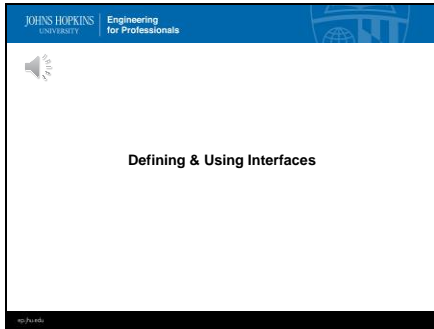
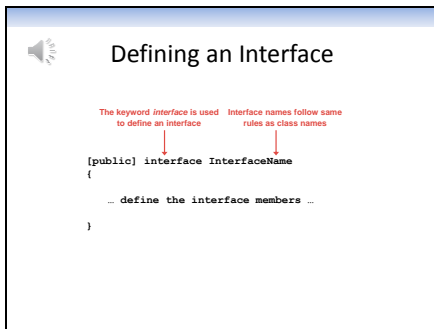


1



In this lecture you will learn how to define and use Java interfaces.

2



Java has many pre-defined interfaces, and you'll learn about some of them later on in this course.

But you can also define your own interfaces.

This is the general syntax for defining an interface. It's very similar to how a class is defined, but the Java keyword 'interface' is used.

In this syntax, the square brackets around the word `public` indicate that the `public` keyword is optional. If the `public` keyword is used then the interface can be imported into any package. And in practice, most interfaces will be `public` since this is the only way to share interface components.

If the `public` keyword is not used, the interface will only be visible to components within its own package.

The rules for naming interfaces are the same as those for naming Java classes...and, by convention, interface names begin with an upper case letter.


3

### Restrictions on Interface Members

An interface may only contain abstract methods or constants

The methods of an interface are implicitly public

```
public interface Drawable
{
    void draw();
}
```



There are two restrictions on the members of an interface:

First, an interface may only contain abstract methods or constants.

And, second, the methods of an interface are implicitly public.

Here's an example of defining an interface .

The interface name is Drawable, and in this case the interface contains an abstract method called draw.

Note that I didn't use the public visibility indicator or the abstract keyword in this declaration. I could have, but because the methods of an interface are implicitly public and abstract...the convention is not to use these keywords when defining interface members.

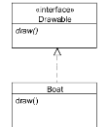
4

### Using an Interface

The keyword implements is used to specify that a class uses an interface

```
class Boat implements Drawable
{
    ...
    public void draw()
    {
        ...
    }
}
```

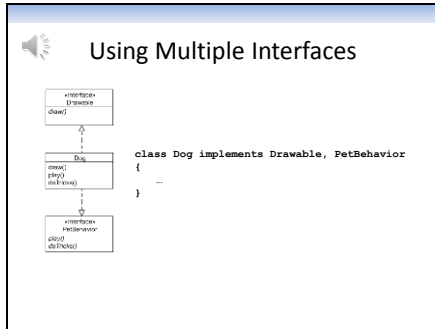
The Boat class must provide a code definition to make draw() a concrete method



A class indicates it is using an interface by including the keyword "implements" in its class definition, as you see here.

Since draw() is an abstract method, the Boat class must provide...or implement...the code for the draw() method to make it a concrete method.

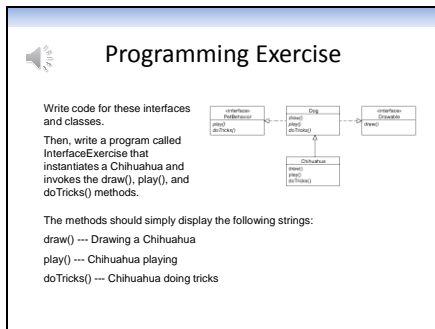
5



It's possible for a class to implement several interfaces as indicated here.

When this is done, each interface name is simply separated by a comma.

6

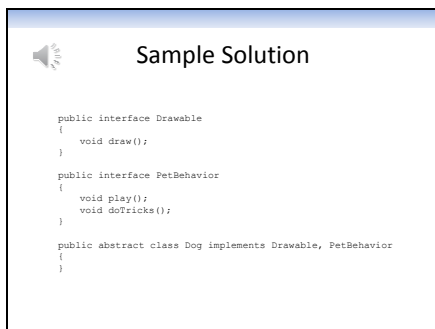


I'd like you to do a programming exercise involving defining and using interfaces.

Write code for the four components you see here. Then, write a program that instantiates a Chihuahua and invokes the 3 methods.

Please pause this lecture now...and resume it once you have your program working.

7



Here's a sample solution to the programming exercise.

Since Dog implements the two interfaces and doesn't do anything else, it is an abstract class and must be labeled as such.

The concrete Chihuahua class will be responsible for providing concrete method definitions for the interface methods.

8

### Sample Solution

```
public class Chihuahua
{
    public void draw()
    {
        System.out.println( "Drawing a Chihuahua" );
    }

    public void play()
    {
        System.out.println( "Chihuahua is playing" );
    }

    public void doTricks()
    {
        System.out.println( "Chihuahua doing tricks" );
    }
}
```

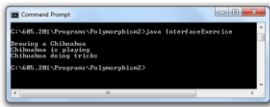
Here's the code for the Chihuahua class...all it really needs to do is provide method definitions.

9

### Sample Solution

```
public class InterfaceExercise
{
    public static void main( String [] args )
    {
        Chihuahua aDog = new Chihuahua();

        System.out.println();
        aDog.draw();
        aDog.play();
        aDog.doTricks();
    }
}
```



And...here's the program to instantiate a Chihuahua and invoke its methods.



And here's the program output.

If your solution doesn't match with this one, please check your program and make any changes that are required.

10

### Extending Interfaces

```
public interface MoreDrawable extends Drawable
{
    void rotate();
    void resize();
}
```

Interfaces can be extended just like Java classes can.

In this example, the interface MoreDrawable is extending the Drawable interface.

Note that since all the methods of an interface are public, MoreDrawable inherits all the method declarations in Drawable.

Note also that MoreDrawable adds a few more method declarations.