CIS 435/535 Computer Networks and System Security

Project #1: Cryptography

Be sure to review the university policy on academic Plagiarism and Cheating.
This is an individual project.

Project Objective

The purpose of this project is to design, implement, and thoroughly test a Cryptography facility, following the Test-driven development process. The Cryptography facility will be used to secure message communication in future Phases.

You should NOT use any existing crypto liberties, not even the ones that are part of the standard Java library. Violation of this requirement will result in **"0"** grade in this project.

Project Requirements

Phase #1: Design, Implement, and Test a Cryptography facility that includes all the cryptography algorithms that we've discussed in class. To specific, the secure facility should include the following cryptography algorithms

1. Shift cipher
2. Substitution cipher
3. Polyalphabetic cipher
4. RSA
5. Block cipher
6. CBC (Cipher Block Chaining)
7. MAC (Message Authentication Code)
8. Digital Signatures
9. CA (Certification Authorities)

You will need to use `BigInteger` in this phase. Here is a link on `BigInteger`
https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html

Test-Driven Development

In reference [1], Test-driven development is defined as a process that relies on the repetition of a very short development cycle. It is based on the test-driven concept of **extreme programming** (**XP**) that encourages simple design with a high level of confidence. The procedure that drives this cycle is called **red-green-refactor**. The procedure itself is simple and it consists of a few steps that are repeated over and over again:

1. Write a test.
2. Run all tests.
3. Write the implementation code.
4. Run all tests.
5. Refactor.
6. Run all tests.

Below are some links on Test-Driven Development for your reference.

1. http://books.tarsoit.com/Test-Driven%20Java%20Development.pdf
2. https://technologyconversations.com/2014/09/30/test-driven-development-tdd/

Below is an example for Shift Cipher Design, Implementation, and Tests. Steps 1 to 3 should be included in the Proposal of Design for every single algorithm. Namely, your proposal should include at least 9 tables.

ShiftCipher:		
1	Design a Test Case	Given: input BigInteger m = “100”, shiftKey s = “5” ShiftCipher.encrypt(m, s): output c = “105” ShiftCipher.decrypt(c, s): output m =“100”

2	Write a test code	
		```
		public void testEncrypt() {
		    step++;
		    System.out.println("Step #" + step + " -- testEncrypt()");
		    BigInteger m = new BigInteger("100");
		    BigInteger shiftKey  = new BigInteger("5");
		    BigInteger expectedResult = new BigInteger("105");
		    BigInteger result = ShiftCipher.Encrypt(m, shiftKey);
		    assertEquals(expectedResult, result);
		    System.out.println("\t-- success!\n");

		  }
		```

3	Define methods Signature/Interface	“Describe, in general, the functionality of EACH method with specific examples on input, expected result/output”
		
		Public BigInteger encrypt(BigInteger m, BigInteger shift);
		
		Public BigInteger decrypt(BigInteger m, BigInteger shift);

4	Write the implementation code	
5	Run tests with different test cases and refactor implementation if needed.	
6	Refactor and Repeat all steps to thoroughly test class	

Proposal Submission

1. Submit a PDF of your project proposal on Sakai
2. Demo your proposal to Peer Students assigned by Prof. Wang and get their approvals. You are responsible of making your proposal clear and sound to Peer Students (Mock Customers).
3. You may be asked to demo the proposal to Prof. Wang or the TA.

Recommendations

In this project and subsequent projects, use a Java unit tester junit4 to thoroughly test your code. Here is a tutorial on using junit in NetBeans https://netbeans.org/kb/docs/java/junit-intro.html

- Start the project Today.
- Use descriptive names for classes, methods, and variables
- Organize your code accordingl to the general "Java Language Coding Guidelines" available at http://www.horstmann.com/bigj2/style.html
- Implement the classes following an easy to difficult order and thoroughly Test each class/method individually and before moving on to the next one.
- Be sure you test ALL methods *thoroughly*, both in isolation and in interleaved fashion.

Submitting

1. Header Comments

Your program must use the following standard comment at the top of *each source code file for which you wrote code*. Copy and paste this comment and modify it accordingly.

```
/**
 * Write a succinct, meaningful description of the class here.
 * You should avoid wordiness and redundancy. If necessary,
 * additional paragraphs should be preceded by <p>,
 *
 * <p>Cryptography  (Describe, in general, the code contained.)
 *
 * @author <your name>
 * @date   <date of completion>
 */
```

2. Inline Comments

Please comment your code with a *reasonable amount of comments* throughout the program. Each function should have a comment that includes information about input, output, overall operation of the function, as well as any limitations that might raise exceptions. Each *block* of code (3-4 or more lines in sequence) in a function should be commented.

It is ***prohibited*** to use *long* comments to the right of lines of source code; attempt 80 character-wide text in source code files.

3. *Proof of Program Execution*

Execute your code and take a screenshot of the associated output window. Place these *labeled* screenshots into a word processing document (Word, OpenOffice, GoogleDocs, etc.) in an ordered manner (an example is provided at the end of this document). Please make sure to crop and enlarge the screenshots so that the picture and / or text is clear. See Figure 1 next page.

Last, create a PDF of this document and call it evidence.pdf.

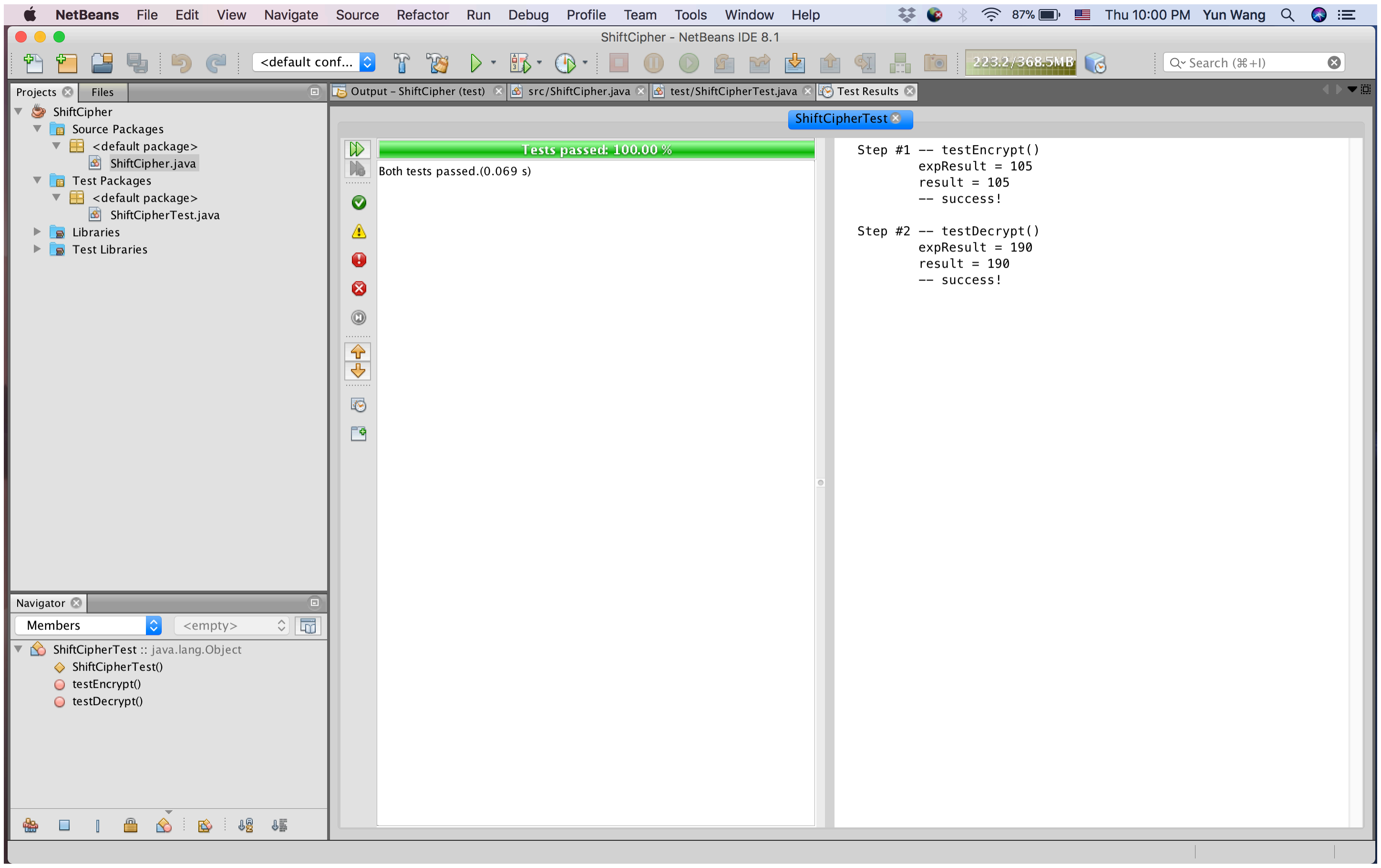
4. *Source Code Files*

Place all source files (only *.java) *containing code you implemented* into a single folder name src. This folder should not contain any subfolders or extraneous files.

5. *Final Submission File*

In a folder named **project1** place (1) folder src, and (2) evidence.pdf. Zip folder **project1** and label that zip file as **project1.zip**.

Submit your zip file via Sakai under Assignments.



```
Tests passed: 100.00 %
Both tests passed.(0.069 s)
```

```
Step #1 -- testEncrypt()
        expResult = 105
        result = 105
        -- success!

Step #2 -- testDecrypt()
        expResult = 190
        result = 190
        -- success!
```