Joanna Klukowska
joannakl@cs.nyu.edu

Project 5: Number Systems

Due date: November 20, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own . Any sharing or copying of assignments will be considered cheating (this includes posting of partial or complete solutions on Piazza, GitHub or any other public forum). If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own). You are responsible for every line in your program: you need to know what it does and why. You should not use any data structures and features of Java that have not been covered in class (or the prerequisite class). If you have doubts whether or not you are allowed to use certain structures, just ask your instructor.

This is a fairly short projects in which you will get a chance to practice your newly aquired skills of writing recursive function. You will design a class that provides a few static methods that perform conversion between different number systems (binary, decimal, hexadecimal).

Objectives

The goal of this programming project is for you to master (or at least get practice on) the following tasks:

- designing and writing recursive function
- using different number systems

Start early! This project may not seem like much coding, but debugging always takes time (especially for recursice functions).

For the purpose of grading, your project should be in the package called `project5`. This means that each of your submitted source code files should start with a line:
```
package project5;
```
Keep in mind that spelling and capitalization are important! The package declaration line has to be the first line in your file!

Background

To complete this project you need to be familiar with the binary, decimal and hexadecimal number systems and how to convert one representation into another.

The binary number system uses two symbols: `0` and `1`. All values are represented using sequences of zeroes and ones, for example: `1000111011`.

The decimal number system uses ten symbols: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`. All values are represented using sequences of these ten symbols.

The hexadecimal number system uses sixteen symbols: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `A`, `B`, `C`, `D`, `E`, `F`. All values are represented using sequences of these sizteen symbols. (The symbols `A`, ..., `F` are used to represent the decimal values of 10, 11, 13, 14, and 15.)

In order to determine the value of any of the sequences written as binary, decimal or hexadecimal, we need to multiply the values of symbols by the powers of the base for that number system (2 for binary, 10 for decimal, 16 for hexadecimal). The powers are determined by the position of the symbol. We count the positions from right to left in the sequence and the counting starts at zero. Examples:

1. Given a binary sequence `11000111011`, its value is

$$1 * 2^1 0 + 1 * 2^9 + 0 * 2^8 + 0 * 2^7 + 0 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 =$$

$$1024 + 512 + 0 + 0 + 0 + 32 + 16 + 8 + 0 + 2 + 1 =$$

$$1595$$

Joanna Klukowska
joannakl@cs.nyu.edu

2. Given a decimal sequence `1595`, its value is

$$1 * 10^3 + 5 * 10^2 + 9 * 10^1 + 5 * 10^0 =$$
$$1000 + 500 + 90 + 5 =$$
$$1595$$

   (Since we tend to think of values of numbers in decimal, the value of the decimal sequence is exactly the same as the sequence itself.)

3. Given a decimal sequence `63B`, its value is

$$6 * 16^2 + 3 * 16^1 + 11 * 10^0 =$$
$$1536 + 48 + 11 =$$
$$1595$$

Conversion Algorithms

Binary to Decimal In order to convert a binary value to decimal, we need to follow the same steps as for calculation of the value shown in the previous paragraph. For each symbol in the binary string, multiply the value of that symbol by the power of two corresponding to the position of the symbol (positions are counted from right to left and start at zero). The sum of those products, gives us the decimal equivalent.

Decimal to Binary To convert a decimal integer to binary sequence:

- calculate the new decimal value by dividing the decimal integer by two (integer division); the remainder from that division is the symbol at position zero in the binary sequence (positions are counted from right to left)
- calculate the next decimal value by dividing the decimal value from the previous step by two (integer division); the remainder from that division is the symbol at position one in the binary sequence
- ...
- continue until the decimal value is equal to zero

Binary to Hexadecimal To convert a binary sequence to a hexadecimal sequence first pad the binary sequence with zeroes so that the sequence length is a multiple of four. Start from righ end and move towards the left end:

- convert the rightmost four symbols to the hexadecimal symbol with the same value
- convert the next four symbols to the hexadecimal symbol with the same value
- ...
- convert the last four symbols to the hexadecimal symbol with the same value

Hexadecimal to Binary To convert a hexadecimal sequence to a binary sequence convert each symbol to a binary sequence consisting of exactly four symbols. Concatenate the binary syquences to form a sequence representing the whole number.

Implementation

The class that you need to implement is the `Converter` class. You can find its documentation at https://cs.nyu.edu/~joannakl/cs102_f18/hwk/Converter/project5/Converter.html. Note that this class should not provide a public constructor since there is no reason to create instances of the class.

The binary and hexadecimal sequences are represented using string with prefixes of ”0b” and ”0x” respectively. The decimal sequences are represented using the `int` type.

Joanna Klukowska
joannakl@cs.nyu.edu

Restrictions

Your implementation is not allowed to use the `parseInt()` method provided by the classes in Java.

All functions in the `Converter` class have to be recursive (or wrappers calling recursive functions).

Programming Rules

You should follow the rules outlined in the document *Code conventions* posted on the course website at https://cs.nyu.edu/~joannakl/cs102_f18/notes/CodeConventions.pdf.

You may use any exception-related classes.

Working on This Assignment

You should start right away!

You should modularize your design so that you can test it regularly. Make sure that at all times you have a working program. You can implement methods that perform one task at a time. This way, if you run out of time, at least parts of your program will be functioning properly.

You should make sure that you are testing the program on much smaller data set for which you can determine the correct output manually. You can create a test input file that contains only a few rows.

You should make sure that your program's results are consistent with what is described in this specification by running the program on carefully designed test inputs and examining the outputs produced to make sure they are correct. The goal in doing this is to try to find the mistakes you have most likely made in your code.

Each class that you submit will be tested by itself without the context of other classes that you are implementing for this assignment. This means that you need to make sure that your methods can perform their tasks correctly even if they are executed in situations that would not arise in the context of this specific program.

You should backup your code after each time you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, do anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even a few hours before the due dates - make sure that you have working code if that happens.

Grading

If your program does not compile or if it crashes (almost) every time it is run, you will get a zero on the assignment. Make sure that you are submitting functioning code, even if it is not a complete implementation.

If the program does not adhere to the specification, the grade will be low and will depend on how easy it is to figure out what the program is doing.

100 points correctness of the four methods that need to be implemented (25 points each method). If the implementation is not recursive or if the built-in Java conversion functions are used, then the given function will receive zero credit. If the function is not documented, 5 points will be deducted from the correctness score for that function.

How and What to Submit

For the purpose of grading, your project should be in the package called `project5`. This means that each of your submitted source code files should start with a line:

```
package project5;
```

Your should submit all your source code files (the ones with .java extensions only) in a single **zip** file to Gradescope.

You can produce a zip file directly from Eclipse:

Joanna Klukowska
joannakl@cs.nyu.edu

- right click on the name of the package (inside the `src` folder) and select Export...
- under General pick Archive File and click Next
- in the window that opens select appropriate files and settings:
  - in the right pane pick ONLY the files that are actually part of the project, but make sure that you select all files that are needed
  - in the left pane, make sure that no other directories are selected
  - click Browse and navigate to a location that you can easily find on your system (Desktop or folder with the course materials or ...)
  - in Options select ”Save in zip format”, ”Compress the contents of the file” and ”Create only selected directories”
- click Finish