\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC253 C# ProGRAMMING**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 01 **CREATING C# CONSOLE APPLICATIONS**

# Objectives

In this lab assignment, students will learn:

* How to create C# console applications
* How to declare a class and use it to create an object
* How to create instance variables, properties and methods
* How to call an object's methods to make them perform their tasks

# Goals

In this lab assignment, students will demonstrate the abilities to:

* Create C# console applications
* Declare a class and use it to create an object.
* Create instance variables, properties and methods.
* Call an object's methods to make them perform their tasks.

# Description

Create a C# console application for each question. When you create a new C# project, Visual Studio creates a folder to hold every file and sub-folder for your project. You need to zip this folder and submit the zip file to Blackboard.

1. Create an Invoice class for a hardware store to create invoices for customer orders.
   1. This class has two instance variables for quantity (integer type) and unit price (decimal type). Provide properties with get and set accessors for both of them. If the value passed to the set accessor is negative, the instance variable should be left unchanged.
   2. Create an auto-implemented property for part code (string type).
   3. Create a constructor with two parameters, which are used to initialize part code and unit price.
   4. Create a DisplayOrder method to displays part code, unit price and quantity. This method also calculates and displays total price.
   5. Write an InvoiceTest class to test the Invoice class. Ask the user to choose one of three parts: A2507, A2714, or A2845. Their unit prices are $16.25, $24.75 and $27.00, respectively. Create an Invoice object. Ask the user to enter quantity. Call the DisplayOrder method to display the order.

Example:

Enter 1 for Part A2507, 2 for Part A2714, 3 for Part A2845: 2

Enter quantity: 4

Your Order:

Part code: A2714

Quantity: 4

Unit price: $24.75

Total price: $99.00

1. Create a Drone class to simulate flying a drone.
   1. Create two instance variables to store the speed (integer type) and height (integer type) of the drone. Provide properties with get and set accessors for both of them. If the value passed to the set accessor is negative, set the instance variable to 0.
   2. Create an auto-implemented property for drone ID (string type).
   3. Create a constructor with one parameter, which is used to initialize drone ID.
   4. Create an Accelerate method to increase the speed by 10.
   5. Create a Decelerate method to decrease the speed by 10.
   6. Create an Ascend method to increase the height by 10.
   7. Create a Descend method to decrease the height by 10.
   8. Write a DroneTest class to test the Drone class. Create a Drone object with ID D1234. Perform the following sequence of operations: ascend, descend, descend, decelerate, accelerate and decelerate. Display drone ID, spend and height after each operation.

Expected output:

ID:D1234 Speed:0 Height:10

ID:D1234 Speed:0 Height:0

ID:D1234 Speed:0 Height:0

ID:D1234 Speed:0 Height:0

ID:D1234 Speed:10 Height:0

ID:D1234 Speed:0 Height:0

# Grading rubric

Program 1:

Instance variables and auto-implemented property [10 pts]

Properties with get and set accessors [10 pts]

Constructor [10 pts]

DisplayOrder method [10 pts]

Create object and use object [10 pts]

Program 2:

Instance variables and auto-implemented property [8 pts]

Properties with get and set accessor [8 pts]

Constructor [8 pts]

Methods for changing speed and height [18 pts]

Create object and use object [8 pts]