Reading:
Chapter 2.1 - 2.6   pages 58-91

Terminology:

string literal	assignment statement	arithmetic operator
method	assignment operator	operator precedence
parameter	constant	increment / decrement operators
string concatenation	byte, short, int, & long	assignment conversion
escape sequence	float & double	promotion
variable	char	casting
variable declaration	boolean	delimiter
variable initialization	character literal	

COMP 1210 Coding Standard Additions

- Follow the standards in the book when naming variables and constants.

  o Class names should begin with a capital letter, and method names and variables should start with a lower case. Both should use medial capitals (also called "camelCase".
  Examples:

    FirstProgram, HelloWorld, main, numberPeople, xAxis, result, finalResult

  o Local variables should be declared at the beginning of the method in which they are declared.

  o Constants should be in all caps with words separated by an underscore. In addition to the type, a constant must be declared as *static final* (e.g., static final int MAX_NUMBER). Constants must be placed at the class level (e.g., before the first method in the class). Examples: *MAX_NUMBER*, *BASELINE*, *TAX_RATE*

  o Both variables and constants should have descriptive variable names. **This convention is not enforced by Checkstyle; however, it will be graded by your GTA in the projects.** Example:
  Good ☺: `int numberJobs, numEmployees;`
  Bad ☹: `int n, m;`

Due:

Activity (in-lab): Monday, August 29, 2016 by the end of lab (you may start on this before lab)

Goals:

By the end of this activity you should be able to do the following:

> ➢ Get user input using the Scanner class
> ➢ Create variables and concatenate variables with a string
> ➢ Use arithmetic operators to manipulate **integer** and **double** data types
> ➢ Use a basic *if* statement to determine which output is appropriate
> ➢ Run your program in a viewer canvas 📽, including creating and saving the canvas file

Program Description:

The goal of this activity is to create a program that displays someone's name, their age, and calculations based on their age.

In-lab Directions: <u>After you enter each segment of code below, be sure to compile and/or run your program. You should correct any errors prior to entering additional code.</u>

Part 1: Getting user's name and age (20%)

- Create the following class in jGRASP and
  Save as H:/COMP1210/Lab2/AgeStatistics.java

```java
import java.util.Scanner;

public class AgeStatistics {

    public static void main(String[] args) {

    }
}
```

- Declare the following variables in the main method then compile your program:

```java
Scanner userInput = new Scanner(System.in);
String name = "";
int ageInYears = 0;
int gender = 0;
double maxHeartRate = 0;
```

- Prompt the user for their name: [Use the `print` method rather than the `println` method.]

```java
//Prompt the user for their name:
System.out.print("Enter your name: ");
name = userInput.nextLine();
```

- Prompt the user for their age:

```java
//Prompt the user for their age:
System.out.print("Enter your age in years: ");
ageInYears = userInput.nextInt();
```

> For this bold text, you have to come up with the code on your own!

- **Add lines to prompt for and input the user's gender after the previous code. Make sure that your program has the EXACT output as below.**

```
  ----jGRASP exec: java -ea AgeStatistics
Enter your name: Pat
Enter your age in years: 71
Enter your gender (1 for female and 0 for male): 1

  ----jGRASP: operation complete.
```

Part 2: Converting age (20%)

- First, calculate and print the age of the user in minutes (minutes = years * 525600)

```java
//convert age:
System.out.println("\tYour age in minutes is "
        + ageInYears * 525600 + " minutes.");
```

- Now calculate the age in centuries. <u>You should be familiar with type conversion between numeric types (pg 83-86 in the book) for the quiz on Wednesday</u>.

```java
System.out.println("\tYour age in centuries is "
        + (double) ageInYears / 100 + " centuries.");
```

```
  ----jGRASP exec: java -ea AgeStatistics
Enter your name: Pat
Enter your age in years: 71
Enter your gender (1 for female and 0 for male): 1
        Your age in minutes is 37317600 minutes.
        Your age in centuries is 0.71 centuries.

  ----jGRASP: operation complete.
```

Part 3: Max heart rate prompt (20%)

- Add the code to print max heart rate:

```java
// display max heart rate
System.out.print("Your max heart rate is ");
```

- Add an *if-else* statement that will be used to assign maxHeartRate to the max heart rate calculation for male and female. <u>The comments below are there to help you in the Part 4 where you will insert the code to do the calculations on the line after each comment.</u>

```java
if (gender == 1) { //calculate female MHR

}
else { //calculate male MHR

}
```

- Add one more print statement to print the max heart rate:

```java
System.out.println(maxHeartRate + " beats per minute.");
```

Part 4: Max heart rate calculations (20%)

- On your own, complete the *if-else* statement by adding an <u>assignment statement</u> for female and another male that assigns the appropriate expression below to the variable maxHeartRate.

   The maximum heart rate is calculated as follows:
   - Max heart rate for females:
     ```
     209 - (0.7 * ageInYears)
     ```

   - Max heart rate for males:
     ```
     214 - (0.8 * ageInYears)
     ```

   Test your program and make sure that the output is exactly as shown below. Your lab instructor may ask you to use a different set of values to test your program when you demo it for your activity grade.

Output # 1:
```
 ----jGRASP exec: java -ea AgeStatistics
Enter your name: Lucy
Enter your age in years: 25
Enter your gender (1 for female and 0 for male): 1
        Your age in minutes is 13140000 minutes.
        Your age in centuries is 0.25 centuries.
Your max heart rate is 191.5 beats per minute.

 ----jGRASP: operation complete.
```

Output # 2:
```
 ----jGRASP exec: java -ea AgeStatistics
Enter your name: Kao
Enter your age in years: 19
Enter your gender (1 for female and 0 for male): 0
        Your age in minutes is 9986400 minutes.
        Your age in centuries is 0.19 centuries.
Your max heart rate is 198.8 beats per minute.

 ----jGRASP: operation complete.
```

Part 5: Using the jGRASP Viewer Canvas (20%)

In this part, you will run AgeStatistics.java in a *viewer canvas*. In the process, you will create visualization of your program that can provide insight into what exactly happens when the program executes (runs). Although AgeStatistics.java is a simple program, this exercise is intended to introduce you to viewer canvas so that you'll know how to use it when you need to debug more complicated programs. For more information on using the canvas see Section 2.10 in *Getting Started with jGRASP 2.0* (Shift-click to open the link in separate browser window).
   http://www.jgrasp.org/tutorials200/jGRASP_02_Getting_Started.pdf

- With AgeStatistics.java in the CSD edit window, click the Run in Canvas button 🖳 on the toolbar. You should see an empty viewer canvas window as shown below. In the CSD window, you should see that the program is stopped at the first executable statement:
 `Scanner userInput = new Scanner(System.in);`
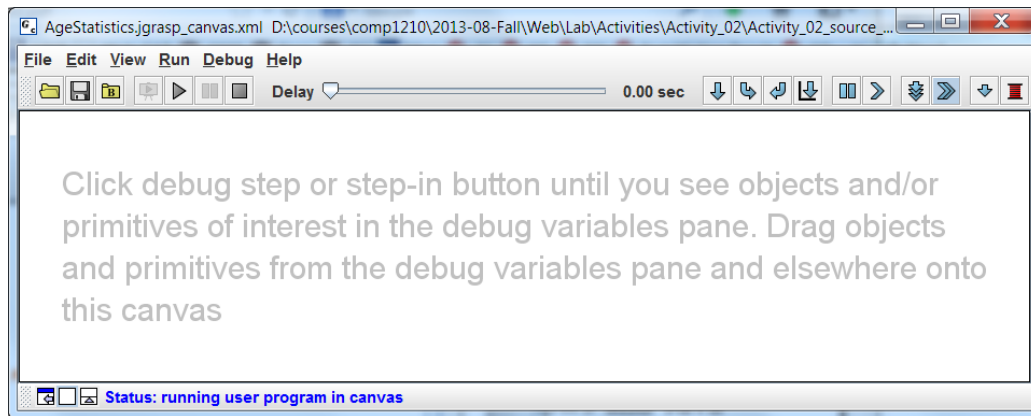
- As indicated by the message in the canvas, click the debug step button ⬇ to execute the statement to create a Scanner object on System.in which is the keyboard. You should now see the variable name `userInput` in the Debug tab. Now select and drag `userInput` from the Debug tab onto the canvas (i.e., left-click the mouse on `userInput` and hold as you drag it onto the canvas). You should now see a viewer for `userInput` in the canvas. This will show the stream of characters coming in from the keyboard after you enter input.
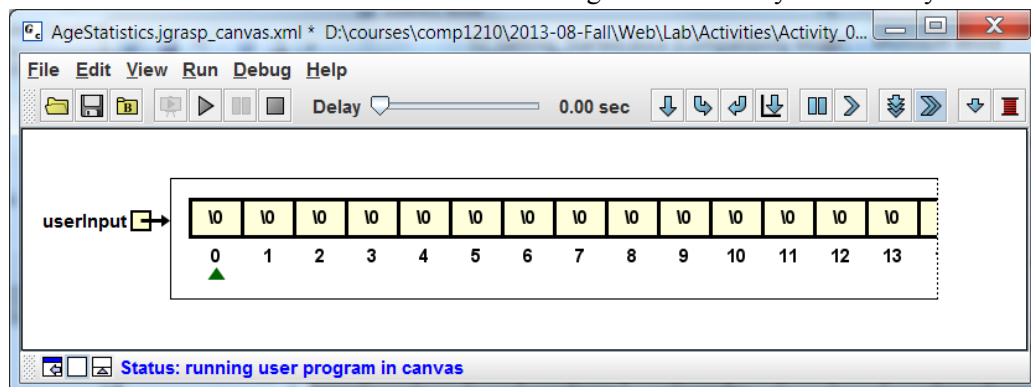
- Now click the step button ⬇ <u>four</u> more times so that you see `name`, `ageInYears`, `gender`, and `maxHeartRate` in the Debug tab. Now drag each of these variables into the

canvas window and arrange them as shown in the figure below. Click the Save button 🖫 on the canvas toolbar. You have now constructed and saved your first canvas.
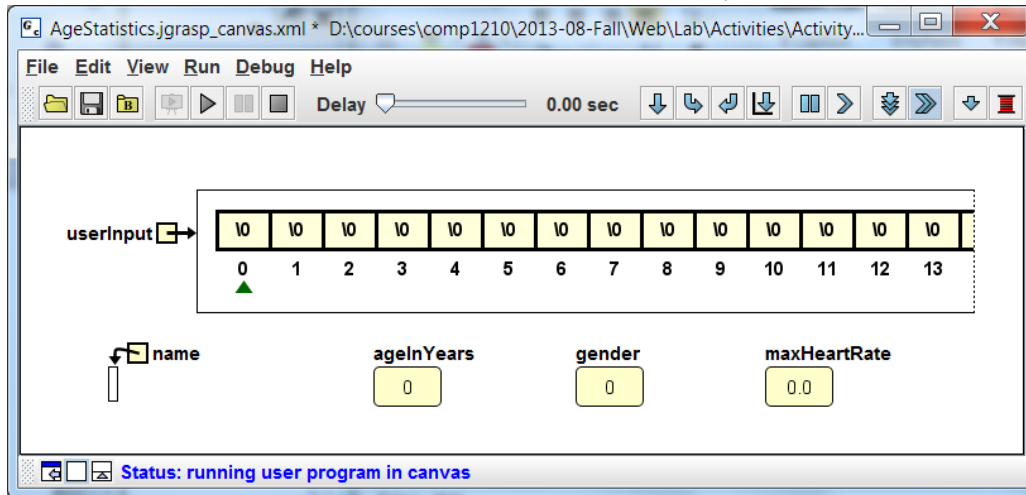
- To complete the *run in canvas*, you can continue to click the step button ⬇ and enter the required input (name, ageInYears, etc.), or alternatively, since the canvas has all the viewers we need, you can click the play button ▷ (which means auto step-in) on the canvas. After the *run in canvas* ends, close the canvas window. Now click the *run in canvas* button on the toolbar as you did in the first step above. Your saved canvas file should open in the canvas window. Click the play button ▷, and your program will begin auto-stepping until is pauses waiting for input. After you enter the requested input each time, the viewers on canvas will be updated. The *run in canvas* will continue until your program terminates at which time the viewers on the canvas will display their last values as shown in the example below. Note that the Scanner `userInput` indicates that "18" was entered followed by a press of the Enter key (which produces the escape characters **\r\n** on Windows or **\n** on Mac OS X or Linux), then "1" was entered followed by a press of the Enter key. The green arrowhead indicates that the "1" has been read but not the escape character(s) **\r\n** or **\n**.

Be sure to demo the ***run in canvas*** <u>for your instructor before you leave lab</u>.