Null and Unassigned References

So far we've seen:

A reference to an object

Multiple references to an object

What about a reference that refers to no object? In C# a reference to no object is called either a *null reference* or *unassigned*. We'll need to apply these concepts in C# whenever we want to show that a reference is "missing", create a reference variable without defining it, or initialize an empty array.

In the first use case, we'd like to create a reference that is "missing" or empty. We set it equal to the keyword `null`:

```
Diary dy = null;
```

In the second case, if we create a reference variable without a value, it is unassigned:

```
Diary dy;
// dy is unassigned
```

In the third case, if we create an empty array of reference types, each element is an unassigned reference:

```
Diary[] diaries = new Diary[5];
// diaries[1] is unassigned, diaries[2] is unassigned, etc.
```

Be careful when checking for `null` and unassigned references. We can only compare a `null` reference if it is explicitly labeled `null`:

```
Diary dy = null;
Console.WriteLine(dy == null);
// Output: true
```

For the other two cases, comparing an unassigned variable we'll get an error:

```
Object o;
Console.WriteLine (o == null);
// error CS0165: Use of unassigned local variable 'o'
```

This might seem annoying at first, but it's actually a good thing: the C# compiler prevents future issues down the road by raising an error the first time an unassigned variable is used.

☑Instructions

1.
Create a variable of type `Book` and set it to `null`.
2.
Print the variable to the console.

Remember that `null` presents a null reference, so you should see nothing printed.
3.
Compare the variable to `null` using the `==` operator and print the result to the console.

Hint

Here's an example of comparing a null `Forest` reference to `null`:

```
Forest f = null;
Console.WriteLine(f == null);
// Output: True
```