Lab 04: Data Types
CSE-020-01, Fall 2019 **Points: 20**

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 lectures 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 03 after introducing them in lecture. It is important to understand the use of variables and differences 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 a person's hometown.

Declaration – Using a variable before declaring it will cause an error since the variable does not exist yet. There are three ways of doing this from what you have already seen:

```
int age;                    // declaring an integer variable called age
int age = 1;                // declaring an integer variable called age, and
                            // initializing it to the value of 1
int age = kb.nextInt()      // declaring an integer variable called age and
                            // assigning a value via user input (assuming a
                            // Scanner called kb is already initialized)
```

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 `varIdentifier =`, it means we are giving `varIdentifier` a new value. We can see in the last two examples of variable declarations above that assignments can be done at the of declaration. Once a variable is declared, you can assign it any valid value conforming to its type.

Read – If you read a variable's value before any prior assignment, the value of the variable is undetermined. It is prudent and good programming practice to always initialize before reading any variable's value. Whenever you refer to a variable, its value is read and used in the statement where the reference is made. If you never use a variable after declaration, Eclipse will throw a warning. Such warnings are helpful during debugging in case the names are mityped or you just simply forgot about the variable.

So, let us take a look at a 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 double quotes `""` around anything makes it a string as the following illustrates: `System.out.println("average");` This will always print out the word 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 a variable, it should just be by its name (without quotes or double quotes).

String literals

We will now discuss 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 double quotes around words or sentences. Since the first lab we have figured out how to form sentences using strings. In Lab 03, we created string objects that stored the inputs from the user. We can actually manipulate strings just like other objects or numbers. One common use of strings is to reduce the number of print statements we need in code. For example, we 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, an operation which is actually called concatenation. So we can combine the above statements into:

```
System.out.print("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 the 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)` before a variable, number or an expression, we can force the variable, number or expression to be of that type. This is called explicit type conversion. If we write `(int)4/3`, we are forcing `4` to be an `int` before division by `3`. If we write `(int)(4/3)`, 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

To get an input of strings containing spaces (and/or tabs) we can use the `nextLine()` method with a Scanner object. However, when mixing `nextLine()` input statements with other input statements that get a single item without whitespaces (like a number or word), the program can exhibit funny behavior due to how newline characters are handled [cf. Section 2.15]. Methods like `next()/nextInt()/nextFloat()/…` that get a single item from the user without whitespaces leave a newline character (`\n`) in the input that must be discarded if the very next input statement is going to be using a `newLine()`. Otherwise this `newLine()` statement that follows `next()/nextInt()/nextFloat()/…` will read this leftover `\n` character as the user's input causing odd behavior (and skipping the user's actual input). This is done by inserting a dummy `newLine()` method (that discards the leftover `\n` character) between each `next()/nextInt()/nextFloat()/…` method and `nextLine()` method as shown below.

```
Scanner kb = new Scanner(System.in);

int itemNumber = kb.nextInt();
float weight = kb.nextFloat();
kb.nextLine();              // discards the \n leftover by previous nextFloat()
String itemName = kb.nextLine();
```

```
String itemDesc = kb.nextLine();
String monthOfManufacture = kb.next();
kb.nextLine();              // discards the \n leftover by previous next()
String countryOfOrigin = kb.nextLine();
```

Please note that we still only need one Scanner object to read all the user inputs in a given program. You just need to make sure that a dummy `nextLine()` is inserted between every single item input statement like `next()`/`nextInt()`/`nextFloat()`/… and a `nextLine()` input statement.

Getting Started

After starting Eclipse, create a new project called **Lab20_4**. You can also import `Interviewer.java` file from Lab 03 if you wish to do the second part of this lab. You may want to look at `Averages.java` from Lab 03 as an example for asking user for two numbers as input and then outputting a result after calculating it.

Part 1: Manipulating Numbers

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

Ask the user for two `int` numbers `n1` and `n2` from the user just like `Averages.java` and display the following 5 results using the two numbers (assuming n1 = 4 and `n2 = 3`):

```
Sum of 4 and 3 is 7
Difference of 4 and 3 is 1
Product of 4 and 3 is 12
Integer cast of (4 / 3) is 1
Integer cast of (4 % 3) is 1
```

Be sure that the result is an `int` by casting it before printing it out.

Now repeat the same operations for two `short`, two `float` and two `double` numbers. More precisely, ask the user for three additional types of number pairs from the user:

- Two short numbers
- Two float numbers
- Two double numbers

For each pair of inputs, perform all 5 math operations where the results should match the input type, i.e., addition of two `short` numbers should result in a `short`, and so on. For each operation, print out the results like you did for the `int` numbers. Make sure proper types are created for each output.

There should be a total of 20 (4 data types, each with 5 operations) output lines as shown in the sample output below.

Sample Run (user input shown in blue):

```
Input two integers:
4
3
Sum of 4 and 3 is 7
Difference of 4 and 3 is 1
Product of 4 and 3 is 12
Integer cast of (4 / 3) is 1
```

```
Integer cast of (4 % 3) is 1

Input two shorts (-32,768 to 32,767):
12
15
Sum of 12 and 15 is 27
Difference of 12 and 15 is -3
Product of 12 and 15 is 180
Short cast of (12 / 15) is 0
Short cast of (12 % 15) is 12

Input two floating points:
1.4
2.35
Sum of 1.4 and 2.35 is 3.75
Difference of 1.4 and 2.35 is -0.9499999
Product of 1.4 and 2.35 is 3.2899997
Float cast of (1.4 / 2.35) is 0.59574467
Float cast of (1.4 % 2.35) is 1.4

Input two doubles:
14.756
12.443
Sum of 14.756 and 12.443 is 27.198999999999998
Difference of 14.756 and 12.443 is 2.3130000000000006
Product of 14.756 and 12.443 is 183.60890799999999
Double cast of (14.756 / 12.443) is 1.1858876476733906
Double cast of (14.756 % 12.443) is 2.3130000000000006
```

Part 2: (Assessment) Level of Understanding

Create a Word document or text file named `Part2` that contains answers to the following:

1. Give the expression for printing `OUTPUT is V` using the variable `char charA = A;` by using type casting and manipulating it (as shown in the examples in **Type Casting**).
2. Are there certain numbers that do not work as input of `Manipulate`?
3. Are there any output differences between `float` and `double` numbers in `Manipulate`?
4. Can the result of some operation on two large integers be converted into a `short` without loosing information? If so, what would be such an operation?

Part 3: Interviewer Program

Modify the Interviewer from Lab 03 with the following requirements:

- Answers must be able to handle sentences or numbers. Like full
- Add a question about height in inches (`float` type).
- 1 in = 2.54 cm.
- Output the user's height in centimeters (`float` type).
- Ask all the questions first before the results are printed out.
- Output is all in one paragraph like Biography of the interviewee.

Sample Run (user input shown in blue):

```
What is your name? Santosh Chandrasekhar
What is your favorite number? 42
What are your hobbies? hike and travel
Do you live on or off campus? off campus
What is your favorite color? blue
What is your height in inches? 69.2
```

```
So your name is Santosh Chandrasekhar. Your favorite number is 42. You like to hike and
travel. You live off campus. Your favorite color is blue. Your height in centimeters is
175.77 cm.
```

What to hand in

When you are done with this lab assignment, submit all your work through CatCourses.

Before you submit, make sure you have done the following:

- Attached the file named `Part2` containing answers to the assessment questions.
- Attached the new `Manipulate.java` file.
- Attached the modified `Interviewer.java` file.
- Filled in your collaborator's name (if any) in the "Comments…" text-box at the submission page.

Also, remember to demonstrate your code to the TA or instructor before the end of the grace period.