Designing with Java Classes & Class Template (Version 0.9)
Prepared by Charles S. Tritt, Ph.D.
Last revised 4/11/05

Classes and Objects

Remember that *classes* are like blue prints that describe how to create *objects* of a particular type. Objects are physical or conceptual entities involved in the problem of interest. In object oriented programming, the "main program" often creates objects and supervises their interaction. Most of the "work" is done by the object methods.

Classes

Generally, create a class for each type of noun in the problem statement. For example, if you were designing software for an x-ray machine classes might include Patient, Image, WorkOrder, MachineSettings. Class definitions list and define the *data members* and *methods* (called member functions in C++) of the class.

Data Members

Each class should include enough data members to fully define the **state** of objects of the class. Extra data members can be included to reduce recalculation of values that rarely or never change, but only after careful consideration. In the x-ray example, Patient data members might include firstName, lastName, middleInitial, patientIDNumber, orderingPhysician, etc.

Methods

Each class should include one or more *constructors* that are used to construct **new** objects (instances) of the class. Classes should also include *mutators* that change objects states (often having names like setQuantity) and *interrogators* that return information about the states of objects (often having names like getQuantity). Several other general types of methods are also common.

Input and Output (I/O)

Input and output generally should be limited to specific user interface and file I/O classes. The I/O methods generally collect information from the user (or a file), create objects, organize the interaction of these objects, collect the results of these interactions and finally collect and display or store the results these interactions. The special `public static void main(String[] args)` method is often the only method in one of these classes. This method is called to start execution of a Java application.

File Structure

Each *public* class definition is placed in a *.java* source code file having the same name as the class. This file may contain additional classes provided that they are not declared to be *public*.

Class Template

```
/*
 Introductory comments go here.
 What the class does or is for and who wrote it.
 Version and/or date of last revision. For example:

 This is a sample Class file.
 Created by Dr. Charles S. Tritt
 Last revised 4/12/05.
 */

//Import statements. Possibly like:
import java.util.*;

public class ClassName {

    // Note "working" classes like this generally don't contain I/O.

    // Data members store the state of objects.
    // Data member and constant definitions go here, for example:
    final int MY_CONSTANT = 3;
    int dataMember;

    // Method (member function) definitions, including constructors, go here.
    // For example:

    public ClassName(int parameter) {
       // Some statements. Possibly like:
       dataMember = parameter * MY_CONSTANT;
    }

    public int aMethod() {
       // Note methods "do the work"
       return dataMember;
    }
}

// Additional, non-public classes as needed or desired go here.
// I recommend putting a main function for class testing here.

class ClassNameTest {

    // main must be public static & void!

    public static void main(String[] args) {
        // Generally the main class will do the I/O and
        // "organize" the objects.
        System.out.print("Enter an integer: ");
        int argument;
        Scanner sysin = new Scanner(System.in);
        argument = sysin.nextInt();
        ClassName objectName = new ClassName(argument);
        System.out.println("Result: " + objectName.aMethod());
    }
}
```