CSC 207 2014S: Extra Session, Week 1

Overview

- Your questions
- My answers
- Anything else of interest

Questions

How should we submit lab 1?

You have a Java file that has extra stuff in it. Copy the file into an email message and send it to me.

Tell me about packages.

Evidence suggests that people like to name their classes/files the same thing. (You’ll see a lot of `Math.java` or `Utils.java`). The package provides a way to distinguish them. `edu.grinnell.cs.rebelsky.Utils` is clearly different than `com.oracle.java.Utils`, even if both are from `Utils.java`.

Packages also provide an intermediate form of protection. You can indicate that the fields or methods of a class are accessible to the other classes in your package, but not to the outside world. (In fact, if you don’t specify a protection level, the default is package protection.)

Are abstract data types naturally polymorphic?

Yes, although we’ll want to do two variants of polymorphism.

Think about a list. We can have a heterogeneous "list of anything", which includes strings, integers, People, whatever. That’s one form of polymorphism. But there are times that we want homogenous lists. For example, if we’re summing the elements of a list, the better all be numbers.

We can achieve heterogenous lists by using the generic `Object` type. (Almost everything can be treated as an object.) We achieve homogenous lists by using Java "generic types", which we’ll cover in a few weeks.

Talk to me about how Java programmers think about variable declarations.

Usually, Java programmers try to limit the "scope" in which a variable is accessible. It should only be usable in the portion of the program where it is meaningful. Here’s a possible demonstration.

```
/**
 * A quick test of Scope in Java.
 */
public class Scope
{
  public static void
  main (String[] args)
      throws Exception
  {
    for (int i = 0; i < 10; i++)
      {
        // Even though we’re declaring this in the body of the loop,
        // the Java compiler only allocates memory for this variable once.
        // (Yes, it’s smart.)
        double x = i*2.0;
        System.out.println (x);
      } // for
    // The following instruction prevents compilation b/c i is
    // not in scope.
    // System.out.println (i);
    for (int i = 0; i < 10; i++)
      {
        // Note that the following instruction redeclares x, and even
        // makes it a different type.
        int x = 2 * i;
        System.out.println (x);
      } // for
  } // main (String[])
} // class Scope
```