Method Overloading

Say you want to use `Math.Round()`, a built-in method. You go to the Microsoft documentationto learn how to use it, and find at least 8 different versions! They all have the same name: `Math.Round()`.

What's happening here is called *method overloading*, and each "version" is called an *overload*. Though they have the same name, the *overloads* are different because they have either (i) different parameter types or (ii) different number of parameters. This is useful if you want the same method to have different behavior based on its inputs.

Let's examine this concept with these two overloads: `Math.Round(Double, Int32)` and `Math.Round(Double)`.

The first overload, `Math.Round(Double, Int32)`, rounds the `double` to the `int`'s number of decimal points

```
Math.Round(3.14159, 2); // returns 3.14
```

The second, `Math.Round(Double)`, rounds the `double` to the nearest integer.

```
Math.Round(3.14159); // returns 3
```

In C#, when we say that the methods are "different", we are really talking about their method *signatures*, which is the method's name and parameter types in order.

For example, both methods above are named `Round()`, but one has `Double` and `Int32` parameters, and the other has a `Double` parameter.

☑Instructions

1.

Let's practice implementing our own overloads. Let's build a method `NamePets` with two overloads.

First write a method `NamePets()` that takes two `string` arguments.

If you call it, like:

```
NamePets("Laika", "Albert");
```

it should announce the newly named pets in the console, like:

```
Your pets Laika and Albert will be joining your voyage across space!
```

Hint

Here's an example method definition:

```
static void YourMethodName(string message, int age)
```

Your method should also be `static` and `void`.

2.
Then write another method `NamePets` that takes three `string` arguments. When you call it:

```
NamePets("Mango", "Puddy", "Bucket");
```

it should announce the newly named pets in the console, like:

```
Your pets Mango, Puddy, and Bucket will be joining your voyage across space!
```

Hint

Here's an example method definition:

```
static void YourMethodName(string message, int age, bool isEmployed)
```

Your method should also be `static` and `void`.

3.
Add a third `NamePets` method with zero arguments. When you call it:

```
NamePets();
```

it should empathize with you, like:

```
Aw, you have no spacefaring pets :(
```

Hint

Here’s an example method definition:

```
static void YourMethodName()
```

Your method should also be `static` and `void`.

4.
Call each method overload once in `Main()`.

Hint

In `Main()`, call `NamePets()` with two arguments, three arguments, and zero arguments.