Demo 3

Animal y PerroChiuaua
Inheritance usage: polymorphism

But with polymorphism, the reference and the object can be different.

```
Animal myDog = new Dog();
```

These two are NOT the same type. The reference variable type is declared as Animal, but the object is created as new Dog().
Why I want it??

```java
Animal[] animals = new Animal[5];
animals[0] = new Dog();
animals[1] = new Cat();
animals[2] = new Wolf();
animals[3] = new Hippo();
animals[4] = new Lion();
```

for (int i = 0; i < animals.length; i++) {
  animals[i].eat();
  animals[i].roam();
}

Cast!!

Dog dog = (Dog) animals[0];
Abstract classes

• Some classes should not be instantiated.
• They have no use unless extended.
• Example: Animal, Canine, Activity

```java
abstract public class Canine extends Animal {
    public void roam() {
    }
}
```
Abstract vs Concrete
Abstract methods

- **Abstract class** means class has to be extended
- So **Abstract method** means method must be overridden.

```
public abstract void eat();
```

ABSTRACT methods need to be in ABSTRACT classes!!!
Interfaces

- Problem, what if we want pet behaviours for animals? E.g. `beFriendly()`, `play()`
Interfaces

To DEFINE an interface:

```java
public interface Pet {...}
```

Use the keyword “interface” instead of “class”.

To IMPLEMENT an interface:

```java
public class Dog extends Canine implements Pet {...}
```

Use the keyword “implements” followed by the interface name. Note that when you implement an interface you still get to extend a class.
Demo 4

pet
Source

- This presentation is based on the content of the book:

*Head First Java*

by Kathy Sierra and Bert Bates
Weekend Homework

• Read chapters 2, 3, 4, 6, 7, (8, 9) from *Head First Java* to understand Objects and Classes
• Read chapter 6 from *Learning Android* to understand the project so far
• Do the java calculator with Calculator class, and Operation class