

Your Email Address

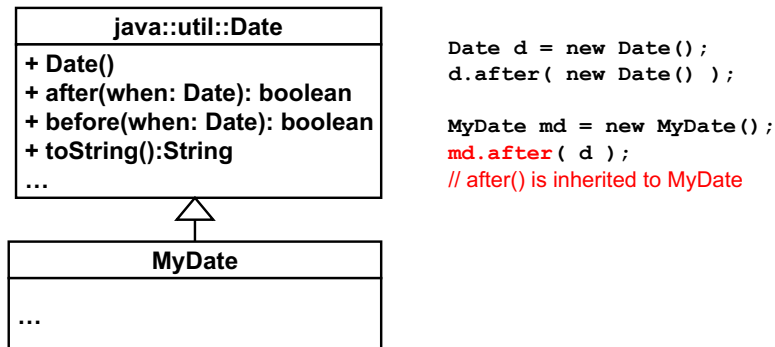
- Send your (preferred) email address to `umasscs680@gmail.com` ASAP.
 - I will use that address to email you lecture notes, announcements, etc.

Inheritance (Generalization)

1

2

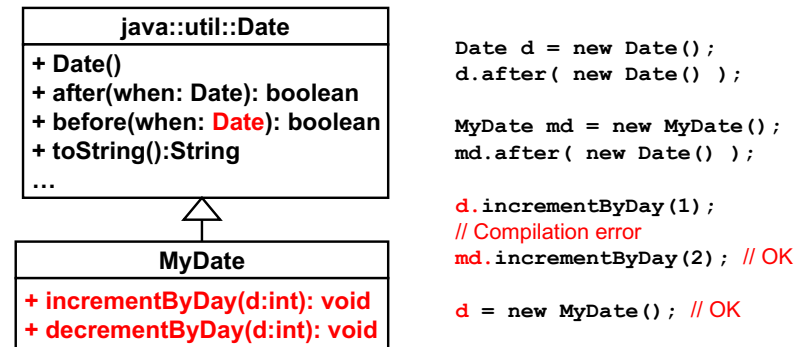
Inheritance



- Generalization-specialization relationship
 - a.k.a. “is-a” relationship; MyDate is a (kind of) Date.
- A subclass can *inherit* all public/protected data fields and methods from its base/super class.
 - Exception: Constructors are not inherited.

3

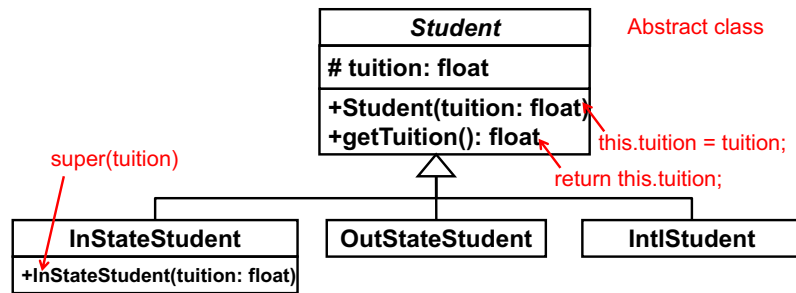
Inheritance (cont'd)



- A subclass can *extend* a base/super class by adding extra data fields and methods.
- An instance of a subclass can be assigned to a variable typed with the class's superclass.

4

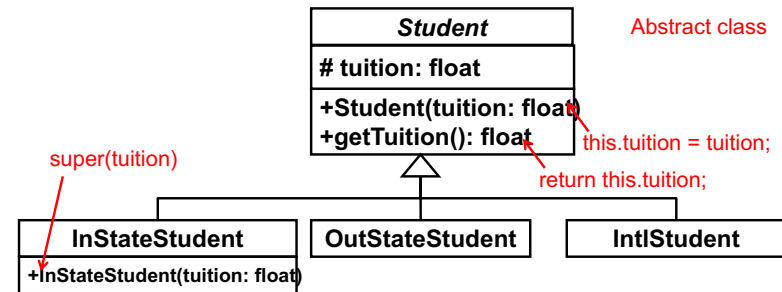
Exercise



- ```

ArrayList<Student> students = new ArrayList<Student>();
students.add(new OutStateStudent(2000));
students.add(new InStateStudent(1000));
students.add(new IntlStudent(3000));
Iterator<Student> it = students.iterator();
while(it.hasNext())
 System.out.println(it.next().getTuition());

```
- What are printed out in the standard output?



- ```

ArrayList<Student> students = new ArrayList<Student>();
students.add( new OutStateStudent(2000) );
students.add( new InStateStudent(1000) );
students.add( new IntlStudent(3000) );
Iterator<Student> it = students.iterator();
while( it.hasNext() )
    System.out.println( it.next().getTuition() );

```
- ```

2000
1000
3000

```

6

## Exercise

- Learn generics in Java (e.g., ArrayList) and understand how to use it.
- Learn how to use java.util.Iterator.
- This code runs.
 

```

ArrayList<Student> al = new ArrayList<Student>();
al.add(new OutStateStudent(2000));
System.out.println(al.get(0).getTuition()); → 2000

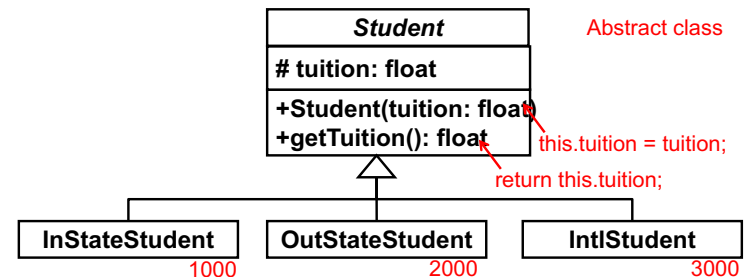
```
- This one doesn't due to a compilation error.
 

```

ArrayList al = new ArrayList();
al.add(new OutStateStudent(2000));
System.out.println(al.get(0).getTuition());

```
- Understand what the error is and why you encounter the error.

## Polymorphism



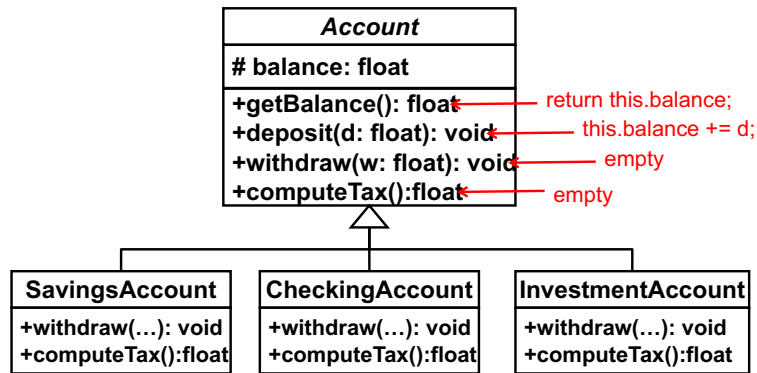
- ```

ArrayList<Student> students = new ArrayList<Student>();
students.add( new OutStateStudent(2000) );
students.add( new InStateStudent(1000) );
students.add( new IntlStudent(3000) );
Iterator<Student> it = students.iterator();
while( it.hasNext() )
    System.out.println( it.next().getTuition() );

```
- All slots in "students" (an array list) are typed as Student, which is an abstract class.
- Actual elements in "students" are instances of Student's subclasses.

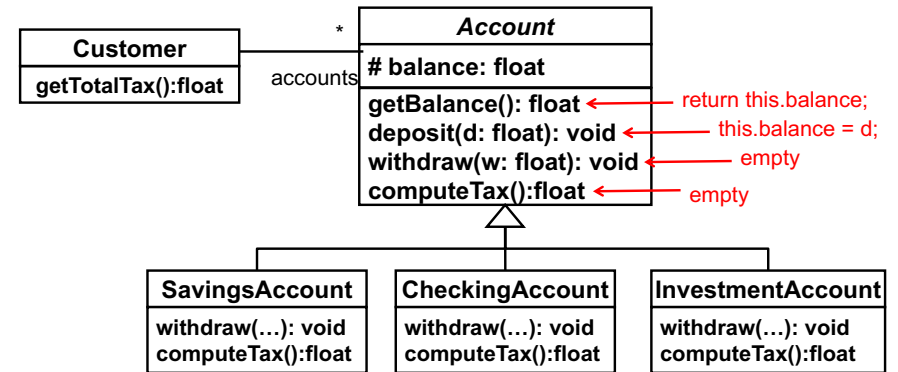
7

8



- 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.

9



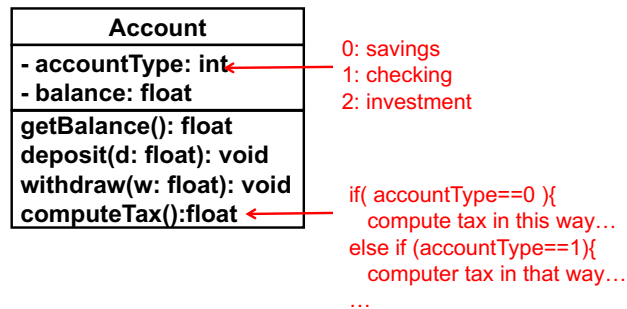
- ```

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 a VERY typical source of bugs.

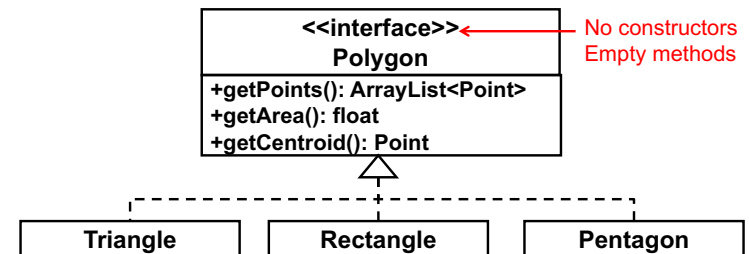
10

## If Polymorphism is not Available...



11

## Another Exercise



- ```

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() );
}

```

12

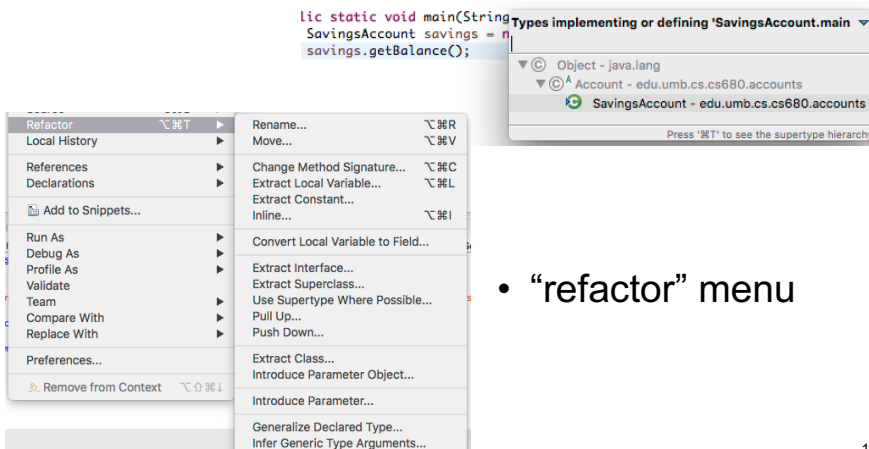
Note

- Write the Polygon interface and two 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 = $\text{Sqrt}(s(s-a)(s-b)(s-c))$
 - where $s=(a+b+c)/2$
 - a, b and c are the lengths of the triangle's sides.
- 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
 - Print outs each polygon's area.
- Keep the encapsulation principle in mind.
 - All data fields must be "private."
 - No setter methods are necessary.

13

An Eclipse Tip

- Ctrl+T to browse a class hierarchy.



- "refactor" menu

15

- If you are not very familiar with class inheritance and polymorphism, you may want to implement Student, Account and/or Polygon examples yourself.

14

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 a build script (e.g. build.xml).
 - Turn in a **single** build script (build.xml) that
 - configures all settings (e.g., class paths, a directory of source code, a directory to generate binary code),
 - compiles all source code from scratch,
 - generates binary code (*.class files), and
 - runs compiled code
- DO NOT include absolute paths in a build script.
 - You can assume my OS configures a right Java API (JDK/JRE) 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 paths manually with a GUI (e.g., Eclipse)
 - Setting an output directory manually with a GUI
 - Clicking the "compile" button manually

16

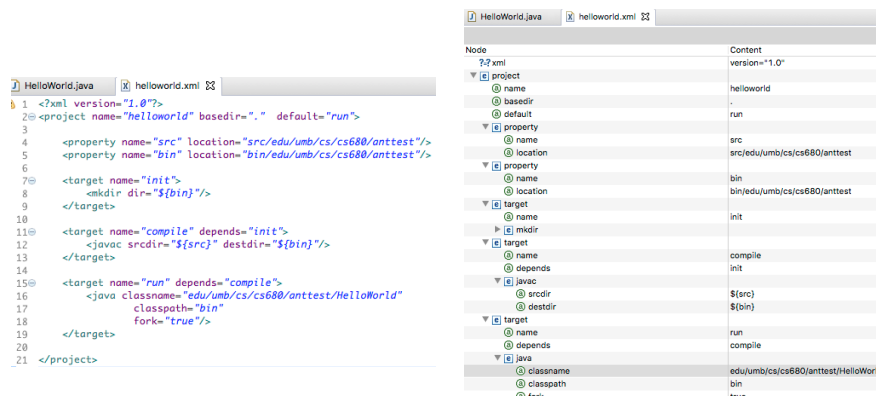
Using Ant on a Shell.

- I will simply type “ant” (on my shell) in the directory where your build.xml is located and see how your code works.
 - You can name your build script as you like.
 - No need to name it build.xml.
 - I will type: ant -f abc.xml
 - 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.

17

Ant in Eclipse

- You can use Ant that is available in your IDE (e.g. Eclipse).
 - However, I will run your build script on a shell.



- If you download Ant from <http://ant.apache.org/> and use it on a shell,
 - Use the ANT_HOME and PATH environment 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
 - c.f. <http://ant.apache.org/manual/>
- You can assume “ant” command works properly on my machine.

18

Expected Directory Structure

- Make “src” subdirectory under the top directory for a HW.
- Have build.xml generate “bin” subdirectory under the top directory, generate all binary 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
 - edu/umb/cs680/HelloWorld.java
 - edu/umb/cs680/Umass.java
 - bin
 - edu/umb/cs680/HelloWorld.class
 - edu/umb/cs680/Umass.class
- Submit me an archive file (in .rar, .tar.gz, .7z, etc.) that contains build.xml and the “src” sub directory.
 - Email it to me at umasscs680@gmail.com, OR
 - Place it somewhere online (e.g. at G Drive) and email me a link to it.

20

Notes

- Avoid .zip files.
 - Google often rejects .zip files or automatically removes .zip file attachments.
- Do not submit binary files (e.g., .class files, Jar files).