K – Looping Structures and Collections

Student Exercises

# Exercises

1. **PhoneBook** – This extends the PhoneBook class by ensuring that duplicate phone numbers are not added to the collection.
2. **Registrar** – The Registrar class is responsible to support the enrollment of students. This class maintains the student body as a collection of Student objects. It supports the ability to find and remove students, switch students to another program, and get the number of students enrolled in a specific program.
3. **BookBag** – The BookBag class represents a simple “shopping cart” for a book consignment store. Books are sold on consignment, and customers can add or remove books from their BookBag as well as search their BookBag for books by ISBN. Customers can also see the total price of the items in their BookBag.
4. **DeckOfCards** –The DeckOfCards class represents a complete deck of cards. When the deck is first created, a card is created for each suit. The DeckOfCards supports two public methods: Draw() and Shuffle().

# PhoneBook

## Problem Statement

This extends the PhoneBook class by ensuring that duplicate phone numbers are not added to the collection. Make the following additions and modifications to complete the solution.

* **AddPhoneNumber()** – This method must be modified to ensure that the telephone number does not already exist; that is, no duplicate phone numbers are allowed, and an exception must be thrown if the supplied PhoneNumber already exists.

## Supporting Classes

# Registrar

## Problem Statement

The Registrar class is responsible to support the enrolment of students. This class maintains the student body as a collection of Student objects. It supports the ability to find and remove students, switch students to another program, and get the number of students enrolled in a specific program. Code the following methods to complete the Registrar class.

* **Add()** – This method takes the supplied Person information as well as the program of study to create a new Student, adding them to the collection of students in the student body. It returns the student Id for the new enrolment.
* **FindStudent()** – This method searches the student body for a student with a matching Id.
* **RemoveStudent()** – This method searches the student body for a student with a matching Id and removes them from the collection, if found.
* **GetProgramEnrollment()** – This method searches the collection of students to find out how many are enrolled in a particular program. Validate the Program name before performing the search.

## Supporting Classes

# BookBag

## Problem Statement

The BookBag class represents a simple “shopping cart” for a book consignment store. Books are sold on consignment, and customers can add or remove books from their BookBag as well as search their BookBag for books by ISBN. Customers can also see the total price of the items in their BookBag. Code the following methods to complete the BookBag class.

* **GetTotal()** – Loop through the collection of books and total the price of all the books. Ensure that the amount is rounded to two decimal places (for dollars and cents).
* **FindBook()** – Look through the collection of books to find a book with the specified ISBN number. Throw an exception if the supplied ISBN is null.
* **RemoveBook()** – Remove the book with a matching ISBN from the collection of book consignments.

## Supporting Classes

# DeckOfCards

## Problem Statement

Modify the DeckOfCards class to support shuffling of the deck.

* **Shuffle()** – This method resets the deck to a full deck of cards and then “shuffles” the deck until the cards are randomly distributed through the collection.

## Supporting Classes