CS 284: Homework Assignment 1
Due: September 24, 11:55pm

1 Assignment Policies

Collaboration Policy. Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. Excerpts of code presented in class can be used.

Your code must include a comment with your name and section .

2 Assignment

Define a class `BinaryNumber` that represents binary numbers and a few simple operations on them, as indicated below. An example of a binary number is

1011

Its *length* is 4. Note that its leftmost digit is the most significant one: it represents the decimal number $1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = 11$. This is called *big-endian* format. Please be sure to use *big-endian* format in your program.

This assignment requests that a number of operations be supported. They are divided into two groups. The first is a set of basic operations, the second is slightly more challenging and addresses addition of binary numbers.

2.1 Basic operations

The following operations should be supported:

- A constructor `BinaryNumber(int length)` for creating a binary number of length `length` and consisting only of zeros.
- A constructor `BinaryNumber(String str)` for creating a binary number given a string. For example, given the string `"1011"`, the corresponding binary number should be created. For this exercise you will have to use some standard String operations. These are listed in the "Hints" section below.

- An operation `int getLength()` for determining the length of a binary number.
- An operation `int[] getInnerArray()` that returns the integer array representing the binary number.
- An operation `int getDigit(int index)` for obtaining a digit of a binary number given an index. The starting index is 0. If the index is out of bounds, then a message should be printed on the screen indicating this fact.
- An operation `int toDecimal()` for transforming a binary number to its decimal notation (cf. the example given above).
- An operation `void bitShift(int direction, int amount)` for shifting all digits in a binary number any number of places to the left or right. The `direction` parameter indicates a left shift when the value is -1. When `direction` is given the value 1, the shift should be to the right. Any other value for `direction` should be seen as invalid. The `amount` parameter specifies how many digits the BinaryNumber will be shifted, and is only valid when it is nonnegative. For example, `'1011'` shifted right by 2 is `'10'`. `'1011'` shifted left by 2 is `'101100'`. Notice that shifting right decreases the number by factors of 2, while shifting left increases the number by factors of 2. These operations are equivalent to the ">>" and "<<" operators in Java.
- An operation `static int[] bwor(BinaryNumber bn1, BinaryNumber bn2)` that computes the bitwise or of the two numbers. Note that both argument BinaryNumbers must be of the same length for the input to be considered valid. The bitwise or of `'1010'` and `'1100'` is `'1110'`.
- An operation `static int[] bwand(BinaryNumber bn1, BinaryNumber bn2)` that computes the bitwise and of the two numbers. Note that both argument BinaryNumbers must be of the same length for the input to be considered valid. The bitwise and of `'1010'` and `'1100'` is `'1000'`.
- An operation `String toString()` that returns the BinaryNumber as the corresponding encoded string.

2.2 Addition of Binary Numbers

Here is an example of how two binary numbers of the same length are added[1].

	1			1		(carried digits)
	0	1	1	0	1	
+	0	1	0	0	1	
=	1	0	1	1	0	= 22

Note that it is possible for the addition of two numbers to yield a result which has a larger length than the summands. In that case, room should be made for the extra digit - meaning the array should be copied over to a new one that is one greater in length.

[1] Source: `https://en.wikipedia.org/wiki/Binary_number`

	1	1	1				(carried digits)
		1	0	1	1	0	
+		1	1	1	0	1	
=	1	1	0	0	1	1	= 51

The `int[]` field `data` should be added to the data fields of `BinaryNumber`.

Implement the following operations:

- `void add(BinaryNumber aBinaryNumber)` for adding two binary numbers, one is the binary number that receives the message and the other is given as a parameter. If the lengths of the two BinaryNumbers do not coincide, then the smaller one should have 0's prepended to it in order to prevent errors. Note how `'101' + '1'` is the same as `'101' + '001'`. The `BinaryNumber` which receives `aBinaryNumber` should be modified with the result of addition.

2.3 Hints

- For the `BinaryNumber(String str)` constructor, the following operations might come in handy:
  - `char java.lang.String.charAt(int index)`, which returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.
  - `int java.lang.Character.getNumericValue(char ch)`, which returns the int value that the specified Unicode character represents.
- For methods where allocating more space is necessary, it may be useful to define a `static void prepend(int amount)` method, that preprends `amount` 0's to the BinaryNumber.

3 Submission instructions

Submit a single file named `BinaryNumber.java` through Canvas. No report is required. Your grade will be determined as follows:

- You will get 0 if your code does not compile.
- The code must implement the following UML diagram precisely (see below).
- We will try to feed erroneous and inconsistent inputs to all methods. All arguments should be checked.
- Partial credit may be given for style, comments and readability.

BinaryNumber
private int data[] private int length
public BinaryNumber(int length) public BinaryNumber(String str) public int getLength() public int getDigit(int index) public int[] getInnerArray() public static int[] bwor(BinaryNumber bn1, BinaryNumber bn2) public static int[] bwand(BinaryNumber bn1, BinaryNumber bn2) public void bitShift(int direction, int amount) public void add(BinaryNumber aBinaryNumber) public String toString() public int toDecimal()