Unit 1 - Exercises

Introduction to .NET

Questions

1. List features of the .NET Framework.
2. What is the Common Intermediate Language (CIL)?
3. What was the CIL formerly called and what is its acronym?
4. What is an assembly? What are the two types mentioned in the lesson?
5. What is the Common Language Runtime (CLR)?
6. What is Just-in-time (JIT) compilation?
7. Explain how Visual Studio solutions are organized.
8. What are the three project types used in this course? What assemblies are they compiled to?

Classes and Objects

**EX1**. Choose the class item for each of the items below for the specified noun. Example: Color: Attribute

Noun: Standard Playing Card

* Rank: **\_\_\_\_***\_\_*
* Flip: **\_\_\_\_***\_\_*
* PlayingCard: **\_\_\_\_***\_\_*
* Suit: **\_\_\_\_**\_\_\_\_\_

**EX2**. Fill in the blanks to create an instance of yourself. Include three attributes of your type and their states.

* **\_\_\_\_\_\_ (instance of \_\_\_\_\_\_**\_)
* **\_\_\_\_\_\_ : \_\_\_\_\_\_**
* **\_\_\_\_\_\_ : \_\_\_\_\_\_**
* **\_\_\_\_\_\_ : \_\_\_\_\_\_**

**EX3**. Provide the names for the behaviours which would report and modify the state of the listed attributes for the specified instance below.

Sony Bravia (instance of Television)

Attributes and State:

* Channel : 1
* Volume : 0

Behaviours:

* **\_\_\_\_***\_\_* (Retrieve)
* **\_\_\_\_***\_\_* (Modify)
* **\_\_\_\_***\_\_* (Retrieve)
* **\_\_\_\_***\_\_* (Modify)

Visual Studio Solutions and Projects

**EX4**. Create a new Visual Studio project.

1. Launch Visual Studio (if not already launched)
2. On the main menu, select File > New > Project…
3. Select Visual C# (panel on the left) > Console Application (panel in the middle)
4. Enter Unit1Exercises in the Name text field.
5. Do no modify the Location text field.
6. Enter ADEV2005Unit1 in the Solution Name text field.
7. Enable Create directory for solution (if it is not already enabled).
8. Leave all other inputs as their defaults.
9. Click the “Ok” button.

Questions

1. What is the file name (with file extension) for the Visual Studio solution created in exercise 4?
2. What is the file name (with file extension) for the project created in exercise 4?
3. What items would you find in a Console Application project when it’s first created?
4. What is the “bin” directory used for?

**EX5**. Load a Visual Studio solution.

1. Close all instances of the Visual Studio application.
2. Using the Windows File Explorer application, locate the .sln file created when you completed exercise 4.
   * If you do not see file extensions in File Explorer, you can enable this from the applications ribbon under View.
3. Double-click the .sln file. This will launch Visual Studio, load the solution and all of the projects within that solution.

Coding in Visual Studio

**EX6**. Add a code file to a project. This exercise is a continuation of the previous exercise.

1. Right-click the “Unit1Exercises” project node in the Solution Explorer. Select Add > New Item… from the context menu.
2. In the new item dialog window, select the Code category from the left panel. This will filter the template list to only display templates under the category “Code”.
3. In the middle panel, select Code File. A code file is a template with no predefined code.
4. Enter PlayingCard.cs in the “Name” text field.
5. Click the “Add” button.

Questions

1. What is the relative path to the PlayingCard.cs file from the solution root directory?

**EX7**. Modify source code file and declare the GameLibrary namespace.

1. Select the PlayingCard.cs file in the Solution Explorer.
2. Open the file for editing (Code View) using one of the following techniques:
   * F7
   * Click the “View Code” toolbar item in the toolbar of the Solution Explorer panel.
   * Main Menu > View > Code
3. Code the using statement to use members of the System namespace.
4. Code the declaration of the namespace GameLibrary.

**EX8**. Declare the PlayingCard class within the GameLibrary namespace. Ensure you add XML documentation for the class.

**EX9**. Declare the following fields for the PlayingCard class.

- rank : string

- suit : string

- isFaceUp : bool

**EX10**. Build (compile) the “Unit1Exercises” project.

1. Build the project using the following techniques:
   * Main menu > Build > Build Unit1Exercises
   * Right-click the Unit1Exercises project in Solution Explorer. Select “Build” from the context menu.

Questions

1. What is generated after building a Console Application project?
2. What is the name of the file (with file extension) of the assembly generated after building the Unit1Exercises project?

**EX11**. Code constructor methods for the PlayingCard class.

+ PlayingCard(rank : string, suit : string) - Initializes a PlayingCard with the specified rank and suit. The card will be face up.

+ PlayingCard(rank : string, suit : string, isFaceUp : bool) - Initializes a PlayingCard with the specified rank, suit and whether the PlayingCard is face up or down.

**Reminder**: Add XML documentation for each method.

**EX12**. Code behaviours (methods) for the PlayingCard class.

+ GetRank() : string - Returns the rank of the PlayingCard.

+ GetSuit() : string - Returns the suit of the PlayingCard.

+ IsFaceUp() : bool - Returns whether the PlayingCard is face up or down.

+ Flip() : void - Flips the PlayingCard.

+ ToString() : string - Returns the String representation of the PlayingCard.

Format the String as:

{rank} {suit} [Face {Up/Down}]

Examples:

Ace Spade [Face Up]

Two Heart [Face Down]

**Reminder**: Add XML documentation for each method.

Constructing Objects

**EX13**. In the Program.cs file, construct two instances of the PlayingCard class.

1. Add a using statement to Program.cs. This is necessary to use the PlayingCard class.
2. In the Main method:
   * Code the construction of PlayingCard using the constructor PlayingCard(string, string).
   * Print the PlayingCard to confirm the constructor works correctly.
   * Code the construction of a second PlayingCard using the constructor PlayingCard(string, string, bool).
   * Print the second PlayingCard to confirm the constructor works correctly.
3. Add code to prevent the console window from closing.
4. Build the project.
5. Run the application using one of the following techniques:
   * F5
   * In the toolbar, click the “Start Debugging” (Play) button.

Modify an Object’s State

**EX14**. Perform object behaviours to test the object.

1. Using three separate statements, print the state of each attribute for the first PlayingCard.
2. Using a single statement, print the state of the second PlayingCard.
3. Flip each card.
4. Print each card a second time.

Enumerations

**EX15**. Code an enumeration called CardRank.

1. Add a new code file to the project. If you forget how to create a new code file, review exercise 5.
2. Code the CardRank enumeration within the GameLibrary namespace. The CardRank enumeration has the following values: Ace, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King.
3. Build the project.

Questions

1. What is the numeric value of the seventh item in the CardRank enumeration?
2. How would you code the literal value of King in the CardRank enumeration?
3. If numeric values are not assigned to the items within the enumeration, what values will they be assigned?

**EX16**. Code an enumeration called CardSuit. The steps are the same as the steps for exercise 15.

1. Code the CardSuit enumeration within the GameLibrary namespace. The CardSuit enumeration has the following values: Heart, Diamond, Club, Spade.

**EX17**. Update the PlayingCard class to use the CardRank and CardSuit enumerations.

1. Update the PlayingCard class such that it does not use a string type for rank and suit. The CardRank and CardSuit enumerations will be used in their place.
2. Update the Program class to accommodate the use of these enumerations.