# CITP 155 – Programming I

## Classes Worksheet

## Graded Activity (30 points)

After reading the C# tutorial articles covering Classes, respond to the following. Leave the questions in a black font. Enter your answers in a blue font.

The articles are found at the following links:

<https://csharp.net-tutorials.com/classes/introduction/>   
<https://csharp.net-tutorials.com/classes/properties/>   
<https://csharp.net-tutorials.com/classes/constructors-and-destructors/>   
<https://csharp.net-tutorials.com/classes/method-overloading/>   
<https://csharp.net-tutorials.com/classes/visibility/>   
<https://csharp.net-tutorials.com/classes/static-members/>   
<https://csharp.net-tutorials.com/classes/inheritance/>   
<https://csharp.net-tutorials.com/classes/namespaces/>

1. What is a class?

A class is a group of related methods and variables.

1. What do class properties allow you to control?

They allow you to control how fields can be accessed and manipulated.

1. A property is like a combination of what two things?

A field and a method.

1. What are the two methods that can be part of a property and what does each one do?

The get method and the set method. The Get method will return a value, while the set method will assign a value that is passed to it.

1. What is a constructor?

Constructors are special methods, used when instantiating a class.

1. What is a destructor?

A destructor is a method called once an object is disposed, and can be used to cleanup resources used by the object.

1. What is method overloading?

A programming technique that allows the programmer to define several methods with the same name, which take a different set of parameters in order to account for optional parameters.

1. In the following code example, highlight (in yellow) the version of the Plus method which would be called with this line of code:   
   returnedValue = SillyMath.Plus(firstNumber, secondNumber, thirdNumber);

class SillyMath

{

public static int Plus(int number1, int number2)

{

return Plus(number1, number2, 0);

}

public static int Plus(int number1, int number2, int number3)

{

return Plus(number1, number2, number3, 0);

}

public static int Plus(int number1, int number2, int number3, int number4)

{

return number1 + number2 + number3 + number4;

}

}

1. When a class, method, variable or property is **public**, from where can it be reached?

It can be reached from anywhere in the outside, without notifying the declaring class.

1. When a class, method, variable or property is **protected**, from where can it be reached?

It can only be reached from within the same class, or from a class which inherits from this class.

1. When a class, method, variable or property is **private**, from where can it be reached?

It can only be reached by members from the same class. This is the most restrictive visibility.

1. What is the usual way to communicate with a class?

Through DTOs

1. True or False: A static class be instantiated.

False

1. True or False: A static class cannot contain non-static members.

True

**Consider the following code sample for questions 15-20.**  
public class Rectangle

{

private int width, height;

public Rectangle(int width, int height)

{

this.width = width;

this.height = height;

}

public void OutputArea()

{

Console.WriteLine("Area output: " + Rectangle.CalculateArea(this.width, this.height));

}

public static int CalculateArea(int width, int height)

{

return width \* height;

}  
}

1. What is the name of the class?

Rectangle.

1. What is the visibility of the class?

Public.

1. What are the names of the methods in this class?

Rectangle, OutputArea, CalculateArea.

1. Which method is the constructor method for this class?

Rectangle.

1. What data type does each of the methods return. List the method name and the type of data it returns.

Rectangle: int

OutputArea: null

CalculateArea: int

1. If the class is instantiated and used with the following lines of code, what will be displayed to the console?

int width = 5;  
int height = 10;  
Rectangle myShape = new Rectangle(width, height);  
myShape.OutputArea();  
Console.ReadLine();

Area output: 50

1. What goes in a utility/helper class (which is usually a static class)?

Functions/operations that are repetitive go into utility/helper classes.

1. Explain the concept of inheritance.

Inheritance is a way to make several similar classes that are not exactly the same. You can create a general Car class from which all car classes will inherit, but more specific Car classes like a Volvo, or BMW class.

1. How does a member of a class have to be designated (or marked) in order to allow overrides?

You must use the “override” keyword.

1. True or False: One class can inherit from several other classes at the same time.

False

1. What is a namespace?

A namespace is like a library of particular functions/methods that a programmer might utilize.

1. What does a *using statement* do?

The using statement is used in order to keep from writing out a complete method name each time you want to use that method.

Utilize the .NET API Browser from Microsoft (<https://docs.microsoft.com/en-us/dotnet/api/?view=netframework-4.8>) to locate System Namespace to answer the following questions.

1. How many different ways can the System.Console.WriteLine method be overloaded?

18 different ways

1. The System.Console class contains an event called CancelKeyPress. This event establishes a way to exit the Console. What is one of the valid key combinations to trigger this event?

Ctrl + C

1. What is the basic description of what the System.Console.ReadLine method does?

Reads the next line of characters from the standard input stream.