General Instructions for Week 1 Activities

1. Start Microsoft Visual Studio.
2. Click File > New > Project…. Give solution the Name as asked in activity description. As the solution Location provide the path to the local Git repository (C:\git\net-bootcamp-activities). Make sure you uncheck the Create new Git repository.

Hints/Tips

1. Remember motivation comes from within. Take responsibility for your own learning.
2. Take notes from each presentation and activity.
3. If you don't understand something, ask questions.
4. Participate in course discussions.
5. Practice what you've learned to keep your skills sharp.

Activity Hints and Tips

Focus your efforts! Do not be overwhelmed by the lines of code that you see for these initial activities. One of traits of a good developer is the ability to isolate code that is relevant to the situation. C#'s design philosophy also encourages this.

Activity 1

Activity Overview

During this you will practice simple logic formulation. The activity requires that you complete two methods that have very specific expected results based on the data that is given to them.

Activity Instructions

Activity 1.0

1. In Visual Studio create new Visual C# > Windows > Classic Desktop > Class Library named **Activity1**. Name the solution also as **Activity1**. When the solution has been loaded, delete the existing file ***Class1.cs***.
2. For the project **Activity1** use Add > Existing Item… command to add the existing ***PlayerProfile.cs*** file (from **Week 1 Activity Code Files** folder).
3. Go through the method ***public int ComputeAge()*** which is a helper method and is designed to compute a player's age based on the birth date given and the current computer date. Your task is to complete the implementation of this method.

   Note: Refer to the **#region** markers within the source file for detailed instructions.
4. You must not use the ***DateTime*** class from .NET class library. The relevant variables and instance of a date/time have already been declared for you. Based on the provided variables compute the current age of the player in years and assign it to the variable ***tempAge***.
5. To test your solution for correctness, add new test project to your existing **Activity1_0** solution: File > New > Project… and select Visual C# > Test > Unit Test Project. Name it as **Activity1.Tests** and make sure you are adding it to the existing solution **Activity1**. This project uses Visual Studio Unit Testing framework, to automate testing of types.
6. Delete ***UnitTest1.cs*** file.
7. In the **Activity1.Tests** project add existing test file ***Activity1_0_Tests.cs*** (from **Week 1 Activity Code Files**).
8. In the **Activity1.Tests** project add a reference to the project **Activity1**.
9. To activate unit tests, open TEST > Windows > Test Explorer and click Run All – if all tests pass green, then your solution is correct. If some tests fail, try to infer why your solution behaves differently.

10. Inform the facilitator by committing to local Git repository and synchronizing with remote TFS Git repository (Commit All and Sync) with the comment “Finished Activity 1.0”. Proceed to the Activity 1.1 below.

Activity 1.1

1. Add existing ***Zodiac.cs*** file (from **Week 1 Activity Code Files** folder) to the project **Activity1**.
2. Go through the method ***public Zodiac GetZodiacSign(DateTime)*** which is a method that returns the appropriate **Zodiac** object based on the passed **DateTime** object.
3. As with the previous activity, you must not work with the **DateTime** object itself. The appropriate variables have already been declared and initialized for you. Your task will be to write code that will return the appropriate **Zodiac** object based on those variables.

   Note: Refer to the **#region** markers within the source file for detailed instructions. For implementation try to use only fundamental C# statements (e.g., **if-else**, **switch**, **return**) and existing objects (**ARIES**, **TAURUS**, etc.) and variables (**month**, **day**). Avoid definition of additional arrays or other objects.
4. Add existing **Activity1_1_Tests.cs** file (from **Week 1 Activity Code Files** folder) to the test project **Activity1.Tests**. This file contains code to test **GetZodiacSign()** method.
5. From the Tests Explorer window run all tests. If all tests pass green, then your solution is correct. If some tests fail, try to resolve why your solution behaves differently.
6. Inform the facilitator by committing to the Git repository with the comment “Finished Activity 1.1”

