

INTERFACES: INTRODUCTION

BTW, THE FUNCTION SIGNATURE
DOES NOT INCLUDE THE RETURN
TYPE OR ACCESS MODIFIER
(PUBLIC, PRIVATE ETC)

**AN INTERFACE IS A
WAY TO DRIVE BEHAVIOUR**

AN INTERFACE IS A CLASS WITH ONLY
MEMBER FUNCTION SIGNATURES
BUT WITHOUT ANY
MEMBER FUNCTION IMPLEMENTATIONS

THE SIGNATURE OF A MEMBER
FUNCTION INCLUDES

1. MEMBER FUNCTION NAME
2. PARAMETER TYPES
3. EXCEPTIONS THROWN

```
public interface IShape {  
  
    public double getArea();  
    public double getPerimeter();  
  
}
```



**MEMBER FUNCTION
SIGNATURES**

CLASSES THEN IMPLEMENT INTERFACES

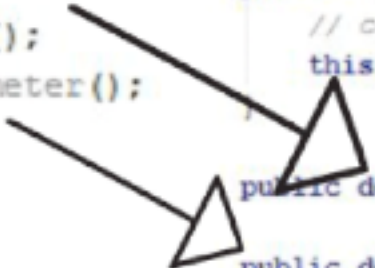
CLASSES THEN IMPLEMENT INTERFACES

A CLASS CAN IMPLEMENT AN INTERFACE,
WHICH MEANS THAT IT INCLUDES
IMPLEMENTATIONS FOR EVERY METHOD
SIGNATURE IN THE INTERFACE

NOTE THE USE
OF THE
"IMPLEMENTS"
KEYWORD

```
public class MyCircle implements IShape {  
    // notice the 'implements' keyword followed by an interface name.  
    // this means that our MyCircle class must have implementations of the  
    // two methods in the Interface IShape, namely getArea and getPerimeter  
  
    public static final double S_PI = 3.1415;  
    // static member variable common across all objects of class MyCircle  
    // the final indicates its value can't be changed  
  
    private double m_radius;  
  
    public MyCircle(double radius) {  
        // constructor must have the same name as the Class  
        this.m_radius = radius;  
    }  
  
    public double getArea() { return S_PI * m_radius * m_radius; }  
    public double getPerimeter() {  
        return S_PI * 2 * m_radius;  
    }  
}
```

```
public interface IShape {  
    public double getArea();  
    public double getPerimeter();  
}
```



WHEN A CLASS IMPLEMENTS AN INTERFACE,
IT IS COMMITTING TO ADHERE TO THE
BEHAVIOUR IN THE INTERFACE (I.E. TO
HAVING THOSE MEMBER FUNCTIONS
AVAILABLE FOR USE)

WHEN A CLASS IMPLEMENTS AN INTERFACE, THE CLASS "IS-A" OBJECT OF THAT INTERFACE

```
IShape myCircle = new MyCircle(10);
```

NOTE THAT AN INTERFACE CANNOT
BE INSTANTIATED DIRECTLY, IT
CAN ONLY BE INSTANTIATED VIA
AN OBJECT OF SOME CLASS THAT
IMPLEMENTS THAT INTERFACE

NOW WE COULD HAVE OBJECTS OF DIFFERENT
CLASSES, ALL OF WHICH IMPLEMENT AN
INTERFACE, AND TREAT THEM ALL THE SAME

```
IShape someShape = new MyCircle(10);  
System.out.println(someShape.getArea());  
someShape = new MyRectangle(5,10);  
System.out.println(someShape.getArea());
```

"POLYMORPHISM"

ABSTRACT BASE CLASSES

AN ABSTRACT CLASS IS A CLASS THAT
IS MARKED ABSTRACT (DUH)

AN ABSTRACT BASE CLASS MAY OR
MAY NOT CONTAIN ANY ABSTRACT
METHODS

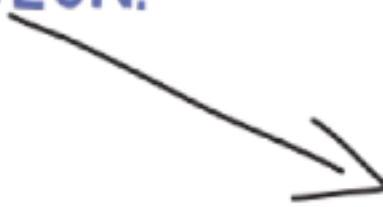
ERRM..AND WHAT MIGHT AN ABSTRACT
METHOD BE?

AN ABSTRACT METHOD IS A METHOD
WITHOUT AN IMPLEMENTATION

AND HOW IS SUCH A METHOD
ACTUALLY IMPLEMENTED IN JAVA?

AN ABSTRACT METHOD IS
MARKED ABSTRACT, HAS
NO CURLY BRACES, SIMPLY
A SEMI-COLON.

```
public abstract class GraphicObject {  
    // declare fields  
    // declare nonabstract methods  
    abstract void draw();  
}
```



ABSTRACT BASE CLASSES
CAN NEVER BE INSTANTIATED
BUT THEY CAN BE EXTENDED

GraphicObject graphicObject = new GraphicObject();

NO! THIS CODE WILL NOT COMPILE

**public class ConcreteGraphicObject
extends GraphicObject {**

YEP! THIS CODE WILL WORK FINE