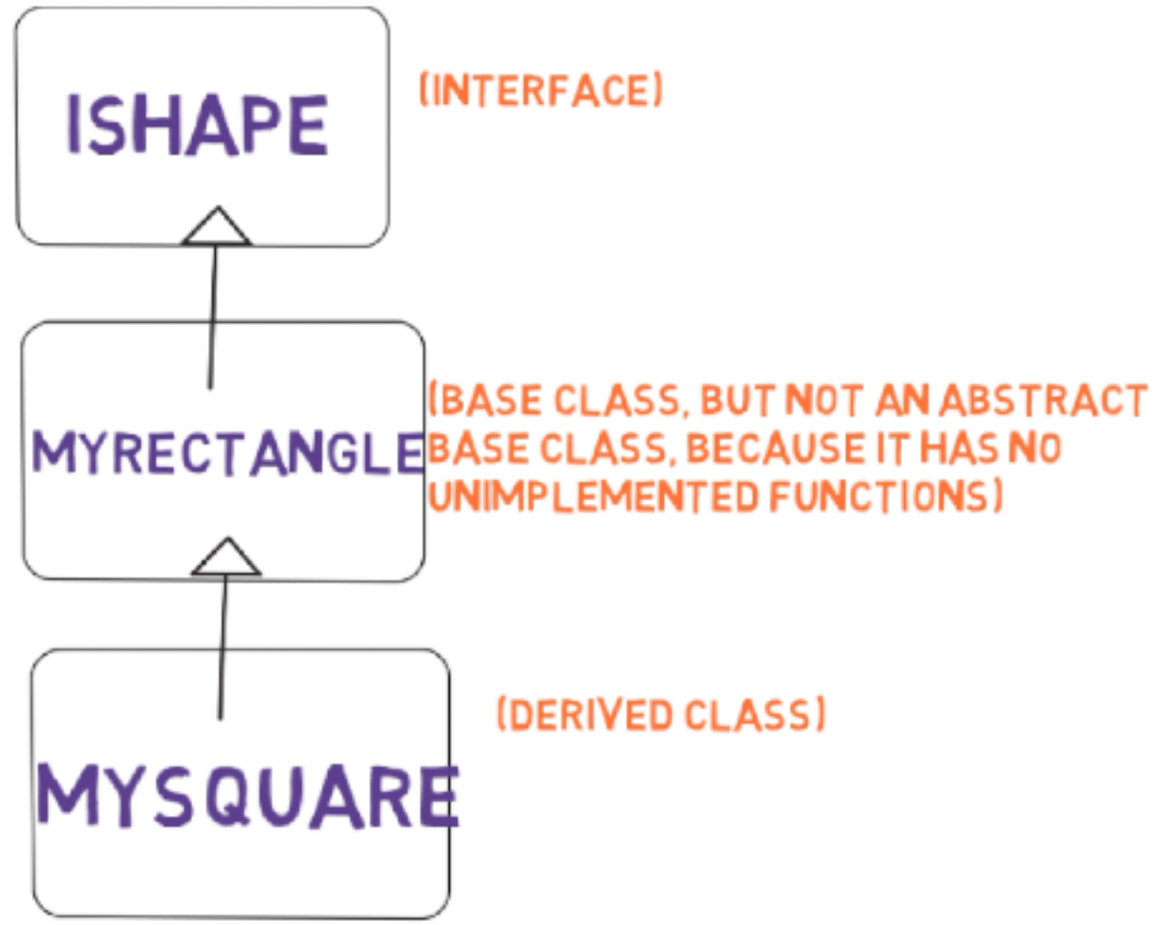
```
public interface IShape {

    public String introduceYourself();
```




```
public String introduceYourself() {
    return "I am a rectangle";
}
```

```
public String introduceYourself() {
    return "I am a square";
}
```

OK - NOW WILL THE CODE BELOW WORK, AND IF SO WHAT WILL IT PRINT?

```
IShape someRectangle = new MyRectangle(5,10);
IShape someSquare = new MySquare(5);
MyRectangle someOtherRectangle = new MySquare(7);

System.out.println(someRectangle.introduceYourself());
System.out.println(someSquare.introduceYourself());
System.out.println(someOtherRectangle.introduceYourself());
```

OK - NOW WILL THE CODE BELOW WORK, AND IF SO WHAT WILL IT PRINT?

```
IShape someRectangle = new MyRectangle(5,10);
IShape someSquare = new MySquare(5);
MyRectangle someOtherRectangle = new MySquare(7);

System.out.println(someRectangle.introduceYourself());
System.out.println(someSquare.introduceYourself());
System.out.println(someOtherRectangle.introduceYourself())
```

YEP EACH LINE WILL WORK.

THAT IS BECAUSE A SQUARE IS-A RECTANGLE, AND RECTANGLE IS-A SHAPE, AND SQUARE IS-A SHAPE

"I AM A RECTANGLE"

"I AM A SQUARE"

"I AM A SQUARE"
(JAVA WAS SMART ENOUGH TO FIGURE OUT THE TYPE OF THE OBJECT AT RUNTIME!)

(THIS IS A TRICKY ONE - THE VARIABLE IS OF TYPE RECTANGLE, BUT WAS INITIALIZED WITH AN OBJECT OF TYPE SQUARE)

GIVEN A CLASS HIERARCHY WITH OVERRIDDEN METHODS, JAVA WILL FIGURE OUT THE CORRECT VERSION OF THE FUNCTION TO CALL AT RUNTIME

THIS IS CALLED DYNAMIC METHOD DISPATCH

"DYNAMIC" BECAUSE THE WORK TO FIGURE OUT WHICH VERSION OF THE METHOD TO CALL IS DONE AT RUNTIME

METHODS CALLED THIS WAY ARE CALLED VIRTUAL METHODS

USE THE "FINAL" KEYWORD TO MARK A MEMBER FUNCTION AS "NOT-VIRTUAL"

BTW, IN C++, THE DEFAULT IS FOR METHODS TO NOT BE VIRTUAL UNLESS EXPLICITLY STATED AS SUCH. IN JAVA THE DEFAULT IS FOR ALL MEMBER FUNCTIONS TO BE VIRTUAL

BTW, USING A VARIABLE OF A BASE CLASS TO HOLD AN OBJECT OF A DERIVED CLASS IS PERFECTLY OK

THIS IS CALLED UPCASTING

```
IShape someRectangle = new MyRectangle(5,10);
IShape someSquare = new MySquare(5);
MyRectangle someOtherRectangle = new MySquare(7);

System.out.println(someRectangle.introduceYourself());
System.out.println(someSquare.introduceYourself());
System.out.println(someOtherRectangle.introduceYourself());
```

BUT USING A VARIABLE OF A DERIVED CLASS TO HOLD AN OBJECT OF THE BASE CLASS IS NOT OK IN GENERAL

BTW, USING A VARIABLE OF A BASE CLASS TO HOLD AN OBJECT OF A DERIVED CLASS IS PERFECTLY OK

THIS IS CALLED UPCASTING

```
IShape someRectangle = new MyRectangle(5,10);
IShape someSquare = new MySquare(5);
MyRectangle someOtherRectangle = new MySquare(7);

System.out.println(someRectangle.introduceYourself());
System.out.println(someSquare.introduceYourself());
System.out.println(someOtherRectangle.introduceYourself());
```

BUT USING A VARIABLE OF A DERIVED CLASS TO HOLD AN OBJECT OF THE BASE CLASS IS NOT OK IN GENERAL

SO, USE THE INSTANCEOF OPERATOR FIRST TO ASCERTAIN THAT THE SHAPE REALLY IS A SQUARE

```
IShape someShape = new MyCircle(10);
System.out.println(someShape.getArea());

if (someShape instanceof MyCircle) {
    MyCircle someCircle = (MyCircle) someShape;
} else if (someShape instanceof MyRectangle) {
    MyRectangle someCircle = (MyRectangle) someShap
} else if (someShape instanceof MySquare)
    MySquare someCircle = (MySquare) someShape;
}
```

IF YOU TRY TO FORCE A 'SHAPE' TO ACT LIKE A 'SQUARE', IT WILL ONLY WORK IF THAT SHAPE REALLY IS A SQUARE.

UPCASTING IS FINE, BUT DOWNCASTING IS RISKY – AVOID IT IF POSSIBLE