ESS 201 Programming II
Java
Lab 1
5 Aug 2020

Part 1

This part consists of a set of short programs that are intended to help you get familiar with aspects of Java - basic language features, compiler and runtime environment. No submissions are needed for this section.

A set of Java source files are available in this location in LMS. You should try to compile and run each of them, and use them to help get started in this course. DO NOT USE AN IDE - work on the command line to invoke the compiler and to run the program

As a first step, ensure you have an appropriate JDK installed on your machine. It is preferable that you stay with Java 8 (JDK 1.8_xxxx). Later versions are OK, but please make sure you are not using any features introduced after Java 8.

Copy each of the source files to your machine and try out the following as indicated. In each case, and each step, try to compare what you observe to similar functionality (or compiler/runtime behaviour) in C or Python. We will discuss all these in detail in subsequent lectures.

For each of these cases, compile and then run the program as follows :

```
> javac <filex>.java
> java <filex>
```

(where <filex>.java is one of the files below)

1. HelloWorld.java:

   A minimal Java program, just to get started. Just to make sure your environment is set up correctly. Don't worry about the specific syntax/keywords for now. Experiment with the following:
   - delete either or both of the keywords "public" in this file
   - change the name of the file, but retain the class name as HelloWorld within the file

2. HelloAgain.java

   Just a little more than the first HelloWorld.
   System.out.println(...) is roughly equivalent to fprintf(stdout, …) in C.

Note how you can combine strings and integers as arguments to this method.

3. TestArray.java

   Compile and run this file by uncommenting one section at a time. Some of them will produce errors. Identify if these are compile-time or run-time errors. Can you fix the errors before moving on to the next step?

4. MemTest.java

   Java manages memory differently from C or C++. How large can you make N and/or M before the program crashes or doesn't compile? If you keep M fixed and keep increasing N, what do you notice about the running time of the program?

5. PrintArray.java

   Here we have the function **main** invoke another function or method. What do you notice about the language here?

6. Try the following on any of the above files.

   When you compile using javac, the compiler writes out a file with the same class name as the input file, and with the extension .class. Copy this .class file to a different OS (say from Windows to Linux or vice versa), or a machine with a different architecture, and try running the file (without recompiling):

   ```
   > java <classname>
   ```

   Note that you are able to essentially move "object files" across OS/machines.

Try to summarize your understanding of the features of the Java language you have observed. How do these compare to C or Python (or C++ if you are familiar with that language)? Do you see any benefits of such features?

Part 2

The source code for this exercise should be submitted by the due date. The mechanism for the submission will be announced soon.

Implement a **C program** that has the following functionality:

A taxi fleet consists of diesel and electric cars. For this exercise, we have data about the fuel efficiency and current fuel (diesel or battery) level of all cars in the fleet. We would like to calculate the range of the car (distance it can cover with the current fuel level).

- For diesel cars, the fuel efficiency of each car is specified in km/l and the amount of fuel remaining in the tank is specified in litres.
- For electric cars, efficiency is specified in W-h/km and the remaining battery charge is specified as a percentage of the total battery energy currently remaining.
- Assume all electric cars have a battery capacity of 100 kW-h, and assume the fuel efficiency is a constant for a given vehicle.

The input data is a file (or standard input) with the following information:

Line 1: integer n – the number of cars in the fleet
Lines 2 to n+1 each contain an integer and two float values as follows:

    The first integer is 1 or 2 indicating if the car is diesel or electric respectively
    The next value, a float, is the fuel efficiency of the current car
    The third value, a float, is the amount of fuel left in the car (diesel) or percentage charge (electric car), as described above.

Write a C program that reads this data, stores relevant information about each car, and then prints out the range for each car, but in the reverse order relative to the input sequence - that is, the information about the last car in the input should be printed out first.

The output should be a series of n lines, each line looking like:
Car i: range = _____

Sample data:

```
5
1 15.0 11.5
2 172.1 35.5
2 150.5 73
1 14.0 14.0
2 160.5 11
```

While you design your solution, you should also consider the following - although they are not strictly needed to solve the above problem:

1. What if the specification changes, and diesel and electric cars have a different number of attributes to define efficiency. For example, if the battery sizes of vehicles are different, and the size of the battery is an additional field in the input for electric cars. How easy would it be to modify your program to accommodate this?
2. If you had to also print out the ranges of the cars in decreasing order of range, what changes would you need to your data structures (if any)