OOP and Exceptions

Class versus Object

- Class – A framework; the definition of the format of the "thing"
- Object – An instance of the "thing"

- Examples:
  - Blueprint / House
  - Outline / Essay
  - Cookie Cutter / Cookie

Important Reminder

Variables, in Java, are references, unless the variable is a primative

Object class

- In Java, all classes derive from a root class, called “Object.”
- Object defines a number of functions which, obviously, all classes derive due to inheritance rules
  - clone – create a copy of the object
  - toString – similar to repr/str in Python
  - finalize – similar to destructor in C++, but less useful
  - hashCode – similar to __hash__ in python

Constructors

- You can have as many as you'd like!
  - Constructors can "invoke" other constructors with "this" calls
  - Default parameters to functions aren't possible so "this()" is going to be very useful

Operator overloading…

- ..you can't… 'nuf said

Access Specifier (protection)

- public
- private
- protected
- (default)

(default)

- Java's "default" access specifier is used if you do not specify another access specifier.
- Default (your book calls it "Package Private") is needed because of Java's heavy use of classes!
- Classes are combined into "packages" and all member methods from all classes inside a package have access to the "default" members.
- (we will cover packages later in the semester)

Static versus Instance members

- Usually we want everything stored in the object, so variables that are "instance" variables are stored as we experienced in C++; inside the object.
  - These require instantiation of the object in order to use
  - We have one per object
- Sometimes, we'd like to "share" a variable across all of the objects in a class. If a variable is made "static" it is stored inside the class
  - These DO NOT require instantiation of the object!
  - We have one copy per class
  - They can be accessed using the name of the class (preferred) or the name of any object of the class (will generate a warning)
- Functions can also be "static." Static functions DO NOT require instantiation of the object.
  - Static functions can only access static variables!

Inheritance

- Java uses “extends” and only has public inheritance
- Java prohibits multiple inheritance
- Everything inherits from Object (the ultimate base class)

Derived calling base

- Constructors – Since the derived class can have only one base class, there is no need to "name" the Base class constructor you wish to call, it's "super."
- Base methods called from Derived class
  - Access specifiers still apply
  - No need to name the base class, again just use "super" i.e. super.method(param)
  - You cannot use super.super!

Abstract Classes

- Similar to C++
- An abstract class contains one or more abstract functions
- Abstract clases cannot be instantiated

Final class / final method

- (no, not our final class, this is the beginning of the semester!)
- A class marked as “final” cannot be used as a base class
- A final method cannot be overridden

Quick question

- Main exists inside of a class. How many objects of that class are instantiated?
- Follow up question – Can main ever access instance variables?

Exceptions – They're exceptional!

- Java is STRICT with exceptions!
- Exceptions must be known at compile time
- All checked exceptions must be caught
  - Unchecked exceptions include errors like IntegerDivideByZeroException ArrayIndexOutOfBoundsException and NullPointerException
- If your method may "throw" an exception, it must listed in the "throws" portion of the function signature.

Try / Throw / Catch

- Similar to C++
- For an object of a class to be thrown, it must extend Throwable (or derive from one that does, somewhere in its ancestry)
- An object is thrown and the object usually carries a "message" which can be retrieved with toString

Finally

- No, we're not done quite yet
- Finally will ALWAYS run even if there is no catch block for the data type thrown.
- Often used to "clean up" objects, but not in the memory management sense

Assertions

- “assert conditional: “conditional is not true!“
- Same as “throw AssertionError”