COMP 1451 – Summary of Session 2 Concepts

- The second session is a discussion of working with library classes, and some practice.
- Interface vs implementation. The interface view of a class is a list of everything public, i.e. the class' constructors and public methods. Complete method signatures are provided, with comments that explain what the method is for. The implementation is the entire source code of the class, including everything that is private. When we use a library class we have access only to the interface view. The interface tells us what we need to know about a class in order to use it. The implementation is hidden from us.
- Packages. Java classes are organized into packages. To view a list of the packages, go to the website http://java.sun.com/j2se/1.5.0/docs/api/index.html. Look in the top left corner.
- Another meaning for "interface". If you click on a package you will see a list of Interfaces and a list of Classes. This use of "interface" is different from the one above. We will discuss this other use of "interface" when we get to chapter 10.
- `import` statement. To import a single class we specify the class' fully-qualified name, e.g. `import java.util.ArrayList;` To import an entire package we can use the `*` wildcard, e.g. `import java.util.*;`
- Documentation. Your classes should be thoroughly documented as if they were library classes, with descriptive comments for each constructor and method.
- Review of public vs private. Fields should always be private. Only methods that are meant to be used by other classes should be public. This is an important concept in OOP, the data of an object is hidden (encapsulated). See interface vs implementation.
- Maps. A map stores data in pairs: key and value. Key and value are both objects, and need not be the same object type. We worked with the Java class HashMap, and some of its methods: put(), get(), remove(), keySet(). The keySet() method returns a set of all the keys in the map.
- Sets: A set stores unique objects, no duplicates. The objects stored in a set are not guaranteed to be in any particular order. A Java class HashSet is mentioned in the chapter. A set can be processed in the same ways as an ArrayList, for example to access each element in the set in turn. A for-each loop works well for this.