Thinking about Java's classes and objects.

1. Questions?

2. Previously, we learned that Java allows you to write static methods to perform standalone computations. Nevertheless, we also learned that all methods are declared as part of a class. This class serves as a wrapper for related methods. For example, Math is a class that hosts a number of static methods (sin, cos) as well as constant or final values (PI, E). You can get a profile of the Math class with:

   javap java.lang.Math

3. Methods that are not declared static are dynamic: they represent methods that appear and act on objects of the class.

4. One of the most important methods is named after the class, itself. This method is the constructor for the class. This is the equivalent of Python’s initializer. Unlike Python, Java allows the definition of multiple constructors that are distinguished by the types of their parameters, by their signatures. (This is because Java does not allow default values for parameters; think about this.)

5. Similarly, non-static constants and variables declared within the class are called instance variables. These variables appear only in the context of an object. They are responsible for holding the object’s state. These variables correspond, roughly to the slots declared in Python objects.

6. The constructor is responsible for initializing the instance variables.

7. The toString method provides a mechanism for constructing a string representation of an object. It’s the equivalent of __str__ in Python.

8. The equals method provides a method for comparing objects, like __eq__ in Python.

9. The hashCode method allows one to generate a integer that represents the object. If two objects are equal, they must generate the same hash code. If two objects are not equal, we hope they have different hash codes; this need not be the case. This is equivalent to __hash__ in Python.

10. Example: The Association class.

*