1. Log in
2. Today We Begin Unit 5
   - Inheritance (superclass & sublass)
   - Polymorphism
   - Abstract Classes
   - Interfaces

Oct 16-7:52 AM

Example of Inheritance ...

Animal Class

* Instance Variables
  - living - Is the animal alive? (boolean: true/false)
  - awake - Is the animal awake? (boolean: true/false)

Nov 11-3:13 PM

Example of Inheritance ...

Animal Class

* Instance Variables
  - living - Is the animal alive? (boolean: true/false)
  - awake - Is the animal awake? (boolean: true/false)
* Constructor Method - Living animal but not awake

Nov 11-3:13 PM

Example of Inheritance ...

Animal Class

* Instance Variables
  - living - Is the animal alive? (boolean: true/false)
  - awake - Is the animal awake? (boolean: true/false)
* Constructor Method - Living animal but not awake
* Other Methods
  - wakeUp( )
  - goToSleep( )
  - getSleepStatus( )
  - death( )
  - getLivingStatus( )

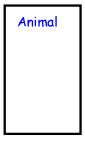
Nov 11-3:13 PM

It might look like this ...

```
public class Animal {
   boolean living, awake;
   public Animal( ) {
      living=true;
      awake=false;
   }
   public void wakeUp( ) {
      awake=true;
   }
   public void goToSleep( ) {
      awake=false;
   }
   public void getSleepStatus( ) {
      if(this.awake==true)
         System.out.println("This animal is awake!");
      else
         System.out.println("This animal is asleep. Shhhh....");
   }
   public void death( ) {
      living=false;
   }
   public boolean getLivingStatus( ) {
      return living;
   }
}
```

Nov 11-3:13 PM

Inheritance ...



*** Once upon a time, there was an animal class... ***

Nov 4-7:21 AM

Build a new class to be the main class: animalMain

Build and play with some animals!

```
public class animalMain {
   public static void main(String[] args) {

      Animal a1 = new Animal( );
      System.out.println("Animal 1 is alive ... "+a1.living);
      System.out.println("Animal 1 is awake ... "+a1.awake);

      a1.wakeUp( );
      System.out.println("Animal 1 is alive ... "+a1.living);
      System.out.println("Animal 1 is awake ... "+a1.awake);

      a1.death( );
      System.out.println("Animal 1 is alive ... "+a1.living);
      System.out.println("Animal 1 is awake ... "+a1.awake);
   }
}
```

Nov 11-3:28 PM

Let's say we want to build a Cat ...

```
public class Cat {
   boolean meowing;
   public Cat( ){
      meowing=false;
   }
   public void makeMeow( ){
      meowing=true;
   }
   public void stopMeow( ){
      meowing=false;
   }
   public boolean getMeowStatus( ){
      return meowing;
   }
}
```

Cat Class creates cat objects!

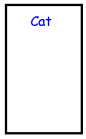
Every cat will have ...
- boolean meowing

We can make cats ...
- meow
- stop meowing

We can check to see ...
- if a cat is meowing

Nov 11-3:41 PM

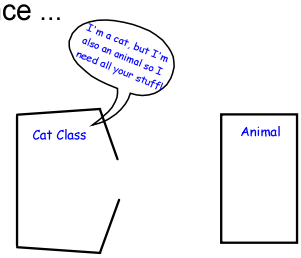
Inheritance ...



*** Once upon a time, there was a cat class... ***

Nov 4-7:21 AM

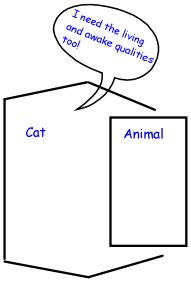
Inheritance ...



*** Let the "inheritance" begin ***

Nov 4-7:21 AM

Inheritance ...

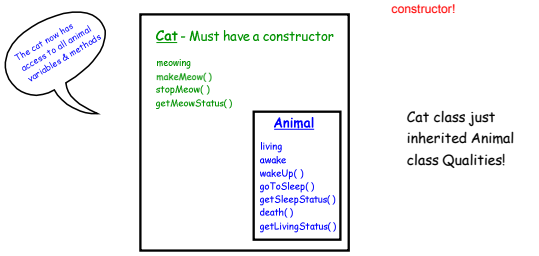


*** Cat Class is about to inherit the Animal Class ***

Nov 4-7:21 AM

Inheritance ...

NOTE: Constructors are NEVER inherited! Be sure your subclass has a constructor!



Cat class just inherited Animal class Qualities!

*** The Cat class still has its old code but now can also use the Animal code ***

*** If you leave out a constructor in Cat, it will default to Animal constructor ***

Nov 4-7:21 AM

General rule to create inheritance ...

```
public class nameOfSuperclass {                    //superclass="original" or smaller

    /* Code for the Superclass */

}

public class nameOfTheSubclass extends nameOfSuperclass {

    /* Code for the Subclass */

}
```

Nov 11-3:41 PM

Here's how we "inherit" ...

Remember ... 2 different classes
Cat & Animal

```
public class Cat extends Animal {
    boolean meowing;
    public Cat( ){
        meowing=false;
    }
    public void makeMeow( ){
        meowing=true;
    }
    public void stopMeow( ){
        meowing=false;
    }
    public boolean getMeowStatus( ){
        return meowing;
    }
}
```

Cat Class creates cat objects!

At the same time it inherits all Animal characteristics.

Every cat wants to have ...

boolean living
boolean awake
AND
boolean meowing

Nov 11-3:41 PM

Use of the word "super": Scenario #1 ...

```
public class Cat extends Animal{
    boolean meowing;
    public Cat( ){
        super();
        meowing=false;
    }
    public void makeMeow( ){
        meowing=true;
    }
    public void stopMeow( ){
        meowing=false;
        super.goToSleep( );
    }
    public boolean getMeowStatus( ){
        return meowing;
    }
}
```

The Cat class will inherit the instance variables of the superclass (Animal) and will initialize them EXACTLY as the Animal class does!

*** Note ***

super( ) MUST be the first line in the instance variable list if it is going to be used (if you are taking instance variables from the animal class.

Nov 11-3:41 PM

Use of the word "super": Scenario #1 ...

```
public class Cat extends Animal{
    boolean meowing;
    public Cat( ){
        super();
        meowing=false;
    }
    public void makeMeow( ){
        meowing=true;
    }
    public void stopMeow( ){
        meowing=false;
        super.goToSleep( );
    }
    public boolean getMeowStatus( ){
        return meowing;
    }
}
```

Use Animal variables

Include meowing as an instance variable along with all animal variables.

Remember:

In this "stage", we are defining instance variables

Nov 11-3:41 PM

Use of the word "super": Scenario #2 ...

```
public class Cat extends Animal{
    boolean meowing;
    public Cat( ){
        super();
        meowing=false;
    }
    public void makeMeow( ){
        meowing=true;
    }
    public void stopMeow( ){
        meowing=false;
        super.goToSleep( );
    }
    public boolean getMeowStatus( ){
        return meowing;
    }
}
```

The Cat class will inherit the instance variables of the superclass (Animal)

Run a method in the superclass ... cat stops meowing, then make them sleep.

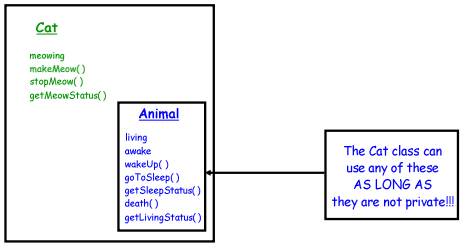
Nov 11-3:41 PM

Now try testing some Cats in the catMain Class ...

```
Cat a4 = new Cat();
System.out.println("Animal 4 is alive ..."+a4.living);
System.out.println("Animal 4 is awake ..."+a4.awake);
System.out.println("Animal 4 is meowing ..."+a4.meowing);
System.out.println("");

a4.makeMeow();
a4.wakeUp();
System.out.println("Animal 4 is alive ..."+a4.living);
System.out.println("Animal 4 is awake ..."+a4.awake);
System.out.println("Animal 4 is meowing ..."+a4.meowing);
System.out.println("");

a4.stopMeow();
System.out.println("Animal 4 is alive ..."+a4.living);
System.out.println("Animal 4 is awake ..."+a4.awake);
System.out.println("Animal 4 is meowing ..."+a4.meowing);
```

Nov 11-3:41 PM

It is at this point where 'public' and 'private' need to be known...



Nov 11-3:41 PM

Revisit a "revised" Animal Class ...

Public - 'living' instance variable is public ... open to sharing!

```
public class Animal {
   public boolean living;
   private boolean awake;

   public Animal( ) {
      living=true;
      awake=false;
   }
   public void wakeUp( ) {
      awake=true;
   }
   public void goToSleep( ) {
      awake=false;
   }

   ...

   }
}
```

Nov 11-3:41 PM

Revisit a "revised" Animal Class ...

Public - Can be inherited by the Cat class
Private - CAN NOT be inherited

```
public class Animal {
   public boolean living;
   private boolean awake;

   public Animal( ) {
      living=true;
      awake=false;
   }
   public void wakeUp( ) {
      awake=true;
   }
   public void goToSleep( ) {
      awake=false;
   }

   ...

   }
}
```

Nov 11-3:41 PM

Revisit a "revised" Animal Class ...

Public - Can be inherited by the Cat class
Private - CAN NOT be inherited

NOT WHAT WE WANT

We probably want to keep track of whether our cat is awake or not, so this should be a public variable!

```
public class Animal {
   public boolean living;
   private boolean awake;

   public Animal( ) {
      living=true;
      awake=false;
   }
   public void wakeUp( ) {
      awake=true;
   }
   public void goToSleep( ) {
      awake=false;
   }

   ...

   }
}
```

Nov 11-3:41 PM

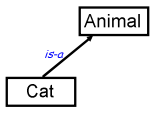
The same rule applies to methods ...

Public - Can be inherited by the Cat class
Public - Can be inherited by the Cat class
Public - Can be inherited by the Cat class

```
public class Animal {
   public boolean living;
   private boolean awake;

   public Animal( ) {
      living=true;
      awake=false;
   }
   public void wakeUp( ) {
      awake=true;
   }
   public void goToSleep( ) {
      awake=false;
   }

   ...

   }
}
```

Nov 11-3:41 PM

Inheritance Hierarchy For Animal-Cat Scenario:

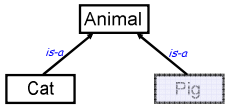


Inheritance Hierarchy

Is a flow-chart representation

Nov 11-8:51 AM

What if we want the following inheritance hierarchy?



Inheritance Hierarchy

*** We can almost copy and paste from Cat class ***

Nov 11-8:51 AM

The Pig class ...

Very similar, compare:

```
public class Pig extends Animal{
    public boolean oinking;

    public Pig(){
        super();
        oinking=false;
    }

    public void makeOink(){
        oinking=true;
    }

    public void stopOink(){
        oinking=false;
        super.goToSleep();
    }

    public boolean getOinkStatus(){
        return oinking;
    }
}
```

Nov 11-8:51 AM

Things to do ...

1. Complete Unit 5 WS 01 Inheritance Introduction
2. Work on Unit 5 WS02 Inheritance Practice

Oct 16-9:12 AM

Nov 12-2:49 PM