Java Tutorial on Methods (Subroutines)
Tom Huber (huber@gac.edu), Gustavus Adolphus College

As you have probably noted, there are sometimes sections of code that you will frequently want to execute. For example, if you wanted to turn off all of the LEDs on the display, you could include the following section of code:

```java
for (i=0;i<=7; i++) {
    LEDs[i].setOff();
}
```

It would be much more convenient if you could issue a single command `LEDSOff();`
And have the program do the commands needed to turn off the LEDs. Similarly, when you are starting to use your robots, you will want to issue a command, such as `LeftTurn();` and have the robot turn on the appropriate motors for the appropriate amounts of time to turn the robot to the left.

These are examples of what are called Methods in Java. Other names for this would be a subroutine or a procedure. See the sample program SampleMethods1

I. Make a method to turn off all LEDs
   a. Add the following code to your program after the end of the main procedure:

   ```java
   void LEDSOff() {
   /*
    * This Subroutine turns off all of the LEDs
    * Tom Huber, January 7, 2008
    */
   int i;
   ITriColorLED [] leds = EDemoBoard.getInstance().getLEDs();
   for (i=0;i<=7;i++) {
       leds[i].setOff();
   }
   }
   ```
   b. In your main program, turn on some LED’s using the setOn() command.
   c. Add the statement `LEDSOff();`
in your main program. It should turn off all of the LEDs.

II. Make a method to set LED color and turn on the LED.
   a. Since you may frequently want to turn on an LED and set its color. You could make a subroutine that will do both operations. The other advantage of using this subroutine is that it would not be necessary to include the ITriColor statement in your main program – this would all be buried in the subroutine.

   ```java
   void LEDRed(int i) {
   /*
    * Turn an LED to red
    * Tom Huber, January 8, 2008
    * Input:  i (Integer) : LED Number to set
    * Output: Turn on this LED
    */
   }
ITriColorLED [] leds = EDemoBoard.getInstance().getLEDs();
leds[i].setColor(LEDColor.RED);
leds[i].setOn();
}

b. Notice that this method has an input variable “i” that is being sent to it. The syntax for a method is that you specify the variable type, followed by the variable name. This will be the name used inside of the subroutine only.

c. In your main program, you can include the statement
   LEDRed(i);
   And it will both set the LED color to red and turn it on. Again, since you have defined the ITriColorLED statement in the subroutine, you would not need to include this in your main program.

III. Method that returns a value: SwitchesClosed()
   a. The following Method returns an integer value that represents which switches are closed (0=Neither, 1=SW1, 2=SW2, 3=Both)

   int SwitchesClosed () {
       /*
       * Returns a number representing the switches that are closed
       * Tom Huber, January 7, 2008
       *
       * Input:  None
       *
       * Output: 0 => Neither Switch Closed
       *         1 => Only SW1 Closed
       *         2 => Only SW2 Closed
       *         3 => Both SW1 and SW2 Closed
       *
       * Sample Calling Sequence:
       *   while ( SwitchesClosed() != 3) { // Wait till both closed
       */

       ISwitch SW1 = EDemoBoard.getInstance().getSwitches()[0];
       ISwitch SW2 = EDemoBoard.getInstance().getSwitches()[1];

       int ReturnValue=0;

       if (SW1.isClosed()) {
           ReturnValue = ReturnValue + 1;
       }

       if (SW2.isClosed()) {
           ReturnValue = ReturnValue + 2;
       }

       return (ReturnValue);
   }

   b. Putting the keyword “int” before the method name indicates that the method is to return an integer value. The value that is to be returned is indicated in the return statement at the end of the function.

IV. See SampleMethods2 for example of how to do nested for loops, as well as the break function.