REFERENCE FUNDAMENTALS

Polymorphism

We just saw how useful it is to have the same interface for multiple data types. This is a common concept across many programming languages, and it's called *polymorphism*.

The concept really includes two related ideas. A programming language supports polymorphism if:

> Objects of different types have a common interface (interface in the general meaning, not just a C# `interface`), and
>
> The objects can maintain functionality unique to their data type

Let's prove to ourselves that this is true in C#. We'll use the example of `Stringify`: `Dissertation` and `Book` have different `Stringify()` methods but can both be referenced as `Book`s.

Here are snippets from each class:

```
class Dissertation : Book
{
  public override string Stringify()
  {
    return "This is a Dissertation object!";
  }
}


class Book
{
  public virtual string Stringify()
  {
    return "This is a Book object!";
  }
}
```

Given that information, what will the below program print?

```
Book bk = new Book();
Book bdiss = new Dissertation();
```

```
Console.WriteLine(bk.Stringify());
Console.WriteLine(bdiss.Stringify());
```

The answer is:

```
This is a Book object!
This is a Dissertation object!
```

Even though `bk` and `bdiss` are both `Book` type references, their behavior is different. `Dissertation` overrides the `Stringify()` method, so all `Dissertation` objects (regardless of reference type) will use that method.

Therefore, C# support polymorphism!

You'll never have to write `polymorphism` in your code, but this vocabulary is essential to communicating with other developers!

So remember: *polymorphism* is the ability in programming to present the same interface for differing data types.

☑**Instructions**

1.
In **Program.cs**, there are `Book` type references to one `Book` and one `Diary` object. First, call `b1.Stringify()` and print it to the console.
2.
Below that call `b2.Stringify()` and print it to the console.

To check your understanding, find both `Stringify()` methods in **Diary.cs** and **Book.cs**.

Hint

Even though they are both `Book` references, the underlying objects have different definitions of `Stringify()`.