# Assignment 2: BabylonianSort

Dr. Sharma Thankachan

COP 3203 – Fall 2018

Due September 19th, 2018

## 1. Deliverable:

A source file named *BabylonianSort.java* written in Java.

## 2. Objective:

The objective of this assignment is to demonstrate an application of radix sort using an exotic number system. It will require that you brush up on skills regarding base conversion, and will touch on the concept of exception handling in Java as well.

## 3. Description:

Numbers can be represented using many different bases besides the base-10 used for “standard” decimal numbers. A few examples include:

base-2 (binary): Represents computer bits and boolean logic.

base-8 (octal): Used in computers for permissions and other settings.

base-16 (hexadecimal): Frequently used to represent color codes succinctly.

These are not the only number systems that exist, nor are they the only systems that are useful. In fact, the ancient Babylonians of Mesopotamia employed a *base-60* number system for daily use. Base-60 is superior when dealing with fractions and is still “hand-countable.” Counting hand bones of the 4 fingers (12 total bones) on one hand and tracking iterations using the fingers of the other hand (5 total fingers) results in a representable range of 5 * 12 = 60. This system is called the sexagesimal number system.

For this assignment, we will be looking at a modified version of the sexagesimal system. The ancient Babylonians obviously did not use Arabic numerals so their system must be *adapted* into Arabic numerals and Latin characters here. What follows is the definition of the sexagesimal system that will be used for this assignment. The top row is the sexagesimal representation of a number, and the bottom row is the decimal representation.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

| k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |

| E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |

## 4. Requirements:

This assignment must be named BabylonianSort.*java* and contain a public class named *BabylonianSort*. Within this public class you must write these 4 methods:

```
public static void babylonianSort(String[] numbers) throws NumberFormatException
```

This method will take an array of strings as input and must sort this array *in place.* Each string in the array represents a sexagesimal number (see section 5 for specifications).

*NOTE:* Sorting an array *in place* means that the function returns nothing, instead the original parameter array is changed.

*NOTE:* You **must** use radix sort to sort the sexagesimal numbers. If you use a different sorting method, you will fail all test cases associated with this function.

*NOTE:* You may **not** convert the sexagesimal numbers to decimal and sort them that way. The numbers passed will be far too large to store as integers or longs anyway, so don't waste time attempting this.

```
public static String decimalToSexagesimal(long number)
```

Return a string representing the sexagesimal version of the given long decimal number. See section 5 for input specifications for this function.

```
public static long sexagesimalToDecimal(String number) throws NumberFormatException,
                                                               ArithmeticException
```

Return a long integer representing the decimal version of the given sexagesimal number stored in the input string. This function must throw the error *NumberFormatException* if the given sexagesimal number is invalid (*e.g.* it contains characters not enumerated in the description of the sexagesimal number system as described in section 3). This function must throw the error *ArithmeticException* if the given number is too large to be converted to a long.

*NOTE:* The contents of the error message for both exceptions do not matter, but they should ideally describe the error that has occurred. Here is an example throw of the *NumberFormatException:*

```
throw new NumberFormatException("This message can say anything");
```

```
public static boolean isValidSexagesimalNumber(String number)
```

This function will return true if a given sexagesimal number string is a valid sexagesimal number, else return false. The number must match the description of sexagesimal numbers given in the description (section 3).

## 5. Input Specification:

Each function for this assignment has a slightly different set of input specifications.

**babylonianSort(String[] numbers):**

This function will receive a list of strings, containing at least one string. Each string will contain at least one character. The strings should represent sexagesimal numbers, however each string may contain any valid ASCII character. Therefore, it is very important that you verify that each string in the array is a valid sexagesimal number, as defined in section 3, before proceeding with the function.

**decimalToSexagesimal(long number):**

This function will be passed a long in the range of [0, *Long.MAX_VALUE*].

**sexagesimalToDecimal(String number):**

A string containing one character or more will be passed to this function. All characters in the string will be valid ASCII characters, so this function should throw the *NumberFormatException* if the given string is not a valid sexagesimal number. The string is of arbitrary length, so it's decimal representation could cause an overflow, even for long integers. If the string cannot be represented as a long integer, throw an *ArithmeticException.*

**isValidSexagesimalNumber(String number):**

This function will receive a valid string with at least one character. The characters in the string can be any valid ASCII characters.

## 6. Testing:

To test, you will want to run the provided test script in a Linux environment. Your assignment will be graded on Eustis, and if your assignment does not work on Eustis then it **does not work.** For the script to function, you will need to set up your directory (folder) to look like this:

```
TestSuite/
        BabylonianSort.java
        BabylonianSortTester.java
        Test.sh
        Inputs/
                Input1.txt
                Input2.txt
                ...
```

To run the test script, use this command:

```
$ bash Test.sh
```

*NOTE*: The "$" character is not part of the command. It is a symbol frequently used to indicate the beginning of a terminal command.

## 7. Grading Criteria:

The grading criteria for this assignment is as follows:

***40%*** *babylonianSort()* unit tests

*25%* *sexagesimalToDecimal()* unit tests

*20%* *decimalToSexagesimal()* unit tests

*15%* *isValidSexagesimalNumber()* unit tests

## 8. Final Notes:

Comments will not be graded generally, however I reserve the right to penalize assignments that include no comments, or whose comments are horrible. In the real world, comments are extremely useful and expected from developers, so get used to writing them. I also reserve the right to penalize assignments with atrocious/inconsistent style (or lack thereof). This does not mean that you will lose points for using one particular style over another—it only means that you need to apply your style consistently.

You are free to discuss concepts and high level ideas about this assignment with your peers, however code sharing and code review is strictly prohibited.

Examples of **acceptable** peer questions about this assignment look like this:

- How do you convert a number in base-10 to base-60 on paper?
- What data structures are most useful when implementing radix sort?
- Can you show me how to use Horner's Rule to calculate the base-10 version of a base-60 number on paper?

Examples of **unacceptable** peer questions about this assignment look like this:

- What does the code look like to convert a number from base-10 to base-60?
- I have an off-by-one error in my radix sort code, can you help me find the bug?
- Can you show me how to code up Horner's Rule?

**You MUST do all of the following:**

- Include your name and NID in a comment at the top of your submission.
- Ensure that your submission compiles and runs on Eustis.
- Name your submission *BabylonianSort.java.*

- Submit the assignment before the deadline.
- Write the assignment in Java.

**If you do not follow the guidelines above you will receive a 0.**