Accelerating Information Technology Innovation

http://aiti.mit.edu

Cali, Colombia
Summer 2013
Introduction to OOP
Classes

ShoppingCart
- cartContents
- addToCart()
- removeFromCart()
- checkOut()

Button
- label
- color
- setColor()
- setLabel()
- dePress()
- unDepress()

Alarm
- alarmTime
- alarmMode
- setAlarmTime()
- getAlarmTime()
- isAlarmSet()
- snooze()

Instance variables
(state)

methods
(behavior)

Song
- title
- artist
- setTitle()
- setArtist()
- play()
Classes

class Dog {
    int size;
    String breed;
    String name;
    
    void bark() {
        System.out.println("Ruff! Ruff!");
    }
}

instance variables

a method

<table>
<thead>
<tr>
<th>DOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
</tr>
<tr>
<td>breed</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>bark()</td>
</tr>
</tbody>
</table>
Classes vs Objects

- Class is the blueprint
- Object is the realization

```java
// make a new object
Dog d = new Dog();
```
Demo 1

Gato
References

Dog d = new Dog();
d.bark();

think of this
like this

Dog myDog = new Dog();

1. Declare a reference variable
2. Create an object
3. Link the object and the reference

myDog
Dog

Dog object

the reference
Dog object

myDog
Dog

Dog object
And for this reason...

String a = new String("Woow");
String b = new String("Woow");
String sa = a;

boolean b = a==b;
boolean b = a==sa;
So with arrays

1. Declare a Dog array variable
   ```java
   Dog[] pets;
   ```

2. Create a new Dog array with a length of 7, and assign it to the previously-declared Dog[] variable pets
   ```java
   pets = new Dog[7];
   ```

   **What's missing?**
   Dogs! We have an array of Dog references, but no actual Dog objects!

3. Create new Dog objects, and assign them to the array elements.
   Remember, elements in a Dog array are just Dog reference variables. We still need Dogs!
   ```java
   pets[0] = new Dog();
   pets[1] = new Dog();
   ```
Constructors

- Methods I want to execute when the object is created
- Usually initialize instance variables
- As many constructors as you want
- Declaration:
  - Public <ClassName> (<Arguments>){}
Accessors

- Instance variables are very important!
- Protect them with *private*
- Access them through *get* (to consult the value) or *set* (to change it)
- *Private vs Public* (be careful, public *by default*)
Demo 2

Perro
Inheritance

Class FriedEggMan extends Superhero{}`
Inheritance

• I can use the father methods and instance variables or create new ones
Rules of the inheritance contract: overriding methods

1. Arguments must be the same and return types compatibles
2. The method can not be less accessible