Today’s topics

Java Language
   Inheritance

Upcoming
   Electric Circuits
      (not in text)

Reading
   Great Ideas, Chapters 5
Inheritance in Everyday Life

- Inheritance kind of ideas are not strange in today’s world
- When you order a new car you:
  - Pick model
    - That implies certain standard features
  - Pick a color
  - Choose engine
    - Often have several choices
  - Choose other options
    - Often many packages to choose from
  - The code, as interpreted by the machine will provide meaning
- Whatever you choose, it will have lots in common with other cars
  - We would say these common things are inherited from the model
  - Many of the things you chose might have sub-options
  - Thus, there would be another level of inheritance
Inheritance in O. O. Programming

- Basic Ideas is *Don't Reinvent the Wheel!*
- Wherever possible, *build* on the work of others
- Reuse as much as possible
  - Modify only where necessary
  - Delete where not appropriate
- Vocabulary
  - Parent Class or Super-Class
  - Child Class or Sub-Class
  - Child inherits from parent class
Inheritance in Graphics

- Assume we want to write a simple drawing package
- A basic feature might be a shape
  - What functions might be associated with shape?
  - I.e., what does every shape have?
  - Also, what do most shapes have?

```
Every:                               Most:
Location                              Area
Size                                  Fill/NoFill
Color                                  Fill Color
Orientation
```
Shape Subclasses

• What are the obvious shapes?
  ➢ Oval
  ➢ Line
  ➢ Triangle
  ➢ Rectangle
  ➢ Polygon
  ➢ Pie
  ➢ Arc

• How do these mesh with some of the methods suggested?
  Location     Area
  Size         Fill/NoFill
  Color        Fill Color
  Orientation  

Subclasses of subclasses

• Which of our Specific shapes might have subclasses?
  ➢ Oval          circle
  ➢ Line
  ➢ Triangle      equilateral-triangle
  ➢ Rectangle     square
  ➢ Polygon       pentagon, hexagon, ...
  ➢ Pie
  ➢ Arc

• Still leaves many things to deal with
  ➢ Parallelogram
  ➢ Arbitrary polygons
  ➢ Various line shapes
  ➢ Use composites...
Inheritance in Java

- We say a subclass extends a parent class
- Remember:
  ```
  public class classname extends java.applet
  ```
  This means our class is a subclass of java.applet
- Some of the other classes we’ve used also invoked inheritance
  - Intfield inherits from TextField
  - DoubleField inherits from TextField
  - StringField inherits from TextField
  - All of these classes include the words
    ```
    extends TextField
    ```
- Both TextField and TextArea are subclasses of a class called a TextComponent
Inheritance in Java

- Let's think of our graphic/drawing problem
- If we had a class `shape`, then we would write
  
  ```java
  public class oval extends shape
  
  public class circle extends oval
  ```

  - In order to implement our `oval` class and
  - To make `circle` a subclass (special case of) `oval`

- So, if the class `shape` had the method `setColor`, then it
  - could be used by `oval` and
  - could also be used by `circle`
Inheritance in Java

- Sometimes, this inheritance doesn’t work quite that easily
- Take a method such as `getArea` which might be considered for the class `shape`
  - It is unlikely that one could come up with an area calculation that works for all shapes
  - This means we need to write specialized versions of `getArea` for each shape
  - This is called *overriding* a method
  - We simply write a new `getArea` in the subclass
- On the other hand, `getArea` for oval *will* work for circle
  - We might still override for efficiency reasons. Formula for circle is simpler.
Access Control

- Methods and data may have restricted access
- Use public, private, or protected to specify
- For methods:
  - public means: anyone can use
  - private means: can only use within class
  - protected means: only class and subclasses can use
- For data fields:
  - public means: anyone can access or modify
  - private means: can only access or modify within class
  - protected means: only class and subclasses access or modify
- Helps support Information Hiding
Polymorphism

- When you have inheritance hierarchy, can be more general
- Just as we can say it’s a Ford, for all models and variations, we can say something is a shape and then allow ovals, circles, etc.
- This ability to be more general is called polymorphism

- In Java everything, by default, inherits from the class object
  - Thus you have a hierarchy containing all classes
  - object is at the “peak”