G – Raising Exceptions

Student Exercises - Documentation

# Exercises

The following exercises are given as practice for this topic. The recommended order of the exercises is as follows.

1. **Account** – Add one more point of validation to the Account class: Do not allow depositing or withdrawing non-positive amounts.
2. **Rectangle** – The height and width of the rectangle must be greater than zero.
3. **Cone** – The radius and height of the cone must be greater than zero.
4. **Course** – The course name and number cannot be blank, and the exam and lab counts must be greater than zero. The class hours must also be greater than zero.
5. **Cylinder** – The radius and height of the cylinder must be greater than zero.
6. **HazardousMaterial** – The class code for the hazardous material can only be the letters ‘A’ through ‘F’, inclusive.
7. **ExamResult** - Requires a positive, non-zero value for the total marks and weight. The weight cannot be over 50. The marks earned can be between zero and the total marks, inclusive. The student ID must be 9 digits, and the exam name cannot be an empty string.
8. **LabResult** – Requires a positive, non-zero value for the lab number, total marks and weight. The weight cannot be over 50. The marks earned can be between zero and the total marks, inclusive. The student ID must be 9 digits.
9. **PeopleCounter** – Does not allow adding a negative number of people to the counter.
10. **BulkItem** – The description cannot be blank and the cost and quantity values must be greater than zero.

# Account

## Problem Statement

Write the code that will add another piece of validation to the Account class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should get the bank name, branch number, institution number, account number, balance, overdraft limit, and account type and allow the overdraft limit to be set
* Should support deposits
* Should only support withdrawals if the amount does not exceed the sum of the balance and the overdraft limit, otherwise an exception stating “Insufficient Funds” should occur
* Should identify if the account is overdrawn
* Should require bank name and account type (that is, they cannot be empty or null)
* Should trim the bank name and account type
* Should verify that the branch number is six digits and the institution number is three digits
* Should require an opening balance
* Should not allow a negative overdraft limit
* ***Should only allow positive, non-zero amounts when performing a deposit or withdrawal***

Use the following class diagram when creating your solution.



# Rectangle

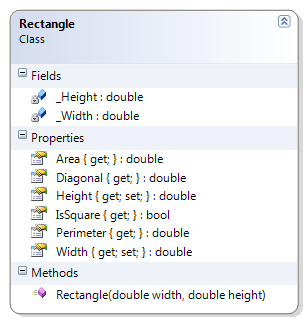
The height and width of the rectangle must be greater than zero.

## Problem Statement

Write the code to provide validation for the Rectangle class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should get and set the height and the width
* Should calculate the area, the perimeter and the diagonal
  + The formula for the diagonal is
* Should determine if the rectangle is a square
* ***Should require the height and width to be greater than zero***

Use the following class diagram when creating your solution.



# Cone

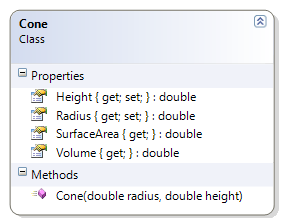
The radius and height of the cone must be greater than zero.

## Problem Statement

Write the code to provide validation for the Cone class that meets the following requirements (new requirements are in ***green, bold italic*** font):

* Should get the radius and the height
* Should calculate the volume and the surface area
* ***Should require the radius and height to be greater than zero***

Use the following class diagram when creating your solution.



# Course

The course name and number cannot be blank, and the exam and lab counts must be greater than zero. The class hours must also be greater than zero.

## Problem Statement

Write the code to provide validation for the Course class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should get the course name and number, the number of exams and labs, and the class hours for the course.
* ***Should not allow empty course names or numbers***
* ***Should trim spaces from the course name and number***
* ***Should require class hours as well as lab and exam counts to be greater than zero***



# Cylinder

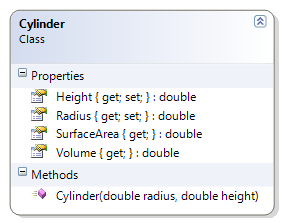
The radius and height of the cylinder must be greater than zero.

## Problem Statement

Write the code to provide validation for the Cylinder class that meets the following requirements (new requirements are in ***green, bold italic*** font):

* Should get the radius and the height
* Should calculate the volume and the surface area
* ***Should make sure the radius and height are greater than zero***

Use the following class diagram when creating your solution.



# HazardousMaterial

The class code for the hazardous material can only be the letters ‘A’ through ‘F’, inclusive.

## Problem Statement

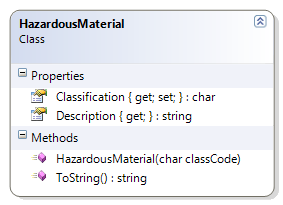
Write the code to provide validation for the HazardousMaterial class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should return the class code as the classification
* ***Should make sure only class codes ‘A’ through ‘F’ are allowed (in either upper or lower case)***
* ***Should make sure the classification is stored in upper case***
* Should get the description for the class, based on the following table

|  |  |
| --- | --- |
| **Class Code** | **Description** |
| A | Compressed Gas |
| B | Flammable and Combustible Material |
| C | Oxidizing Material |
| D | Poisonous and Infectious Material |
| E | Corrosive Material |
| F | Dangerously Reactive Material |

* Should override the toString() method to get the full description and class code in the following format:
  + “Class ClassCode - Description”

Use the following class diagram when creating your solution.



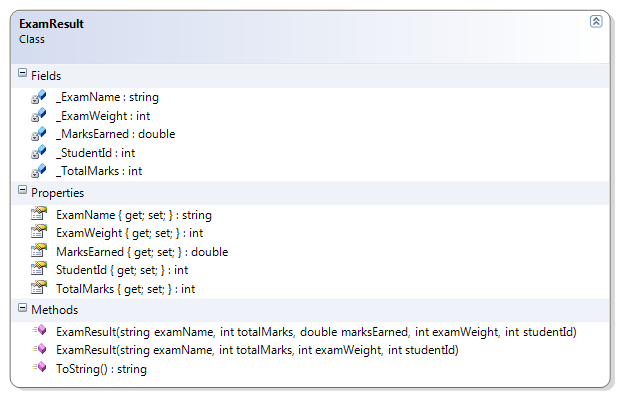
# ExamResult

Requires a positive, non-zero value for the total marks and weight. The weight cannot be over 50. The marks earned can be between zero and the total marks, inclusive. The student ID must be 9 digits, and the exam name cannot be an empty string.

## Problem Statement

Write the code to provide validation for the ExamResult class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should get the name, student Id, total marks, and exam weight
* Should get and set the marks earned
* Should override the toString() to show  
  “The student (*studentId*) received *earnedMarks*/*totalMarks* for this *examName* exam.”
* ***Should require the total marks and weight to be a positive, non-zero value***
* ***Should not allow a weight over 50***
* ***The marks earned must be between zero and the total possible marks, inclusive***
* ***The student ID must be nine digits***
* ***The exam name cannot be an empty string (and must be trimmed)***



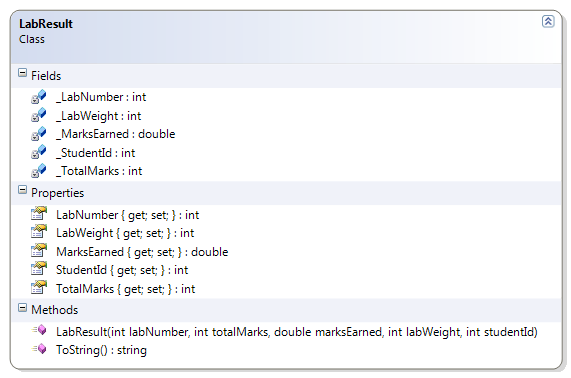
# LabResult

Requires a positive, non-zero value for the lab number, total marks and weight. The weight cannot be over 50. The marks earned can be between zero and the total marks, inclusive. The student ID must be 9 digits.

## Problem Statement

Write the code to provide validation for the LabResult class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should get the lab number, student Id, total marks, and lab weight
* Should get and set the marks earned
* Should override the toString() to show  
  “The student (*studentId*) received *earnedMarks*/*totalMarks* for this lab.”
* ***Should require the total marks and weight to be a positive, non-zero value***
* ***Should not allow a weight over 50***
* ***The marks earned must be between zero and the total possible marks, inclusive***
* ***The student ID must be nine digits***
* ***The lab number must be a positive number***



# PeopleCounter

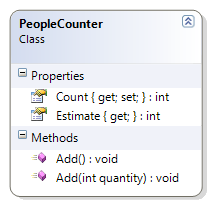
Does not allow adding a negative number of people to the counter.

## Problem Statement

Write the code needed to track people entering and leaving a store. It must be able to estimate the number of people that have entered the store at the end of the day. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should default to having a count of zero people when first started up
* Should increment by one (that is, count a single person)
* Should increment by a specified quantity (for when a group of people enter)
* Should adequately estimate the number of people who entered the store, assuming that each person who enters also leaves the store
  + Each estimate should be rounded down; for example, if 15 people are counted, then the estimate should be for only 7 (not 7.5 or 8)
* ***Should not allow a zero or negative quantity when adding to the counter***

Use the following class diagram when creating your solution.



# BulkItem

The description cannot be blank and the cost and quantity values must be greater than zero.

## Problem Statement

Write the code for the BulkItem class. The solution must meet the following requirements (new requirements are in ***green, bold italic*** font):

* Should get and set the cost and quantity of the bulk item
* ***Should get and set the description***
* ***Should ensure the description is not empty (and that it is trimmed)***
* ***Should ensure the cost and quantity values are greater than zero***
* Should calculate the cost for each item in the bulk item
* Should properly round values for currency (either the US or Canadian dollar)

Use the following class diagram when creating your solution.

