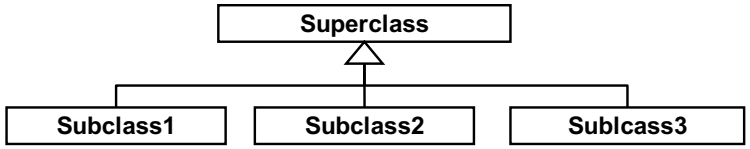
# Your Email Address

- Send your (preferred) email address to umasscs680@gmail.com ASAP.
  - I will use that address to email you lecture notes, announcements, etc.



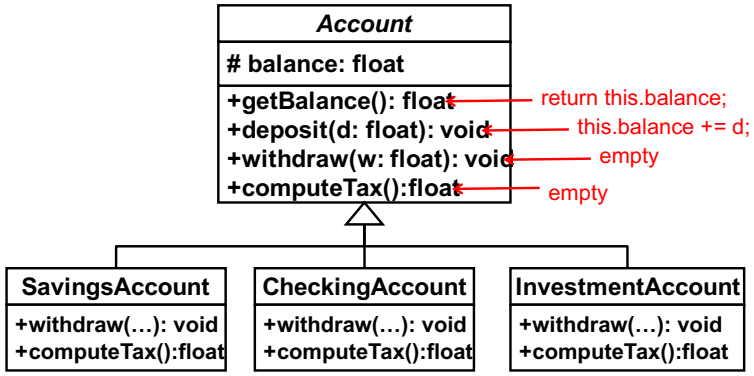
# Why Inheritance?




- Reusability
  - You can define common data fields and methods in a superclass and make them reusable in subclasses.
- Customizability and extensibility
  - You can customize method behaviors in different subclasses by overriding (re-defining) inherited methods.
  - You can add new data fields and methods in subclasses.

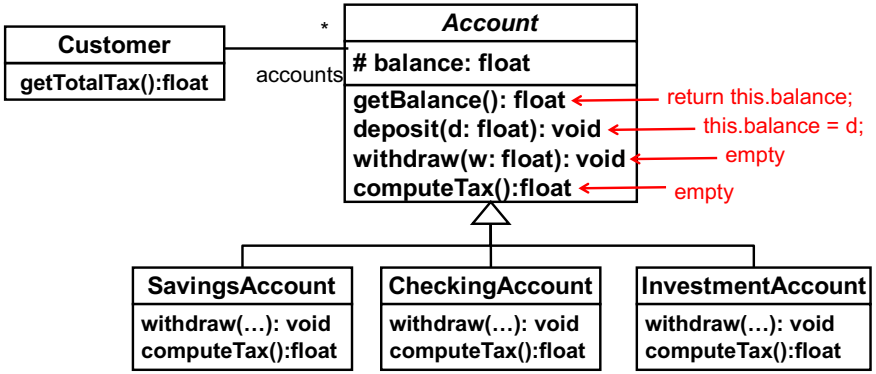


# An Example




- Subclasses can redefine (or override) inherited methods.
  - A savings account may allow a negative balance with some penalty charge.
  - A checking account may allow a negative balance if the customer's savings account maintains enough balance.
  - An investment account may not allow a negative balance.

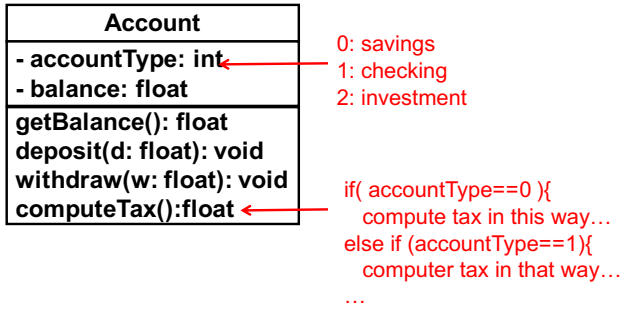




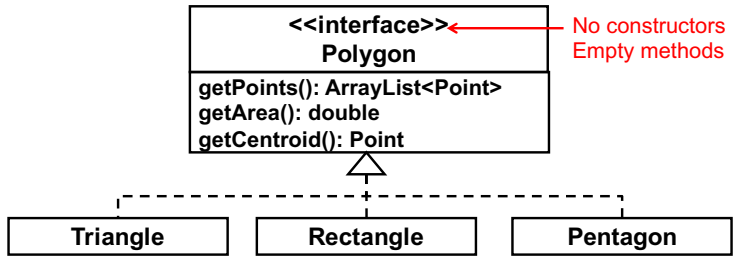

```
public float getTotalTax(){
    Iterator<Account> it = accounts.iterator();
    while( it.hasNext() )
        System.out.println( it.next().computeTax() ); }
```

- Polymorphism can effectively eliminate conditional statements.
  - Conditional statements are VERY typical sources of bugs.

# If Polymorphism is not available…






```
ArrayList<Polygon> p = new ArrayList<Polygon>();
p.add( new Triangle( new Point(0,0),
                     new Point(2,2),
                     new Point(1,3) ));
p.add( new Rectangle ( new Point(0,0)... ));
Iterator<Polygon> it = p.iterator();
while( it.hasNext() ){
  Polygon nextP = it.next();
  System.out.println( nextP.getPoints() );
  System.out.println( nextP.getArea() );
  System.out.println( nextP.getCentroid() ); }
```



# HW 2

- Write the Polygon interface and its two implementation classes: Triangle and Rectangle.
  - You can reuse Point in Java API or define your own.
- Implement getPoints() and getArea() in the two subclasses.
  - Use Heron's formula to compute a triangle's area.
    - The area of a triangle = Sqrt(s(s-a)(s-b)(s-c))
      - where s=(a+b+c)/2
      - a, b and c are the lengths of the triangle's sides.
- In the main() method, write test code that
  - makes two different triangles and two different rectangles,
  - contains those 4 polygons in a collection (e.g. ArrayList),
    - Use generics and an iterator
  - printouts each polygon's area.
- Keep the encapsulation principle in mind.
  - All data fields must be "private."
  - No setter methods are required.



# Note

- If you are not very familiar with class inheritance and polymorphism, you may want to implement Student and Account examples as well as HW2.

# Automatic Build

- Use Ant (http://ant.apache.org/) to compile/build all of your Java programs in every coding HW.
  - Learn how to use it, if you don't know that.
  - Turn in *.java and build.xml for every coding HW.
    - Turn in a ***single*** build script (build.xml) that
      - configures all settings (e.g., class paths and a directory to generate binary code),
      - compiles all source code from scratch,
      - generates binary code, and
      - runs compiled code
    - DO NOT include absolute paths in build.xml.
      - You can assume my OS configures a right Java API JAR file (in its env setting).
    - DO NOT turn in byte code (class files).
    - DO NOT use any other ways for configurations and compilation.
      - Setting class paths manually with a GUI (e.g., Eclipse)
      - Setting an output directory manually in a GUI
      - Clicking the "compile" button manually



  - I will simply type "ant" (on my shell) in the directory where your build.xml is located and see how your code works.
    - If the "ant" command fails, I will NOT grade your HW code.
  - Fully automate configuration and compilation process to
    - speed up your configuration/compilation process.
    - remove potential human-made errors in your configuration/compilation process.
    - Make it easier for other people (e.g., code reviewers, team mates) to understand your code/project.



- If you download Ant from http://ant.apache.org/ and use it on a shell,
  - Use the ANT_HOME and PATH environmental variables to specify the location of the "ant" command (e.g., ant.sh and ant.bat)
  - ANT_HOME
    - Reference the top directory of an Ant distribution
      - e.g. Set ~/code/ant/apache-ant-1.9.7 to ANT_HOME
      - e.g., Set ${ANT_HOME}/bin to PATH
- You can use Ant that is available in an IDE (e.g. Eclipse).
  - However, I will run your build.xml on a shell.



# Expected Directory Structure

- Make "src" sub directory under the top directory for a HW.
- Have build.xml generate "bin" sub directory under the top directory, generate all class files in there, and run your code by calling main()
- Place build.xml in the top directory.
- An example:
  - <top directory for a HW>
    build.xml
    src
      a.java
      b.java
    bin
      a.class
      b.class
- Submit me an archive file (in .zip, .rar, .tar.gz, .7z, etc.) that contains build.xml and the "src" sub directory.
  - Email it to umasscs680@gmail.com, OR
  - Place it somewhere online (e.g. at G Drive) and email a link to it.