CSE20 : Lab #4 - Data Types

Overview

This week we will be exploring different data types and the different meanings we can extract from the same bits (binary). It's a look inside the hood at what we have covered in lecture and some types you have already used in previous lab. This lab is also aimed at making you more comfortable with everything we have done so far.

Variables

We relied heavily on variables in lab 3 after introducing them in lecture. It is important to understand the use of variables and difference in the errors or warnings that stems from their usage. Generally it is useful to give meaningful names to the variables so whoever is reading the code (including yourself) can quickly discern their purpose. So rather than using names like *string1*, you should also use names like *hometown* if you want it to store the person's hometown.

Declaration – It is an error to use a variable before declaring them since technically it does not exist yet. There are three ways of doing this from what you have already seen:

```
int age;                    // Just declaring the type (int) and name (age)
int age = 1;                // Includes initialization of value 1
int age = input.nextInt();  // Declare and assign value at the same time
```

Once you declared a variable, it can be manipulated (write) or referenced (read) from then after.

Write – Assignments are done with the operator '=' so anytime you see a '*variable =*' means we are giving it a new value. We can see in the last two examples of declaration that assignment is done at the same time. Once a variable exists then you can always assign it any valid value for its type. If you never use a variable after declaring then it will create a warning to make sure you didn't mistype the name or forgot about it. In general, only create variables you will assign values and actually use.

Read – If you read a variable's value before any prior assignment then the value of the variable is *undetermined*. So it is prudent and good programming practice to always initialize before reading any variable. Whenever you refer to a variable

then its value is read and used. If a variable is not read from then it will have a warning saying you never used it. It is helpful in debugging in case the names are mismatched or you just simply forgot about the variable.

So let's take a look at simple example from Averages.java:

```
average = (n1 + n2)/2;
```

Here we read the values of *n1* and *n2*. We then add the two numbers and divide by 2. The resulting number is then written or assigned to the variable *average*. Then we read *average* inside a println statement like this:

```
System.out.println(average);
```

Notice that putting " " around anything makes it a string as the following illustrates: `System.out.println("average");` This will always print out the words average and not the value inside the variable *average*. Notice that we didn't write our equation above as ("n1" + "n2")/2 which would be an error. So whenever we refer to the variable then it should just be by its name.

String literals

Now we talk a little more about strings which we used but have not discussed in depth. Everyone is familiar with the simple form of creating strings by putting quotes around it. Ever since the first lab we have figured out how to form sentences using strings. In lab 3, we created string objects and stored the input from the user. We can actually manipulate strings just like other objects or numbers. One common way use is to reduce the number of print statements we need to call. For example before we would use multiple lines to print the following sentence:

```
System.out.print("Their age is ");
System.out.print(age);
System.out.println(".");
```

Just like numbers we can 'add' strings together which is actually called concatenation. So we combine the above into:

```
System.out.println("Their age is " + age + ".");
```

We can now use this form to create dynamic strings.

Type casting

The meaning of a number depends on the type it represents. For *char* type 65 represents a character 'A'. For *int* type it is just 65 and for *float* it will be 65.0. We can see the differences by running the following code:

```
char charA = 'A';

System.out.println("OUTPUT is " + (char) (1 + charA));
System.out.println("OUTPUT is " + (short) (1 + charA));
System.out.println("OUTPUT is " + (int) (4/3));
System.out.println("OUTPUT is " + (float) (4/3));
System.out.println("OUTPUT is " + (double) (4/3));
System.out.println("OUTPUT is " + (float) 1);
```

By putting a type like (int), we can force the number or expression to be that type. If we do (int) 4/3 then its forcing 4 to be an int before division by 3. If we use (int) (4/3) then the result of 4 divided 3 is being forced to type int. Using parentheses is very important to see what the type casting is being applied to. Play with this in your lab to see the different effects before you attempt the exercises.

Use of Scanners

We need to add one line for Scanner objects in order to handle multiple word strings so the answers can be more complete. Also nextLine has a funny behavior when combined with other types due to how newlines are handled. So along with the declaration of a Scanner variable, we set it to use newlines to separate inputs instead of any whitespace (like space or tab).

```
Scanner input = new Scanner(System.in);
input.useDelimiter(System.getProperty("line.separator"));  // add

age = input.nextInt();
city = input.next();
```

Please note that we only need one Scanner to read all the user inputs in a given program. You just call next* methods as you have before to get the different inputs.

(Reading) Chapter 2.1, 2.2 & 2.4

- Answer Participating Activity 2.1.3 & 2.1.5
- Answer Participating Activity 2.2.2 & 2.2.4
- Answer Participating Activity 2.4.3 & 2.4.4

Getting started

After starting Eclipse, create a new project called Lab 4. You can also import Interviewer.java file from Lab 3 if you wish to do the second part of this lab. You may want to look at Averages.java from Lab 3 as an example for asking user for two numbers as input and then outputting a result after calculating it.

(Exercise) Manipulating Numbers

You need to understand Type Casting before you do this exercise. Create a new class called Manipulate with the following behavior

Ask for 2 int numbers from the user just like Averages.java. Then you will display 4 results using those two numbers:

- Add of n1 + n2 is ****
- Sub of n1 – n2 is ****
- Mul of n1 * n2 is ****
- Div of n1 / n2 is ****
- Rem of n1 % n2 is ****

Obviously instead of **** it'll be the result of the operation. **Be sure that the result is an *int* by casting it before printing it out.**

- Ask for additional 3 types of data from user
  - 2 shorts
  - 2 floats
  - 2 doubles
- For each of the 3 types perform all 4 math operations where the result should match the original type, ie addition of two shorts results in a short
  - +, -, *, /
- For each operation, print out the results like you did for ints
- Make sure proper types are created for each output

There should be a total of 20 (4 data types X 5 operations) output lines.

(Assessment) Level of understanding

1. Give the expression for printing "OUTPUT is Z" using variable *charA* by using type casting and manipulating it (like in the examples)
2. Are there certain numbers that do not work for input of Manipulate?
3. Are there any output differences between floats and doubles?
4. Can two large integers after some operation result in a short?

(Exercise) Interviewer Program

Modify the Interviewer from lab 3 with the following requirements

- Answers must be able to handle sentences or numbers
- Add a question about weight in lbs (int type)
- 1 kg = 2.2 lbs
- Output the user's weight in kgs (int type)
- Ask all the questions first before the results are printed out
- Output is all in one paragraph like Biography of the interviewee

What to hand in

When you are done with this lab assignment, you are ready to submit your work. Make sure you have done the following ***before*** you press Submit:

- Include answers to Participating Activity 2.1.3, 2.1.5, 2.2.2, 2.2.4, 2.4.3 & 2.4.4 from your textbook
- Include answers to Level of Understanding 1-4 questions
- Attach your Manipulate.java
- Attach your new Interviewer.java
- List of Collaborators