C# Refresher Exercises

**Topic: C# coding**

Lecturer: John Walsh

**Exercise 1**

Examine the following program and answer the questions that follow:

using System;

class Schedule

{

public static void Main()

{

const int Dress = 45;

const int Eat = 30;

const int Drive = 30;

const int Class = 60;

int totalmins = 0;

totalmins = Dress + 3 \* Eat + 2 \* Drive + 4 \* Class;

Console.Write("You spend " + totalmins / 60);

Console.Write(" hours and " + totalmins % 60 );

Console.WriteLine(" minutes a day on scheduled activities");

}

}

1. What is written by program **Schedule** (What does it do)?
2. List the identifiers that are defined in program Schedule.
3. Which of these identifiers are named constants.
4. What is the purpose of the % in the program.

**Exercise 2**

Design and write a program that converts 60 miles to kilometres. One mile equals 1.60935 kilometres

**Exercise 3**

A user has to change their password to a system since it has expired. Design & develop a program that allows a user to type in a new password and a confirmation of the new password. Print an error message if the confirmed password if different from the first one entered. Print a message to indicate success or otherwise. (Password1)

**Exercise 4**

Write a program that asks the user for two numbers and then displays the sum of these two numbers. It then gives the user the opportunity to have another go or quit, by asking then to enter “Yes” to continue, otherwise quit.

**Exercise 5**

Design and develop a program that prints the following pyramid shape of asterisks

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

**Exercise 6** ( OO and Classes)

1. Implement a *BankAccount* class to represent a generic bank account. Each account has an account number, balance, and methods to make deposits and withdrawals from the account. More specifically the class should contain the following items:

Fields:

The bank account number (alpha-numeric)

The balance on the account

Properties:

A read-only property for the account number field

A read-write property for the balance field

Constructors:

One default constructor which sets the account number to an empty string (“”) and sets the balance to zero

One constructor which sets the account number to the value of an input parameter, and sets the balance to zero

Methods:

*MakeDeposit* -allow an amount of money to be deposited in the account

*MakeWithdrawal* - allow an amount of money to be withdrawn from the account

1. Implement a *SavingsAccount* class that will inherit/extend the *BankAccount class.* A *SavingsAccount* will be exactly the same as the *BankAccount* except that when you make a deposit, the bank will add an extra 10 Euro to your account balance

Specifically your *SavingsAccount class* should contain :

Methods:

*MakeDeposit* –overrides the *BankAccount*  *MakeDeposit*() method to add 10 euro to the balance when a person deposits money

**Hint** :

Remember to update the ***BankAccount*** *MakeDeposit*() methof to include the virtual keyword e.g

public virtual void MakeDeposit(double amount)

{

…..

}

Rremember to include the override keyword in the ***SavingsAccount*** *MakeDeposit*() method e.g

**public virtual void MakeDeposit(double amount)**

{

…..

}