**Spaceman**

**Spaceman**

**Overview**

This project is slightly different than others you have encountered thus far on Codecademy. Instead of a step-by-step tutorial, this project contains a series of open-ended requirements which describe the project you’ll be building. There are many possible ways to correctly fulfill all of these requirements, and you should expect to use the internet, Codecademy, and other resources when you encounter a problem that you cannot easily solve.

**Project Goals**

Invaders from outer space have arrived and are abducting humans using tractor beams. Players must crack the codeword to stop the abduction!

Ok, we’ll admit it’s quite a bit like that classic game, [“Hangman”](https://en.wikipedia.org/wiki/Hangman_%28game%29), but with a better premise. Plus, building this command-line game was the Codecademy 2019 Software Engineer Internship Backend Programming Challenge!

**Setup Instructions**

If you choose to do this project on your computer instead of Codecademy, you can download what you’ll need by clicking the “Download” button below. If you’re using Visual Studio, you’ll define the methods in **Game.cs** and run the code to see your game in action. If you need help setting up on your own computer, read our [article about running C# with Visual Studio Community](https://www.codecademy.com/courses/learn-c-sharp/articles/csharp-go-off-platform).

**1.**

In order to complete this project, you should have completed the first 6 sections of [Learn C#](https://www.codecademy.com/learn/learn-c-sharp) (through Learn C#: Classes).

### Project Requirements

**2.**

You’ll design a Game class in **Game.cs**, which will be used in the Main() method in **Program.cs**. Your code will primarily focus on the game logic — we’ve already provided a Ufo class that you can use to draw the spaceship with each wrong guess.

Check the GIF below to get a sense of what you’ll be building. The game is started with dotnet run. In the game, the user guesses one character at a time. The game ends once the word is complete or the use makes five wrong guesses. The game can be shut down early by pressing CTRL + C.

**3.**

In **Game.cs**, make a Game class within the Spaceman namespace.

Make a Greet() method that will be called at the beginning of each. It should print out a greeting to the player.

4. In the Game class we’ll need to keep track of these six items:

* Codeword — the word players will guess
* Current word — to be filled in as the player guesses correctly
* Maximum number of guesses — the number of wrong guesses before the abduction is finished
* Current number of wrong guesses — increases with each guess that is not in the codeword
* An array containing a few options for the codeword — so that the codeword isn’t the same every game
* An instance of the Ufo class — to print the UFO images

Make a property/field for each.

**5.**

Create a constructor that:

* Sets the codeword to a random value in the array of strings
* Sets the maximum guesses to 5 — there are only 5 wrong guesses before the abduction is complete
* Sets the wrong guesses to 0
* Sets the current word to a string of underscores (\_) that is the same length as the codeword

**6.**

We need a way to check if the player won. Create a DidWin() method that will return true when the current word matches the codeword.

To compare strings, use the Equals() method:

CodeWord.Equals(CurrentWord);

**7.**

We also need to check if the player lost. Create a DidLose() method that returns true when the number of wrong guesses is greater than or equal to the max number of guesses.

**8.**

Next, create a Display() method that will print all the necessary game information to the screen. That includes:

* The UFO
* Current word
* Number guesses remaining

The string version of a Ufo can be accessed with its own Stringify() method.

**9.**

With everything else in place, it’s time to write the code for the interaction. Define an Ask() method that:

* Asks the user to guess a letter and captures their input
* Checks that the input is length of 1. If it is not, tell the user to input one letter at a time and use return to break out of the method
* Checks if the codeword Contains() the guess
* If it does, tell the user, find all instances of the guess in the codeword, and replace it with the guess, e.g. if the guess is g and it appears in the first and last position, then replace the first and last letters with g
* If the codeword does not contain the guess, tell the user, increase the number of wrong guesses by 1, and call the AddPart() method on the Ufo object

To replace a letter in a string use the Remove() and Insert() pattern:

someWord = someWord.Remove(index, 1).Insert(index, "z");

**10.**

Your Game class is complete! Let’s use it in **Program.cs**:

* Create a new Game object and have it Greet() the player
* Create a loop that calls the object’s Display() and Ask() methods. The loop should run until DidWin() or DidLose() returns true
* It should print a message if the user wins or loses

### Extensions & Solution Code

**11.**

Great work! If you’d like to see the solution, move to the next task. If you’d like to extend your project on your own, you could consider the following:

* The game doesn’t record past guesses and doesn’t check if a user has already guessed a letter. Add that feature.
* Create your own spaceship design! Go into **Ufo.cs** and read through it. Edit the code as needed.

**12.**

Visit [our forums](https://discuss.codecademy.com/t/spaceman-challenge-project-c/462367) to compare your project to our sample solution code. You can also learn how to host your own solution on GitHub so you can share it with other learners! Your solution might look different from ours, and that’s okay! There are multiple ways to solve these projects, and you’ll learn more by seeing others’ code.