



## Exercise

Let's get a little practice using the **private** keyword by **creating a small Hi-Lo game**. The game starts with a pot of 10 bucks, and it picks a random number from 1 to 10. The player will guess if the next number will be higher or lower. If the player guesses right they win a buck, otherwise they lose a buck. Then the next number becomes the current number, and the game continues.

Go ahead and **create a new console app** for the game. Here's the Main method:

```
public static void Main(string[] args)
{
    Console.WriteLine("Welcome to HiLo.");
    Console.WriteLine($"Guess numbers between 1 and {HiLoGame.MAXIMUM}.");
    HiLoGame.Hint();
    while (HiLoGame.GetPot() > 0)
    {
        Console.WriteLine("Press h for higher, l for lower, ? to buy a hint,");
        Console.WriteLine($"or any other key to quit with {HiLoGame.GetPot()}.");
        char key = Console.ReadKey(true).KeyChar;
        if (key == 'h') HiLoGame.Guess(true);
        else if (key == 'l') HiLoGame.Guess(false);
        else if (key == '?') HiLoGame.Hint();
        else return;
    }
    Console.WriteLine("The pot is empty. Bye!");
}
```

*Don't forget—it's  
not cheating to  
peek at the solution!*

Next, add a **static class** called **HiLoGame** and **add the following members**. Since this is a static class, all of the members need to be static. Make sure to include either **public** or **private** in the declaration for each member:

1. A constant integer **MAXIMUM** that defaults to 10. Remember, you can't use the **static** keyword with constants.
2. An instance of **Random** called **random**.
3. Two **int** fields called **currentNumber** and **nextNumber** that are both initialized to random numbers.
4. An **int** field called **pot** with the number of bucks in the pot. **Make this field private.** ←

*We made pot private because we don't want other classes to be able to add money, but the Main method still needs to be able to print the size of the pot to the console. Look carefully at the code in the Main method—can you figure out how to let the Main method get the value of the pot field without giving it a way to set the field?*

5. A **method** called **Guess** with a **bool** parameter called **higher** that does the following (look closely at the Main method to see how it's called):
  - If the player guessed higher and the next number is  $\geq$  the current number **OR** if the player guessed lower and the next number is  $\leq$  the current number, it writes "You guessed right!" to the console and increments the pot.
  - Otherwise, it writes "Bad luck, you guessed wrong." to the console and decrements the pot.
  - It sets **currentNumber** to **nextNumber**, then sets **nextNumber** to a new random number for the player to guess.
  - It writes "The current number is {currentNumber}" to the console.
6. A **Hint** method that finds half the maximum and writes either "The current number is {currentNumber}, the next is at least {half}" or "The current number is {currentNumber}, the next is at most {half}" then decrements the pot.

**BONUS QUESTION:** If you make **HiLoGame.random** a public field, can you figure out a way to use what you know about how the **Random** class generates its numbers **to help you cheat at the game?**



## Exercise Solution

Here's the rest of the code for the Hi-Lo game. The game starts with a pot of 10 bucks, and it picks a random number from 1 to 10. The player will guess if the next number will be higher or lower. If the player guesses right they win a buck, otherwise they lose a buck. Then the next number becomes the current number, and the game continues.

Here's the code for the HiLoGame class:

```
static class HiLoGame
{
    public const int MAXIMUM = 10;
    private static Random random = new Random();
    private static int currentNumber = random.Next(1, MAXIMUM + 1);
    private static int nextNumber = random.Next(1, MAXIMUM + 1);
    private static int pot = 10;

    public static int GetPot() { return pot; }

    public static void Guess(bool higher)
    {
        if ((higher && nextNumber >= currentNumber) ||
            (!higher && nextNumber <= currentNumber))
        {
            Console.WriteLine("You guessed right!");
            pot++;
        }
        else
        {
            Console.WriteLine("Bad luck, you guessed wrong.");
            pot--;
        }
        currentNumber = nextNumber;
        nextNumber = random.Next(1, MAXIMUM + 1);
        Console.WriteLine($"The current number is {currentNumber}");
    }

    public static void Hint()
    {
        int half = MAXIMUM / 2;
        if (nextNumber >= half)
            Console.WriteLine($"The current number is {currentNumber}," +
                              $" the next number is at least {half}");
        else Console.WriteLine($"The current number is {currentNumber}," +
                              $" the next is at most {half}");
        pot--;
    }
}
```

If you try to add the `static` keyword to a constant you'll get a compiler error because all constants are static. Try adding one to any class—you can access it from another class just like any other static field.

The `pot` field is private, but the `Main` method can use the `GetPot` method to get its value without having a way to modify it.

**This is a good example of encapsulation. You protected the `pot` field by making it private. It can only be modified by calling the `Guess` or `Hint` methods, and the `GetPot` method provides read-only access.**

↑  
This is an important point. Take a few minutes to really figure out how it works.

The `Hint` method needs to be public because it's called from `Main`. Notice how we didn't include the curly brackets for the `if/else` statement? An `if` or `else` clause that only has a single line doesn't need brackets.

**BONUS:** You can replace the public `random` field with a new instance of `Random` that you initialized with a different seed. Then you can use a new instance of `Random` with the same seed to find the numbers in advance!

```
HiLoGame.random = new Random(1);
Random seededRandom = new Random(1);
Console.Write("The first 20 numbers will be: ");
for (int i = 0; i < 10; i++)
    Console.Write($"{seededRandom.Next(1, HiLoGame.MAXIMUM + 1)}, ");
```

Every instance of `Random` initialized with the same seed will generate the same sequence of pseudo-random numbers.