Object Oriented Development

# Abstract Classes, Interfaces and Constructors - Exercise

# What does this exercise cover?

# This exercise will give you a chance to get used to defining and using Abstract Classes, Interfaces and Constructors within Java.

# How long will the exercise take to complete?

# 1-2 hours

# What should you have already completed?

UML Training, Four Pillars of Object Oriented Development Training

# What do you need?

In order to complete this tutorial exercise you will need:

* Java Development Kit 1.6 or above
* Apache Maven
* Eclipse IDE Kepler or above
* Subversion

# Instructions

Create a new project, setting it up as you have been shown.   
  
Your task is to produce a series of simple code samples that demonstrates each the concepts outlined below. You should use this exercise as a chance to experiment with what the Java language can do.

Remember: Each class, abstract class and interface should be its own file. Create sub-packages to keep your code organised.

**CHALLENGE:** One of the concepts below is illegal within Java. Which one is it? Once you find it, satisfy yourself that the remaining concepts are indeed possible.

|  |  |
| --- | --- |
|  | Concept Theory |
| #1 | An abstract class can contain both abstract and non-abstract methods. |
| #2 | An abstract class can have member variables. |
| #3 | An abstract class can implement an interface. |
| #4 | An abstract class can have a constructor. |
| #5 | All variables in an interface are implicitly final, even if the final modifier is not used. |
| #6 | An interface can *extend* another interface. |
| #7 | Any class that implements an interface must provide method bodies for each method in the interface, unless it is an abstract class. |
| #8 | Assume interface B extends interface A. Demonstrate that a class that implements B must provide method bodies for methods in both A *and* B. |
| #9 | Every class must have a constructor, even if it is the default constructor that the compiler adds. Demonstrate making an instance of a class by using the compiler-generated no-arg constructor. |
| #10 | Constructors can be overloaded. |
| #11 | Methods can be overloaded. |
| #12 | Methods can be inherited. |
| #13 | Member variables can be overridden. |
| #14 | A sub-class constructor can call its super-class constructor using the *super* keyword. |
| #15 | Any concrete class with an abstract class in its hierarchy must implement all abstract methods from the superclass(es). |