B – Starting Classes

Student Exercises – Documentation

# Exercises – Overview

* **AnsweringMachine** - Create the AnsweringMachine class so that it can provide a default answer for an incoming phone call as well as a customizable answer.
* **Account** - Extend the Account class from the example to include more information. Specifically, include an ***AccountType:String***, ***BankName:String***, ***BranchNumber:Integer***, and ***InstitutionNumber:Integer***.
* **CanadianAddress** - Create the CanadianAddress class so that it can represent the majority of possible addresses that some place may have in Canada.
* **Course** - Create the Course class so that it represents a post-secondary course.
* **ExamResult** - Create the ExamResult class so that it represents the results of an exam written by a student.
* **LabResult** - Create the labResult class so that it represents the results of a lab submitted by a student.

### AnsweringMachine

The AnsweringMachine class provides a method to give an answer to an incoming phone call.

## Problem Statement:

Create the AnsweringMachine class so that it can provide a default answer for an incoming phone call as well as a customizable answer. The methods should be named answer and they should return a String. There should be two methods in total, and both of them should be declared as *instance* members (*non-*static). The actual content of the default message can be whatever you choose, while the customizable method will receive a single String argument that has the name of the person receiving the message.

Also create a driver that demonstrates the AnsweringMachine in a Console environment

Create a driver for the AnsweringMachine class that creates an instance of the class and displays the results of calling the answer methods.

### Account

Extend the Account class from the example to include more information.

## Problem Statement:

Extend the Account class from the example to include more information. Specifically, include an **AccountType:String**, **BankName:String**, **BranchNumber:Integer**, and **InstitutionNumber:Integer**.

Also modify the driver to make use of the added information.

#### Notes

The branch number and the institution number together make up the Transit Number for a bank. "The bank transit number is 8 digits long. This is divided into a 5 digit branch number and 3 digit institution code, for example 10000-200." (See <http://en.wikipedia.org/wikiSort_code>)

For more information on bank accounts and transit numbers in Canada, see <http://en.wikipedia.org/wiki/Routing_transit_number#Canadian_transit_number>.

### CanadianAddress

This class represents an address for some place in Canada.

## Problem Statement:

Create the CanadianAddress class so that it can represent the majority of possible addresses that some place may have in Canada. Design the class to have only public fields, as specified in this document.

* Data Attributes of the CanadianAddress class:
  + Street : String
  + Unit : String
  + City : String
  + Province : String
  + PostalCode : String
  + RuralRoute : String
  + BoxNumber : String

Also create a driver for testing this class; you may use any name for the driver as long as it is not already mentioned in this package. In the driver, create instances of the CanadianAddress class that represent your current address as well as the address of your school (use hard-coded data).

### Course

This class represents a post-secondary course with a theory (exam) and a lab portion.

## Problem Statement:

Create the Course class so that it represents a post-secondary course. Design the class to have only public fields, as specified in this document.

* Data Attributes of the Course class:
  + CourseName : String
  + CourseNumber : String
  + ExamCount : Integer
  + LabCount : Integer
  + ClassHours : Integer

Also create a driver for testing this class; you may use any name for your driver as long as it is not already mentioned in this package. In the driver, instantiate all of the first term classes you are taking and populate those objects with data (use hard-coded data).

### ExamResult

This class represents the results of an exam for a student.

## Problem Statement:

Create the ExamResult class so that it represents the results of an exam written by a student. Design the class to have only public fields, as specified in this document.

* Data Attributes of the ExamResult class:
  + Name : String
  + TotalMarks : Integer
  + MarksEarned : Real
  + ExamWeight : Integer
  + StudentId : Integer

Also create a driver for testing this class; you may use any name for the driver as long as it is not already mentioned in this package. In the driver, instantiate all of the exams you have taken to date in this course and populate those objects with data (use hard-coded data).

### LabResult

This class represents the results of an lab for a student.

## Problem Statement:

Create the labResult class so that it represents the results of a lab submitted by a student. Design the class to have only public fields, as specified in this document.

* Data Attributes of the LabResult class:
  + LabNumber : Integer
  + TotalMarks : Integer
  + MarksEarned : Real
  + LabWeight : Integer
  + StudentId : Integer

Also create a driver for testing this class; you may use any name for the driver as long as it is not already mentioned in this package. In the driver, instantiate all of the labs you have submitted to date in this course and populate those objects with data (use hard-coded data).