Liskov Substitution Principle

Overview

- the Liskov Substitution principle was introduced by Barbara Liskov

- the principle defines that objects of a superclass can be replaceable with objects of its subclasses without breaking the application
  - requires the objects of your subclasses to behave in the same way as the objects of your superclass
  - methods which use a superclass type must be able to work with the subclass without any issues

- an overridden method of a subclass needs to accept the same input parameter values as the method of the superclass
  - do not implement any stricter validation rules on input parameters than implemented by the parent class
    - any code that calls this method on an object of the superclass might cause an exception, if it gets called with an object of the subclass

- the return value of a method of the subclass needs to comply with the same rules as the return value of the method of the superclass
  - you can only decide to apply stricter rules by returning a specific subclass of the defined return value or by returning a subset of the valid return values of the superclass

- in order to follow LSP the subclass must enhance functionality, but not reduce functionality

Overview (cont'd)

- this principle is in line with what Java also allows
  - a superclass reference can hold a subclass object
    - superclass can be replaced by subclass in a superclass reference at any time
  - the Java inheritance mechanism follows the Liskov Substitution Principle

- LSP is closely related to the Single responsibility principle and Interface Segregation Principle
  - If a base class has more functionality than a subclass might not support some of the functionality and does violate LSP

- this principle extends the Open/Closed principle
  - the Open/closed principle says that a class should be open for extension and closed for modification
    - we override the original class and implement the functionality to be changed in the overriding class
    - when the subclass object is used in place of the super-class the overridden functionality is executed
    - this is exactly in line with the Liskov Substitution Principle

Another Example (Circle-Ellipse)

- all circles are inherently ellipses with their major and minor axes being equal

- in terms of classes, Ellipse could be a base class and Circle a subclass
  - an object of type Circle can be assigned to a reference of type Ellipse
  - all the methods in Ellipse can be invoked on a Circle object

- one inherent functionality of an ellipse is that its stretchable
  - the length of the two axes of an ellipse can be changed

- so, we will add methods setLengthOfAxisX() and setLengthOfAxisY() to the Ellipse class

- calling any of these two methods on an object of type circle inside a reference of type ellipse would lead to a circle no longer being a circle
  - for a circle the length of the major and minor axes are equal

- this is a violation of the Liskov Substitution Principle
  - assigning a subtype object to a super type reference does not work