# CSCE 155 - Java

### Lab 2.0 - Data Types

## Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.
2. Review Oracle's Java Tutorial section on Primitive Data Types:
   http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

## Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

# 1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- Using command line arguments
- Variable declarations
- Basic primitive data types
- How to choose appropriate data types for a given problem

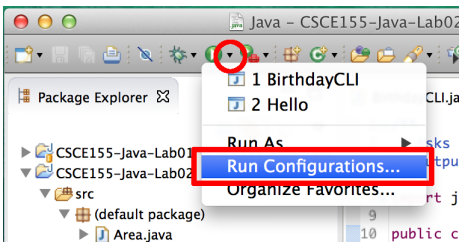
# 2 Activities

## 2.1 Using Command Line Arguments

When you run a program from the command line, you can also provide the program with *arguments* that the program can use for input or for configuration. To understand this, we have provided you two completed Java programs that compute an age given a birthdate. The first works by prompting the user for input interactively, while the second uses command line arguments.

### Instructions

1. Open Eclipse and choose the same workspace on your Z: drive.
2. Open the Git perspective and clone the Lab 2.0 project from Github using the following URL:
   https://github.com/cbourke/CSCE155-Java-Lab02
   For more detailed instructions, refer to Lab 1.0.
3. Open and run the `Birthday.java` program and answer the questions on your worksheet. Note: the program will prompt for input in Eclipse's Console window.
4. The second program, `BirthdayCLI.java` is non-interactive: it reads input directly from the command line when you run it. With an IDE, there is no direct command line, but it is still possible to do so:
   a) Run the program by clicking the "play" button.
   b) An error should be echoed because the program expected command line arguments; since we don?t really have a command line in an IDE, we need to setup a *run configuration*.
   c) Click the play button's down arrow and select "Run Configurations"




d) Click the "Arguments" tab and enter the command line arguments as follows: `James 1955 5 19`.

e) Click "Apply" then "Run"

5. Answer the questions on your worksheet.

## 2.2 Basic Data Types

A variable is a name associated with a memory cell whose value can change. When variables are stored in memory, the computer has to have a way of knowing what type of data is stored in a given variable. Data types identify the type of values stored in a memory location and operations that can be performed on those values. A computer will not use the same amount of memory to store a single letter as it does to store a very large real number, and the values are not going to be interpreted the same way. Therefore, different data types may have different sizes. The (incomplete) table in question 4 on the worksheet shows some basic data types with their sizes and ranges. Size is represented in bytes. A byte is a unit of storage capable of holding a single character. A byte is usually considered equal to 8 bits. A bit (short for binary digit), is the smallest unit of information in a computer–either a zero or a one. Some basic data types can be either signed (either positive or negative) or unsigned (non-negative).

Java is a *statically-typed* language; all variables must be declared by giving them a type and a name before they can be used. Declarations can optionally be followed by an assignment in the same line. Some examples:

```
1  int a;
2  int b = 10;
3  int c, d = 5, e;
4  double x, pi = 3.14;
```

For basic data types (called primitive data types), Java specifies exactly how many bytes each type takes and consequently defines a fixed range for each type. A Java program has been provided, Ranges.java that demonstrates the size and range of each basic data type.

### Instructions

1. Open the source file `Ranges.java` in your Eclipse project and run it.
2. On the lab worksheet, complete all the size and range entries in the table with the values of the output of the program.

Note: Java also supports a primitive `boolean` type that can be assigned the value true or false. Thus, there is no range. Conceptually, a `boolean` should only require one bit, however memory addressing would make this inefficient. In practice, a `boolean` takes 32 bits if in the system stack, and 8 bits when used in arrays.

## 2.3 Currency Conversion

Write a program that will convert US Dollars to British Pounds and Japanese JPY. 10% of the total amount of US Dollars will be taken as an exchange fee. For the rest of the US Dollars, half will be changed to British Pounds and the other half to JPY. Assume the exchange rate is: 1 US Dollar = 0.6 British Pound; 1 US Dollar = 76.8 JPY. The program should prompt the user to input the amount of US dollars then print an appropriate output. An example run would look something like the following:

```
Please input the total amount of US Dollars: 100.00
You get 27.00 British Pounds and 3456.00 Japanese JPY.
```

### Instructions

1. In Eclipse, create a new class named `Dollar` by right-clicking your project and selecting New → class.
2. Write your complete Java program in this source file. Be sure to place your code in a `main` method and to choose appropriate data types for your variables
3. Run your program and answer the questions on your worksheet.

## 2.4 Mixed types

A mixed-type expression is an expression with operands of different data types. When an assignment statement is executed, the expression on the right-hand-side is evaluated, and then the resulting value is placed into the variable on the left-hand-side. For example:

```
1  int x = (8 * 4) + (3 * .5);
```

The data types of the operand affect the data type of the result. This can lead to some initially unintuitive results. When performing division between two integers, the result is necessarily an integer. For example:

```
int a = 10, b = 20, c;
c = a / b;
```

In the code snippet above, the floating point result of `10 / 20` *should* be `0.5`, but the actual value stored in the variable `c` is zero! This is because the result of an operation of two integers is an integer: thus the decimal part of the result is truncated (dropped). We could fix this by making at least one of the operands (and the resulting variable) a floating point variable through type-casting:

```
int a = 10, b = 20;
double c;
c = (double) a / b;
```

You have been given a Java program, `Area.java` that reads in the base and the height of a triangle and calculates the total area.

1. Read through and understand the source code
2. Compile and run your program to answer the remaining questions on your worksheet.
3. Using the previous birthday programs as a reference, change the area program to accept command line arguments instead of prompting for input.

# 3 Handin/Grader Instructions

1. Hand in your completed files:
   - `Dollar.java`
   - `Area.java`
   - `worksheet.md`

   through the webhandin (https://cse-apps.unl.edu/handin) using your cse login and password.
2. Even if you worked with a partner, you *both* should turn in all files.
3. Verify your program by grading yourself through the webgrader (https://cse.unl.edu/~cse155h/grade/) using the same credentials.

4. Recall that both expected output and your program's output will be displayed. The formatting may differ slightly which is fine. As long as your program successfully compiles, runs and outputs the *same values*, it is considered correct.

# 4 Advanced Activities (Optional)

1. Many applications require the use of numerical values greater than or with greater precision than what 32-bit signed integers and 64-bit floating point numbers can represent. It is typical to use a multi-precision library that offers integers and floating point number types with arbitrary precision but with a greater performance cost. Java offers several classes as part of its Standard Developer Kit (SDK): `BigInteger` (http://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html) and `BigDecimal` (http://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html). Read the provided documentation and modify your program to use these types instead so that US National Debt can be represented and correctly converted to Yen.