Chulalongkorn University
International School of Engineering
Department of Computer Engineering
2140105 Computer Programming Lab.

Name ______________________
Student ID __________________
Station No. __________________
Date ________________________

Lab 12 - Review

Objectives:

- Learn to use Eclipse as Java integrated development environment.
- Practice basic java programming and object-oriented programming.
- Review for final examination.

Binary Number

The *binary numeral system*, or base-2 number system, is a numeral system that represents numeric values using two symbols, usually 0 and 1.

Representation

A binary number can be represented by any sequence of bits (binary digits), which in turn may be represented by any mechanism capable of being in two mutually exclusive states. The following sequences of symbols could all be interpreted as the same binary numeric value:

```
1 0 1 0 0 1 1 0 1 0
| - | - - | | - | -
x o x o o x x o x o
y n y n n y y n y n
```

Binary numbers are commonly written using the symbols 0 and 1.

Bitwise operations

AND

A bitwise AND takes two binary representations of equal length and performs the logical AND operation on each pair of corresponding bits. In each pair, the result is 1 if the first bit is 1 AND the second bit is 1. Otherwise, the result is 0. For example:

```
    0101
AND 0011
  = 0001
```

Right shift

In a right shift, the bits are moved to the right, and zero(s) are shifted in (on left end). The LSB (lease significant bit) is discarded as shown in Figure 1.

Left shift

In a left shift, the bits are moved to the left, and zero(s) are shifted in (on right end). The MSB (most significant bit) is discarded as shown in Figure 2.

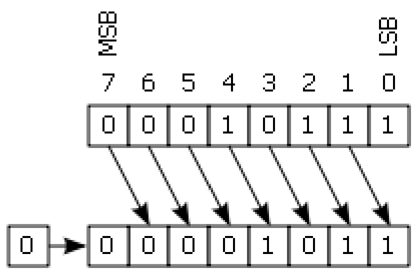

Figure 1 Right shift

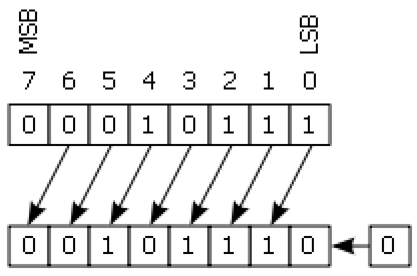

Figure 2 Left shift

Class `BinaryNumber` which represents the binary numeral system described above uses array of boolean, which its size is divisible by eight (8), to represent each bit. The LSB (least significant bit) is the element 0 and the MSB (most significant bit) is the element `bits.length() – 1`.

Questions

1. Create a new Java project using your student ID as the project name.
2. Setup the project to be ready for JUnit.
3. Import `finalsrc.jar` into your project.
4. Complete `BinaryNumber.java` which represents the binary numeral system as described by the following specifications: (you must use the provided unit test cases, `BinaryNumberTest.java` to test your class. DO NOT modify `BinaryNumberTest.java`.) You will get one mark for each test case that passes.
   4.1. `decimalToBinaryString`: converts an integer decimal argument to the string representation of the unsigned integer value represented by the argument in binary (base 2). You are NOT ALLOWED to call `Integer.toBinaryString()`, but you can use this method to check your result.

> To convert from a base-10 integer numeral to its base-2 (binary) equivalent, the number is divided by two, and the remainder is the least-significant bit. The (integer) result is again divided by two, its remainder is the next most significant bit. This process repeats until the result of further division becomes zero.
>
> For example, 118_{10}, in binary, is:
>
> ```
>   Operation          Remainder
>  118 ÷ 2 = 59        0
>   59 ÷ 2 = 29        1
>   29 ÷ 2 = 14        1
>   14 ÷ 2 = 7         0
>    7 ÷ 2 = 3         1
>    3 ÷ 2 = 1         1
>    1 ÷ 2 = 0         1
> ```
>
> Reading the sequence of remainders from the bottom up gives the binary numeral $\mathbf{1110110}_2$.

   4.2. Constructor with an integer argument: constructs a newly allocated `BinaryNumber` that represents the binary number equivalent to the integer parameter. In allocating the bits array, the size of the arrays is equal to the length of the String resulting from method `decimalToBinaryString(decValue)`. The bit which has binary value of 1 has the equivalent boolean `true`, and the bit which has binary value of 0 has the equivalent boolean `false`.

For example, new BinaryNumber(118) will do the following:

- Allocates the array of boolean, `bits`, to the size of the length of the binary string, which is 7 (as shown in question 4.1).
- For each bit, set the value of the array element to `false` if the bit's value is 0 and to `true` if the bit's value is 1. In this example, the array bits will contain:

(MSB)[`true, true, true, false, true, true, false`](LSB).

4.3. `value`: returns the value of this integer in base-10 (decimal).

> To convert from a base-2 integer numeral to its base-10 (decimal) equivalent, each bit's value is multiplied by it's position value, 2^0, 2^1, 2^2, …, for bit 0, 1, 2, and so on.
>
> For example, **01110110**$_2$, in decimal, is:
>
> $\mathbf{0}(2^0) + \mathbf{1}(2^1) + \mathbf{1}(2^2) + \mathbf{0}(2^3) + \mathbf{1}(2^4) + \mathbf{1}(2^5) + \mathbf{1}(2^6) + 0(2^7)$
>
> $= 0 + 2 + 4 + 0 + 16 + 32 + 64 + 0$
>
> $= 118_{10}$

4.4. `add`: returns a new `BinaryNumber` which is the result of addition with another `BinaryNumber`. You can use any method that is written by you, but cannot use Java's library.

4.5. `shiftLeft`: returns a new `BinaryNumber` which is the result of bitwise left shift as described above and shown in Figure 4.

4.6. `and`: return a new `BinaryNumber` which is the result of bitwise **AND** with another `BinaryNumber`.

4.7. `compareTo`: compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

5. Create a new class called `SignedBinaryNumber` which extends from `BinaryNumber`. You must do the following: (each test case is equal 1 mark)

5.1. add a `boolean` field called `positive` which is `true` if the binary number is positive and `false` if the number is negative.

5.2. add a constructor with an integer decimal argument. This constructor will initialize the array bits to the absolute value of the argument, and set the field `positive` to appropriate value. Test case must be created to test this constructor, at least, check for a positive and a negative value.

5.3. override `value()` to return negative integer if the binary number is negative and return positive integer otherwise. Test case must be created to test this constructor, at least, check for a positive and a negative value.

5.4. override `compareTo()` so result from comparing the positive, negative binary numbers is similar to comparing two signed integers. Test case must be created to test the combination of input value to get all possible results.

6. Export all files to an archive file with the name, ***yourID.zip*** by follow the instruction below:
   6.1. In the package explorer view, right-click the project name
   6.2. Select **Export…**
   6.3. Select **JAR File**, click **Next**
   6.4. Select export java source file and resources.
   6.5. In the “Select the export destination:” box, type ***C:\Temp\49xxxxxxxx.zip*** and click **Finish**.