific task or functionality should this class be changed
- the Single Responsibility Principle is closely related to the concepts of coupling and cohesion
- coupling is the degree of interdependence between software classes or methods
  - a measure of how closely connected two classes or two methods are
  - the strength of the relationships between classes

Overview (cont'd)

- low coupling means small dependencies between classes/methods
  - easier to change code without introducing bugs in other classes or other methods
- tight coupling means two classes/methods are closely connected
  - a change in one module may affect another module
- cohesion refers to what the class (or method) can do
- low cohesion would mean that the class does a great variety of actions
  - it is broad, unfocused on what it should do
- high cohesion means that the class is focused on what it should be doing
  - contains only methods relating to the intention of the class
- the single responsibility principle is about limiting the impact of change by designing loosely (low) coupled classes that are highly cohesive

Examples of responsibilities

- some examples of responsibilities to consider that may need to be separated include:

- Persistence
- Validation
- Notification
- Error Handling
- Logging
- Class Selection / Instantiation
- Formatting
- Parsing
- Mapping