ITI 1120
Lab #3

Use 2 Methods

- Use two methods for developing your solutions
  - Main method:
    - interacts with the user in terms of input/output.
    - Calls the problem solving method to perform a task.
  - Problem solving method:
    - Receives some parameters from the main method, does some computation and returns a result to the main method.

Program Template

```
class Template // Replace 'Template' with your own program name.
{
    // the main method contains all interactions with the user
    public static void main (String[] args)
    {
        // DECLARE VARIABLES/DATA DICTIONARY
        // READ IN GIVENS

        Problem_Solving(); // Calling the Problem_Solving method

        // PRINT OUT RESULTS
    }

    // the 'main' method calls the Problem_Solving method

    public static ? Problem_Solving(?)
    {
        // DECLARE VARIABLES

        // BODY OF ALGORITHM

    }

    /*Replace this with a descriptive comment for each method.*/
}  // Don't remove this brace bracket!
```

Getting ready to write programs…..

- You may use **Template.java** as a guide for your programs
  1. Copy **Template.java** (from Lab 3 on Blackboard Learn into your working directory.
  2. Start up Dr. Java
  3. Click on "open" and select **Template.java**
  4. Start a "new" file.
  5. Copy and paste the contents of the **Template.java** file into the (empty) unnamed file.
  6. Close the file **Template.java**

- In your own program, do not forget to replace the two "?" in the Template.java with whatever is appropriate.

Exercise 1

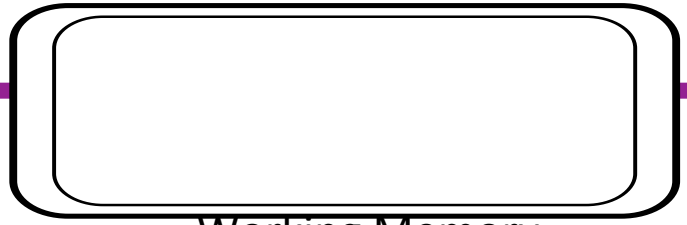
- Write the complete Java program
  - Use the two methods
    - Main method:
      - Main gets 3 numbers from the user
      - Invokes the problem solving method with those 3 numbers
      - Gets the average back
    - Problem solving method:
      - Receives 3 numbers
      - Computes the average
      - Returns average

Exercise 2

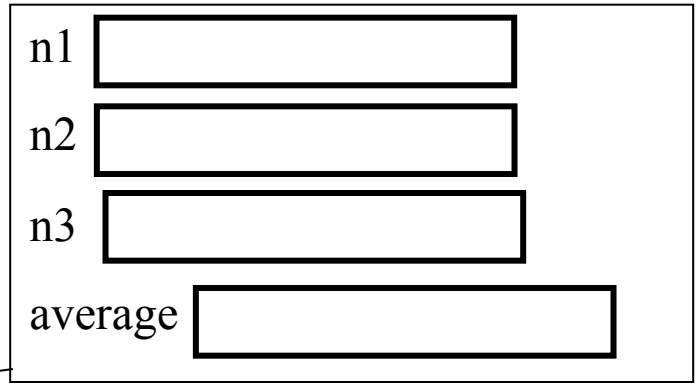
Program Memory

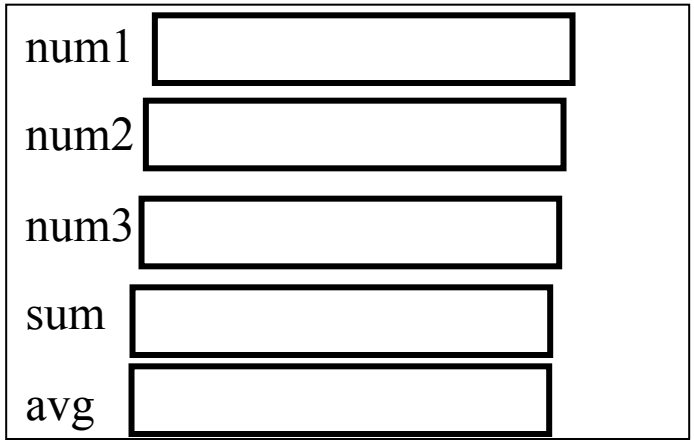
```
/* Lab 3, Exercise 1. */
class Average
{
   // the main method contains all interactions with the user
   public static void main (String[] args)
   {
        // Declare Variables
        double n1, n2, n3;  // numbers to averages
        double average;     // the averagage
        // prompt the user to enter 3 numbers
        System.out.print( "Enter three numbers: " );
        n1 = ITI1120.neadDouble(); // read first number
        n2 = ITI1120.readDouble(); // read second number
        n3 = ITI1120.readDouble(); // read third numre
        // call problem solving method
        double result = average( n1, n2, n3 );
        // display maximum value
        System.out.println( “The maximum is: " + result );
   }
   public static void averagedouble num1, double num2,double num3)
   {
        // Declare variables
        double sum;   // sum of the numbers
        double avg;    // RESULT: average of the numbers
        // BODY
        sum = num1 + num2 + num3 ;
        avg = sum / 3;
        // RETURN RESULTS
        return(avg);
   }
}
```

Terminal/Output Screen



Working Memory





Using the Debugger

- Using Dr. Java's "debug mode", you can trace your Java program as it runs.
  - You can go through the program one step at a time.
  - You can stop the program at "break points" of your choosing.
  - You can check the values of variables.

- Try this for the program you wrote for Exercise 1, the average of 3 numbers.
- Use the programming mode to follow the execution of the program.

Break Points

- Select a line of your program, and under the debug menu, choose "toggle break point on this line".
  - The first `System.out.println` statement is a good choice
  - This will change the colour of the chosen line of code to red.
- You can also right-click on a line and select "Toggle BreakPoint".
  - Many lines can be (de)selected this way.
- When you run the program, the program will stop just before this line is going to be executed.
- In the interactions window, the debugger will tell you where the program is, and the current line of code will be coloured light blue.

Watches

- To keep track of the values of variables as they change, use a "watch"
  - Double-click on an empty area in the "name" column, then type in the name of a variable, and hit 'enter'.
  - If the variable already has a value, it will be shown. If the variable does not yet have a value, the value will say <not found>.
- Try this for all of the variables you use in your program for example 1.
- As the program executes, each time the program stops in the debugger, the current values of the variables will be shown.

Controlling Execution

- With the debugger, there are four ways to advance through a program
- Resume
  - The program will run up to the next break point, or the end of the program if there are no more break points.
- Step into
  - Use this for the most detailed debugging
  - The program will move to the next statement – even if that statement is in another method.
  - This will not go into methods in the Java software development kit.

Controlling Execution

- Step over
  - Most commonly used
  - Use this to move to the next statement in the current method.
  - If the current line of the program calls one or more methods, all of those methods will be invoked, and returned from.

- Step out
  - Often used when you have stepped into a method and you want to go back quickly to the previous method.
  - Use this to run as far as the end of the current method.

- Try using "Step over" to go through your Exercise 1 program one statement at a time.
- But use "Step into" when you arrive at the call of the problem solving method (computeAverage).

Exercise 3

Write a complete Java program that asks a user for temperature in Fahrenheit and converts it to Celsuis.

Your program should have

1. the main method that communicates with the user
2. and a method caled calculateCelsuis that converts temperature expressed in Fahrenheit to Celsius, according to the following formula:

$$C \leftarrow (F - 32) * 5 / 9$$

and returns the computed value to the main.

Exercise 4

Write a complete Java program that given a two digit positive integer prints that number in reverse.

- For example: Your program will transform the two digit integer 12 into 21 (and print 21).

- Hints:
  - The first digit is the result of dividing the integer by 10 (integer division)
  - The second digit is the remainder of the division by 10
    - e.g.:original integer: 12
      - first digit is 12 / 10 = 1
      - second digit is 12 % 10 = 2
- Use two methods model – as you did in Exercise 3).
- Complie and Test your program

Built-in math functions

- The `Math` class
  - Automatically loaded: no import required.
- `Math.abs()` - absolute value | x |
- `Math.pow()` - exponentiation
- `Math.sqrt()` - square root $\sqrt{x}$
- Examples
  - `Math.abs(-3)` Result: `3` | -3 | = 3
  - `Math.pow(2,5)` Result: `32.0` $2^5 = 32$
  - `Math.sqrt(49)` Result: `7.0` $\sqrt{49} = 7$
- See other math functions in Section 5.9 of the textbook
- On line description at http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Math.html

Exercise 5

- Consider the following problem: Given coordinates of 2 points in the plane (xa,ya) and (xb, yb), compute the distance between the two points, according to the following formula:

$$\sqrt{(xa - xb)^2 + (ya - yb)^2}$$

Write a complete Java program for the above problem. (Use two methods – as you did in Exercises 3 and 4).

Compile and test your code.

Note

- According to standard convention, the class names in Java start with Upper-case and names for variables and methods start with lower case.

- Use indentation to make your programs readable
  - HINT: in Dr Java, if you type Cntrl-A (all your code will be selected) and then type Tab, Dr Java will organize your code using standard indentation convention.