More on Inheritance

- Last time we introduced the concept of inheritance, and creating sub-classes that added specialized functionality.

- When we need to add new functionality to a class, we have some options:
  1. Create a new class that replicates the old functionality as well as the new features.
  2. Add the new functionality (variables & methods) to the class itself.
  3. Create a new subclass that extends the original class functionality.
More on Inheritance

- Issues with *creating a new class* that replicates the old functionality as well as the new features:
  - Requires you to re-implement existing methods.
  - Introduces duplication of code that needs to be maintained in parallel.

More on Inheritance

- Potential issues with *updating the existing class*:
  - Changing the code in the existing class requires recompiling the class, and any packages or applications that use that class.
  - Adding the new functionality to the existing class means that every instance of the class has the new feature, even if it doesn’t need it.
  - It may require creating and maintaining multiple versions of the same methods, which can lead to complexity.
More on Inheritance

- **Benefits of creating a new sub class:**
  - Inheritance allows for the creation of a new class of objects that behave like some other objects without duplicating code from the original class.
  - The new class allows for using different instances of different classes in different applications.
  - Eliminates the chance of introducing new bugs into the already functional class.
  - Does not require modifying existing programs that can still use instances of the original class, that don’t need instances of the new class.
  - By taking advantage of polymorphism, we can interchange our objects in certain situations to provide additional flexibility.

Designing a SubClass

- **Questions to ask yourself:**
  - What new instance variable[s] do I need?
  - What new methods do I need to implement?
  - What existing methods do I need to override?

- **Remember:**
  - You can call the super class’ methods from the sub class using the super keyword.
  - You need to implement a constructor for each one inherited from the super class.
  - If you need to access private instance variables in the super class you must either
    1) change the variable to protected or
    2) write a protected accessor method to get the data.

(Try not to need to do this.)