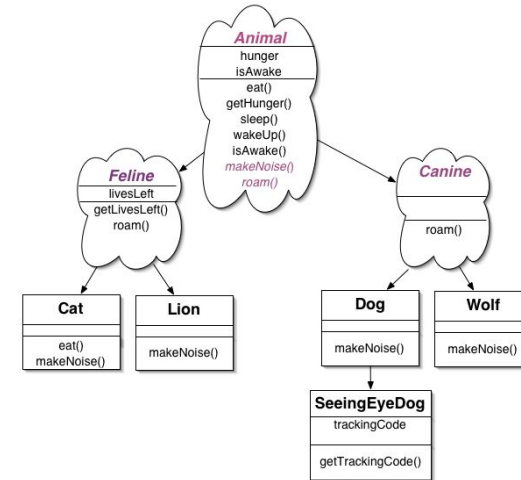


Introduction to Programming

with Java, for Beginners

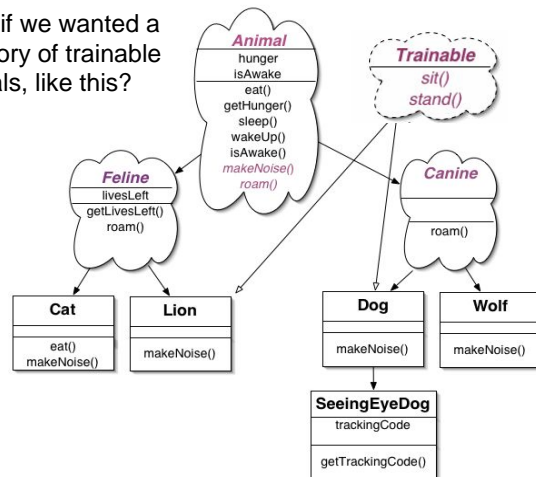
Interfaces



ESE112

1

What if we wanted a category of trainable animals, like this?



ESE112

2

Creating an Interface

```
public interface Trainable{
    public void sit();
    public void stand();
}
```

- An interface is like a class except:
 - The keyword **interface** is used instead of **class**
 - All of its methods are body-less
 - It has no instance variables
 - It can not be instantiated (Illegal: new Trainable())
- An interface is like a contract, protocol, role, or point of view
- Code written for an interface type works with **any object** whose class **implements** it
 - It can assume that all of the subtypes have the methods listed in the interface (e.g. sit and stand)

ESE112

3

A Class implements an Interface

```
public class Dog implements Trainable{
    public void sit(){ // code for sit method }

    public void stand(){ // code for stand method }
}
```

A class that implements an interface

- Must provide concrete methods for each interface method
- May have additional methods
- May implement multiple interfaces

```
public class Dog implements Trainable, Comparable{
    // code
}
```

ESE112

4

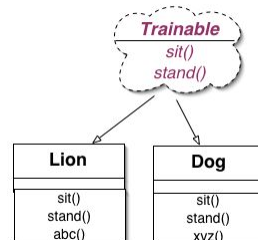
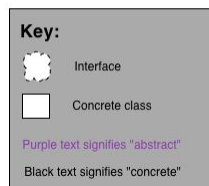
Abstract class vs. Interface

- When a partial implementation is feasible, abstract classes make sense as they can provide some functionality with the methods
- Pure abstract classes (with all abstract methods) in Java are functionally equivalent to an interface, but restricted to single inheritance
- Java will allow you to implement more than one interface
 - Multiple interface inheritance allows an object to inherit many different method signatures with the caveat that the inheriting object must implement those inherited methods.
- You can use access modifiers (e.g. protected) in an abstract class though. Interfaces are always public.

ESE112

5

Example: Trainable Interface



```
> Lion lion = new Lion();
> Dog dog = new Dog();
> Trainable beast;
> beast = lion; //beast is of type Trainable pointing to lion object
> beast.sit(); //calls lion's sit method
> beast = dog;
> beast.sit(); //calls dog's sit method
```

ESE112

6

Examples of Interfaces

- Interfaces are typically "lightweight", with only a few method headers

```
public interface Displayable{
    public void display();
}

public interface XYZProtocol {
    public void connect();
    public void disconnect();
    public int receive();
}
```

ESE112

7

instanceof Operator with Interfaces

- Usage: *Ivalue instanceof T*
- The expression is true if *Ivalue* has type T or is a subtype of T.
- Example:

```
public class Rectangle implements Displayable { .. }
public class Square extends Rectangle { .. }
```

```
> Rectangle r = new Rectangle(2,3);
> r instanceof Rectangle
true
> r instanceof Displayable
true
> r instanceof Square
false
> r = new Square(10);
> r instanceof Square
true

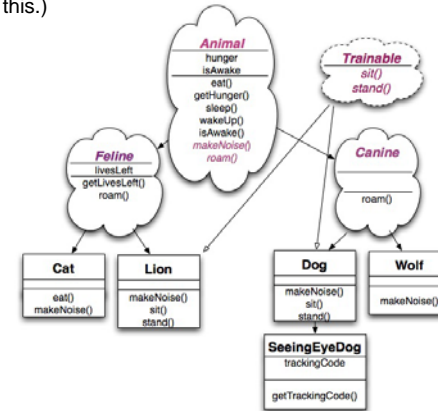
> Square s = new Square(8);
> s instanceof Square
true
> s instanceof Rectangle
true
> s instanceof Displayable
true
```

ESE112

8

A Design Problem

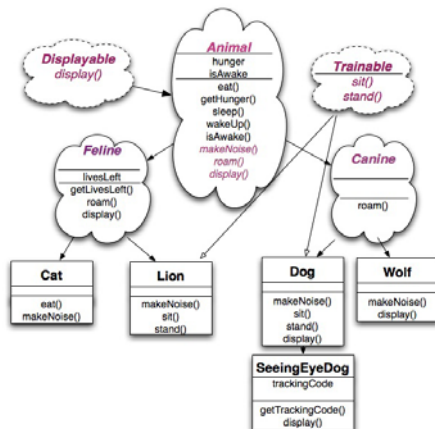
- What if we want to write graphics code for all the animals, and want to require that they all have a display(..) method? (There are many ways to accomplish this.)



ESE112

9

One of Many Solutions



ESE112

10

Extending and Implementing

- Class can only extend one other class
- But can implement multiple interfaces
- The exact order must be used as shown below in declaring a class below:

```
public class SubclassName extends SuperclassName implements Interface1,
Interface2 {
```

```
}
```

- Otherwise there is compile error

ESE112

11

Summary

	Regular Class	Abstract Class	Interface
Methods	all concrete	concrete and/or abstract	all abstract
May have instance variables	yes	yes	no
May be instantiated	yes	no	no