Object Oriented Programming

Week 5 Part 2
Java Interfaces

Lecture

- Animals example interfaces
- Implementing and Using Interfaces
- Multiple Interfaces
- Hierarchies of Interfaces
- When to use an interface or a superclass

Examples

- We will extend out Animals model
  - *Domesticated* animals are animal that are bred by humans
  - *Pets* are animals that love and are loved by humans
  - *Wild* animals fear humans
  - *Tame* animals do not fear humans
  - *Feral* animals are domesticated animals that have gone wild.

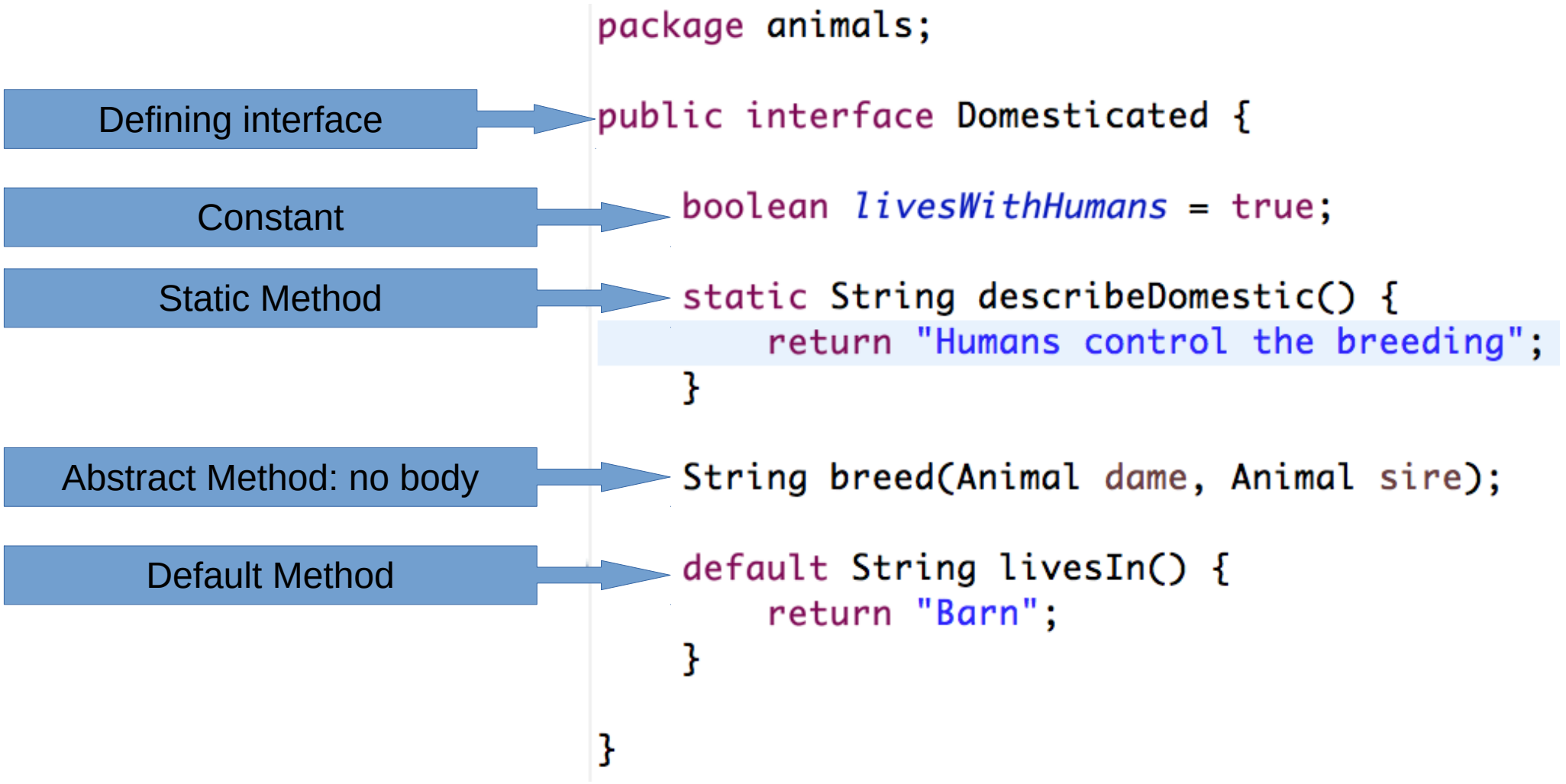
Defining Interfaces

- Interfaces may contain
  - *Abstract methods*: methods that are terminated by a ';' like C declarations.
  - *Default methods*: methods that may contain bodies and be overridden
  - *Static methods*: methods associated with the interface rather than the object
  - *Constants*: fields whose values never changes

Syntax for defining Interfaces

- Methods are automatically public, so you need not include “public”
  - The interface defines how the object is used from the outside
- Methods with bodies mush be declared using the either the “default” keyword or the “static”.
  - The interface is *implemented* in the object
  - The interface is only the contract.
- Fields are automatically declare “public static final”

Example Interface: Domesticated




```
package animals;

public interface Domesticated {

    boolean livesWithHumans = true;

    static String describeDomestic() {
        return "Humans control the breeding";
    }

    String breed(Animal dame, Animal sire);

    default String livesIn() {
        return "Barn";
    }

}
```

Try It

- Create an interface for shapes called sized that has a constant called Pi and an abstract method that will calculate the area of a shape

Implementing Interfaces

- Only classes can implement interfaces
  - Objects cannot be created from interfaces, only from classes that implement interfaces
- A class implements an interface by defining the the abstract method
  - The class defines the action the method performs
  - Because the interface is just the contract
- A class may override a default method
- The class may not redefine a static method or a field.

Example Implementation: Cow

Cow extends Mammal, the superclass

Class “Cow” implements Domesticated

```
public class Cow extends Mammal implements Domesticated {

    private static final String offspring = "Calf";
    private int yearsOld;

    Cow(String food) {
        super(food);
    }

    Cow(String food, int age) {
        super(food);
        this.yearsOld = age;
    }

    public String getOffspring() {
        return offspring;
    }

    @Override
    public Animal breed(Animal dame, Animal sire) {
        return new Cow("Hay", 0);
    }
```

Overrides breed(): signature must match

Object created from Cow, not Interface

Example Implementation: Dog

"Dog" is a Domesticated Wolf

Default dogs are mixed breed

Dogs may have breeds

Need to cast parameters to "Dog"

Pure bred if mother and father same breed

Otherwise Mixed

```
public class Dog extends Wolf implements Domesticated {

    private String breed = "Mixed";

    public Dog() {
        super();
        this.breed = "Mixed";
    }

    public Dog(String breed) {
        super();
        this.breed = breed;
    }

    @Override
    public Animal breed(Animal dame, Animal sire) {
        Dog mother = (Dog)dame;
        Dog father = (Dog) sire;
        if (father.getBreed() == mother.getBreed()) {
            return new Dog(mother.getBreed());
        } else {
            return new Dog();
        }
    }

    public String getBreed() {
        return breed;
    }
```

Try It

- Created a class called sizedCircle that implements the interface sized.

One Superclass; Multiple Interfaces

- Java allows only one superclass
  - Avoid conflict between methods and fields inherited from two superclasses
- Java allows multiple interfaces
  - Reintroduces conflicts
    - If two interfaces define methods with the same signature; both will refer to the same implementation
  - Conflicts are limited
    - No conflicts in fields

Example: Pet Dogs

- Pets can be implemented as interfaces
  - A pet can giveLove and getLove
    - Implement as methods in the interface

Example Interface Definition: Pet

Public Interface Pet

Declares two methods: givesLove(), getsLove()

```
package animals;

public interface Pet {

    String givesLove();
    String getsLove();

}
```

Example Interface Use: PetDog

Implements both Domesticated and Pet

Defined for Wolf; not needed for Dog

Added from Pet

Added from Pet

Added from Domesticated

```
package animals;

public class PetDog extends Wolf implements Domesticated, Pet {

    /*
     * Dogs are not born into packs
    public PetDog(String food, Pack p) {
        super(food, p);
    }
    */

    private String breed = "Mixed";

    public PetDog() {
        super();
        this.breed = "Mixed";
    }

    public PetDog(String breed) {
        super();
        this.breed = breed;
    }

    public PetDog(String food, String breed) {
        super(food);
        this.breed = breed;
    }

    @Override
    public String givesLove() {
        return "Lick face";
    }

    @Override
    public String getsLove() {

        return "Pat head";
    }

    @Override
    public Animal breed(Animal dame, Animal sire) {
        // TODO Auto-generated method stub
        return null;
    }
}
```

Hierarchies of Interfaces

- Interfaces may form hierarchies, like classes
- For example, a Feral animal is a Domesticated animal that is wild
  - E.g. Cats in Old Delhi were bred by humans, but also fear humans.

Example Interface Hierarchy: Feral

Wild interface: fears humans

```
package animals;

public interface Wild {

    String attitude = "Fears Humans";

}
```

Feral interface: Domesticated and Wild

```
package animals;

public interface Feral extends Domesticated, Tame {

}
```

Using Hierarchical Interface: Feral

Inherited from Dog,

Feral Dogs are Mix Breed,

```
package animals;

public class FeralDog extends Dog implements Feral {

    public FeralDog() {
        super();
    }
/*
    public FeralDog(String breed) {
        super(breed);
        // TODO Auto-generated constructor stub
    }
*/
}
```

Since Dog already implements Domesticated, we really only needed to add Wild to make it Feral. By creating the Feral interface we define a particular term we can use in our programs.

Using Java Interfaces

When superclass; when interface

- A class can implement multiple interface, but they are more restricted the superclasses
  - Still there is a lot of overlap
- Use an superclass to indicate that the subclass is a kind of the superclass
- Use interfaces to indicate that the interface operations are additional attributes

Animals Example

- The distinction is clearer in the example than it will be in real life
  - Animals inherit in the real world through DNA
  - Attributes such as Domesticated, Wild, and Feral are attributes added by humans
- Read Object Oriented Design will require careful thought about the intended means of the interface or type