**CS101**
**Input, I/O**

**Input using Native Java**

So far we have "hard coded" all values we want directly in our code. For example, in the race pace calculation, we put the pace and distance directly as variables in our program. It would be nice to be able to run our program, have it ask the user to enter what pace and distance we want, let the user enter those values, and then calculate the time. This makes the program much more general and functional with easily-changed values instead of having to change the values in the program, recompile it, and then re-run the program.

Unfortunately, input in Java is a bit unwieldy (this has been one of its criticisms). All data that is input from the keyboard must be read as a String. Recall that a String is a sequence of ASCII characters. If we want to read a number, such as an integer, then we must convert the String to an integer. For example, this means converting from ASCII to two's complement.

Here is an example of a program that shows how to read a string in Java:

```
import java.io.*;
class InputTest {
      public static void main(String[] argv) throws Exception {
                String sUserInput;
                BufferedReader inFromUser = new BufferedReader(
                              new InputStreamReader(System.in));

                System.out.println("Enter something: ");
                sUserInput = inFromUser.readLine();
                System.out.println("You entered: " + sUserInput);
      }
}
```

In the first line, we are importing the java.io package. This package includes code routines we will need that read data from the keyboard. This is code written by someone else (at Sun Microsystems) and illustrates the idea of code reuse – once some code is written to perform some function, it is easy to use that code in new programs.

In the definition of main, we added "throws Exception" before the body of the method. An **exception** occurs when something unexpected happens, typically an error. For example, if data could not be read, then an exception may occur. Another situation that would cause an exception is behavior like trying to divide a number by zero. This is undefined (or infinity in some cases). By adding this statement, Java will notify us if an exception occurs.

The variable sUserInput is a string variable that will hold the contents of what the user types at the keyboard.

InFromUser is a variable of type BufferedReader. Don't worry about what this line does, it essentially defines an object that is capable of reading input from the keyboard (System.in). However, you might be wondering why sometimes we use the keyword **new** and sometimes we don't. The answer is that whenever we want to create an Object or Class, we must use *new* to create an instance of that class. By invoking *new*, we are allocating memory and initializing the object. However, whenever we are declaring a simple variable (type int, float, char, double, etc.) we don't need to worry about initializing or allocating any extra memory and therefore don't need to use *new*.

The line : sUserInput = inFromUser.readLine(); waits for the user to type data at the keyboard. The data entered by the user is stored in ASCII format in the variable sUserInput.

The last line simply prints out the contents of this variable.

**Reading Values Other Than Strings**

The above code shows how to read data into strings. What if we want to read data into other types of variables, such as integers or doubles or floats? The answer is that we must read the data as a string, and then convert it to the respective type of interest. This is not as simple as typecasting the string, because we need much more logic to translate a string of characters to a 2's complement or IEEE 754 value (float or double).

For example, Java will complain if you try the following:

```
import java.io.*;
class InputTest {
      public static void main(String[] argv) throws Exception {
                String sUserInput;
                int i;
                BufferedReader inFromUser = new BufferedReader(
                              new InputStreamReader(System.in));

                System.out.println("Enter an integer: ");
                sUserInput = inFromUser.readLine();
                i = (int) sUserInput;
                System.out.println("You entered: " + i);
      }
}
```

This results in an error message while compiling, complaining that i requires an integer be assigned to it.

Fortunately, Java has already defined code that does this conversion for us. Examples are shown below for ints, floats, and doubles:

```
import java.io.*;

class InputTest {
      public static void main(String[] argv) throws Exception {
                String sUserInput;

            int i;
            float f;
            double d;
            BufferedReader inFromUser = new BufferedReader(
                        new InputStreamReader(System.in));

            System.out.println("Enter an integer. ");
            sUserInput = inFromUser.readLine();
            i = Integer.parseInt(sUserInput);
            System.out.println("You entered: " + i);

            System.out.println("Enter a double. ");
            sUserInput = inFromUser.readLine();
            d = Double.parseDouble(sUserInput);
            System.out.println("You entered: " + d);

            System.out.println("Enter a float. ");
            sUserInput = inFromUser.readLine();
            f = Float.parseFloat(sUserInput);
            System.out.println("You entered: " + f);

            // Now sum up all three things entered as a double
            d = d + (double) f + (double) i;
            System.out.println("The sum of all three is: " + d);
      }
}
```

Things to note: The line

Integer.parseInt(sUserInput);

is case-sensitive! In particular, notice the capital I on Integer. This is necessary, because Integer is different from int. Integer refers to an object defined in Java that deals with integers. One of the things this object can do is convert from String to the Integer object. However, we then have to convert from the Integer object to the actual int value (i.e., the 2's complement value). The parseInt() method does this conversion.

There is a similar behavior for doubles and floats. Double refers to the Java object that deals with doubles, while Float refers to the Java object that deals with floats. We could do the same thing with char's and long's by creating a Char object or a Long object and then invoking parseChar or parseLong.

Here is the output for a sample run:

```
Enter an integer.
5
You entered: 5
Enter a double.
2.412
You entered: 2.412
Enter a float.
3.14
You entered: 3.14                    // Note roundoff below!
The sum of all three is: 10.552000104904174
```

**Using the Console Class for Input**

Performing input with what Java provides directly can be a bit cumbersome as seen above. Fortunately, the authors of your textbook have provided a package you can use that simplifies this process. They have written code to make it easy to input data. This code has been compiled and distributed as a class, or object, file. The class is called **Console** because it includes console (i.e. the keyboard) functionality. Note that this class is not part of standard Java, so the code for the class must be available for programs that want to use it.

To use this class, the file Console.class it must be in the same directory as the program you are working on. On mazzy, this file can be copied from the ~afkjm/cs101misc directory into your own directory. Or, you can download the file from http://www.math.uaa.alaska.edu/~afkjm/cs101/Console.class .

The Console class provides a way to read a value and give a prompt in one statement to either read in an integer, double, char, or string. The format is:

```
intVar = Console.readInt("Prompt");
strVar = Console.readString("Prompt");
doubleVar = Console.readDouble("Prompt");
charVar = Console.readChar("Prompt");
```

Here is an example that squares two integers:

```
public class Square {
      public static void main(String[] args) {
            int n;

            n = Console.readInt("Enter a whole number: ");
            System.out.println("The square is " + n*n);
      }
}
```

To input an integer, we invoke the method Console.readInt. This method displays a prompt of "Enter a whole number". The value entered by the user is converted into an integer and stored in variable n. Finally, n*n is output:

```
Enter a whole number:  9
The square is 81
```

Here is another version that builds on this simple example. Can you tell what it will do?

```
public class Square {
      public static void main(String[] args) {
            int n;
            n = Console.readInt("Enter a whole number");
            System.out.println("The square is " + n*n);
            System.out.println("The cube is " + n*n*n);
      }
}
```

Here is an example that reads in a double and a string in addition to an integer:

```
public class ConsoleTest {
      public static void main(String[] args) {
            int i;
            double d;
            String s;

            i = Console.readInt("Enter a integer ");
            System.out.println("You entered: " + i);

            d = Console.readDouble("Enter a double ");
            System.out.println("You entered: " + d);

            s = Console.readString("Enter your name ");
            System.out.println("Hello, " + s);

            d = d+ (double) i;   // try summing up the values
            System.out.println("Sum of the values entered is " + d);
      }
}
```

This new program will prompt the user for three types, for an integer, double, and a string. The output is all sent to the text screen.

```
Enter a integer  5
You entered: 5
Enter a double  3.4
You entered: 3.4
Enter your name  Kenrick
Hello, Kenrick
Sum of the values entered is 8.4
```

Feel free to use either the Console class or native Java I/O input to read data into variables.