Institute of Technology, University of Washington Tacoma
TCSS 342 Data Structures
Assignment 1
Decimal to Binary conversion using an array-based Stack implementation
Value: 5% of the course grade
Due: Thursday, 17 January 2019, 23:59:59

Purpose:

This program has a number of purposes. It is intended to provide practice with the following:

- setting up and using an SVN repository
- development of an interactive program
- implementation of an algorithm that is described in pseudocode
- solving a problem using a Stack
- writing unit tests

This project also serves to exercise your ability to use Eclipse and plugin tools to support the development of high quality code that adheres to conventions.

Program Description:

Write an interactive program which accepts a positive integer value from a user and then displays the binary equivalent of that value. The program should allow the user to repeat this process until the user chooses to quit.

Your program may be a console application or a GUI application, whichever you prefer.

Your project must include a package called '`program`'.
Inside the '`program`' package you must create a class called '`StackUtilities`'.
Inside the '`StackUtilities`' class you must provide a public static method called '`decimalToBinary`' which accepts an `int` parameter and returns a `String` representation of the binary equivalent of the `int` parameter.

For example, if the number 13 is the parameter, then the method should return the String "1101". Your method should perform the conversion by implementing the algorithm shown in pseudocode below:

1. read a number 'n'
2. while 'n' > 0
    1. 'remainder' = n % 2
    2. push 'remainder' onto a stack
    3. n = n / 2
3. while the stack is not empty
    Pop an element and add it to the output String

NOTE: There are many other ways to convert decimal numbers to binary numbers; however, any solution which does not use a Stack will receive a VERY low grade. Use the ArrayStack class provided in the starter code.

Provide unit test cases for the `decimalToBinary` method. You are not required to write unit tests for any other part of your program. The code you write (program code and tests) should generally adhere to course coding conventions and should be fully documented with Javadoc comments.

Getting Started:

Create your Eclipse project by downloading the assignment1.zip file from the Assignment 1 page on Canvas, importing it into your workspace, and using "Refactor" to change "username" in the project name to your UWNetID. Incorrectly-named projects *will lose points.* Be sure to rename your project BEFORE sharing it to your SVN repository.

NOTE:

The Eclipse plugin tools generate a few warnings about the provided starter code. You are not required to silence these warnings. You can safely ignore the warnings in the starter code.

Documentation within the code:

Include complete javadoc comments in your program. That is, include a javadoc comment for each interface, class, field, constant, and method including those declared private. Refer to How to Write Doc Comments for the Javadoc Tool - http://java.sun.com/j2se/javadoc/writingdoccomments/index.html. Try to emulate the style and quality of the javadoc comments in the Java API.

Submission and Grading:

When you have checked in the revision of your code you wish to submit, make a note of its Subversion revision number. To get the revision number, *perform an update* on the top level of your project; the revision number will then be displayed next to the project name. Include your project's revision number in your executive summary.

Part of your program's score will come from its "external correctness." I will run your program and enter several inputs and check for correct outputs.

Another part of your program's score will come from its "internal correctness." Internal correctness includes meaningful and systematically assigned identifier names, proper encapsulation, avoidance of redundancy, the use of javadoc on all class members, the use of non-javadoc implementation comments on particularly complex code sections, and adherence to coding conventions as identified by the output of the Eclipse plugin tools.

The assignment will be graded using the following breakdown: 10% executive summary, 60% external correctness (as determined by my tests run on your code), 30% internal correctness (code quality), including your unit tests.