CS107
Introduction to Computer Science

Java Basics

Java

- Java was developed in the early 90s by Sun Microsystems
- Java is a high-level language
- Java programs are portable across platforms
  - Each program is translated into Java bytecode
  - Each machine has a Java Virtual Machine (JVM) which knows how to execute Java bytecode
- Java is object-oriented
  - We will not use objects in this class

A Java program

```
/*
Here you describe what your program does.

Laura Toma
Csci107
*/

public class CLASS-NAME {
    public static void main (String args[]) {

        // your program goes here

    } // end of main
} // end of class
```

Compiling and Running

In order to run a Java program:

- First you compile it
  - that is, you run a program called **compiler** that checks whether the program follows the Java syntax
  - if it finds errors, it lists them
  - If there are no errors, it translates the program into Java bytecode
  - Example: assume you created a program called Hello.java
    prompt>javac Hello.java
  - If successful, this creates a file Hello.class which contains the translation (Java bytecode) of Hello.java
- Then you execute it
  - That is, you call the Java Virtual Machine to interpret and execute the Java bytecode of your program
  - Example:
    prompt>java Hello

The infamous **Hello world** program

When learning a new language, the first program people usually write is one that salutes the world :). Here is the Hello world program in Java.

```
/*
This program prints out "hello world!" and terminates.
*/
public class Hello {
    public static void main (String args[]) {

        System.out.println("Hello world!");

    } // end of main
} // end of class
```

Notes

- Comments
  - what follows after // on the same line is considered comment
  - Or, what is in between /* this is a comment */
- Indentation
  - is for the convenience of the reader; compiler ignores all spaces and new lines ; the delimiter for the compiler is the semicolon
- All instructions end by semicolon
- Lower vs. upper case matters!!
  - Void is different than void
  - Main is different that main

Writing to user (output)

System.out.println(variable-name);
prints the value of variable <variable-name> to the user

System.out.println("any message ");
prints the message within quotes to the user

System.out.println("hello" + "world" + a + "plus" + b);
assuming the value of a is 3 and of b is 7, it prints helloworld3plus7

Note: System.out.println() always prints on a new line.

Example

```
/*
This program illustrates the System.out.println command.
*/
public class Hello {
    public static void main (String args[]) {

        System.out.println("This is my first Java program!");
        System.out.print("I like Java.");
        System.out.print("I think Java is cool.");

    } // end of main
} // end of class
```

- Exercise: change the print to println above. What is the difference?

Variable declaration

type variable-name;
Meaning: variable <variable-name> will be a variable of type <type>

Where type can be:

- int //integer
- double //real number

Example:

```
int a, b, c;
double x;
int sum;
```

Example

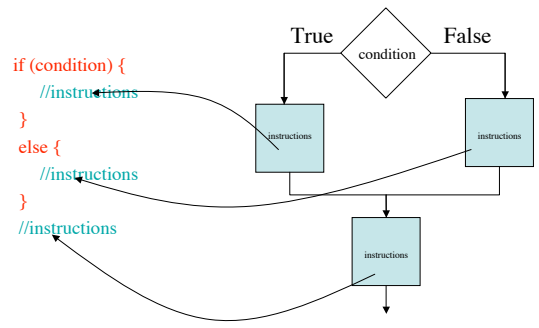
```
/*
Printing ages.
*/
public class MyFirstJavaProgram {
    public static void main (String args[]) {

        int myAge, myFriendAge;  /* declare two integer variables */

        myAge = 20;
        myFriendAge = myAge + 1; //one year older
        System.out.println("Hello, I am " +myAge + "years old, and my friend is " +
                           myFriendAge + " years old");
        System.out.println("Goodbye");

    } // end of main
} // end of class
```

Reading from user (Input)

```
/*
This program shows an example of reading from user.
*/
public class  MyFirstJavaProgram{
    public static void main (String args[]) {

    /* this line should appear once in your program;  basically it declares a variable r which knows how read
    input from the user */
    ReadStream r = new ReadStream();

    /* Then you can use r.readInt to read an integer value from the user. */

    System.out.print("Dear user, please enter an integer number:");
    int a;
    a= r.readInt();
    r.readLine();
    /* Meaning: read an integer from the user and store it into the variable called a */

    } // end of main
} // end of class
```

If statements

```
if (condition) {
    //instructions
}
else {
    //instructions
}
//instructions
```



Boolean conditions

..are built using

- Comparison operators
  - == equal
  - != not equal
  - < less than
  - > greater than
  - <= less than or equal
  - >= greater than or equal

- Example:

```
int x, y;  //two variables
//assume they have some values
 if (x <= y) {
     System.out.println("x is smaller");
} else {
     System.out.println("x is larger");
}
```

Class work

- Write a program that asks for two scores and prints out the loser and the winner. For example, it could look like this (or use your imagination!) . Make sure you handle all cases (victory, loss, tie).

```
Tell me Bowdoin score: 23
Tell me Bates score: 16
Congratulations, Bowdoin won!!
Goodbye.
```

Class work

- Write a program to compute the miles-per-gallon gas consumption of a car during a trip. Your program should ask the user for start and end mileage, and the amount of gas consumed. Then it should print out the result, and an appropriate message, depending whether the consumption is lower, equal to, or greater than 25 miles-per-gallon.

```
Enter the start mileage of your trip:  56345
Enter the end mileage of your trip: 56987
Enter the number of gallons you filled the tank: 20.7
Your car gives 31.15 miles-per-gallon.  Congratulations!!
Goodbye.
```