Exercise

Create a program with an Elephant class. Instantiate two Elephant instances and then swap the reference values that point to them, ***without*** getting any Elephant instances garbage-collected. Here's what it will look like when your program runs.

You're going to build a new console app that has a class called Elephant.

Here's an example of the output of the program:

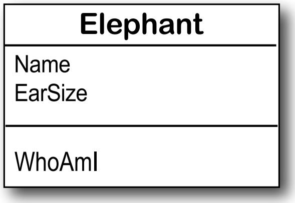
```
Press 1 for Lloyd, 2 for Lucinda, 3 to swap
You pressed 1
Calling lloyd.WhoAmI()
My name is Lloyd.
My ears are 40 inches tall.

You pressed 2
Calling lucinda.WhoAmI()
My name is Lucinda.
My ears are 33 inches tall.

You pressed 3
References have been swapped

You pressed 1
Calling lloyd.WhoAmI()
My name is Lucinda.
My ears are 33 inches tall.

You pressed 2
Calling lucinda.WhoAmI()
My name is Lloyd.
My ears are 40 inches tall.

You pressed 3
References have been swapped

You pressed 1
Calling lloyd.WhoAmI()
My name is Lloyd.
My ears are 40 inches tall.

You pressed 2
Calling lucinda.WhoAmI()
My name is Lucinda.
My ears are 33 inches tall.
```

The Elephant class has a WhoAmI method that writes these two lines to the console to display the values in the Name and EarSize fields.

Swapping the references causes the lloyd variable to call the Lucinda object's method, and vice versa.

Swapping them again returns things to the way they were when the program started.

Here's the class diagram for the Elephant class you'll need to create.



The CLR garbage-collects any object with no references to it. So here's a hint for this exercise: if you want to pour a cup of coffee into another cup that's currently full of tea, you'll need a third glass to pour the tea into.

Your job is to create a .NET Core console app with an Elephant class that matches the class diagram and uses its fields and methods to generate output that matches the example output.

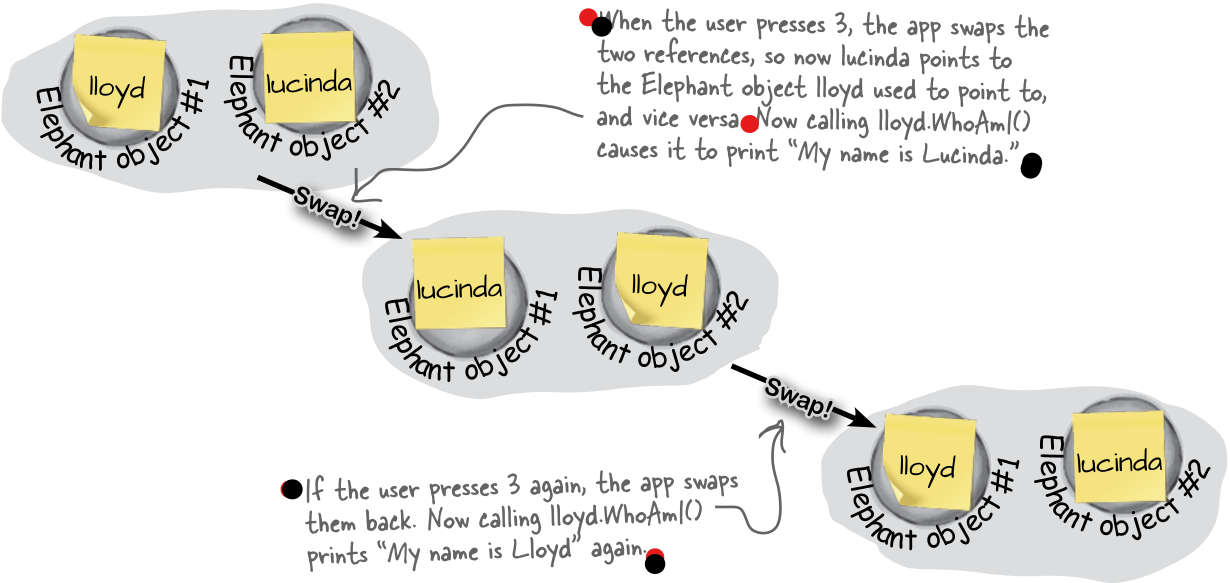
1. **Create a new .NET Core console app and add the Elephant class.**
   Add an Elephant class to the project. Have a look at the Elephant class diagram—you'll need an int field called EarSize and a string field called Name. Add them, and make sure both are public. Then add a method called WhoAmI that writes two lines to the console to tell you the name and ear size of the elephant. Look at the example output to see exactly what it's supposed to print.

2. **Create two Elephant instances and a reference.**
   Use object initializers to instantiate two Elephant objects:

```
Elephant lucinda = new Elephant() { Name = "Lucinda", EarSize = 33 };
Elephant lloyd = new Elephant() { Name = "Lloyd", EarSize = 40 };
```

3. **Call their WhoAmI methods.**
   When the user presses 1, call lloyd.WhoAmI. When the user presses 2, call lucinda.WhoAmI. Make sure that the output matches the example.

4. **Now for the fun part: swap the references.**
   Here's the interesting part of this exercise. When the user presses 3, make the app call a method that ***exchanges the two references***. You'll need to write that method. After you swap references, pressing 1 should write Lucinda's message to the console, and pressing 2 should write Lloyd's message. If you swap the references again, everything should go back to normal.



When the user presses 3, the app swaps the two references, so now lucinda points to the Elephant object lloyd used to point to, and vice versa. Now calling lloyd.WhoAmI() causes it to print "My name is Lucinda."

If the user presses 3 again, the app swaps them back. Now calling lloyd.WhoAmI() prints "My name is Lloyd" again.