# Chapter 05 - Buggy

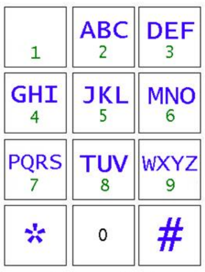
This homework assignment is meant to make sure you can **debug** Java programs using IDE and its debugger. **<u>This is a group assignment</u>**; you may not share code with other groups of students. This assignment provides an introduction to some of the major features of debugger (from any IDE) by leading you through the debugging process of a slightly buggy program. A debugger allows the programmer to stop a program in the middle of its execution so that he or she may observe how the program branches and executes. Debuggers can help speed the process of isolating and correcting bugs.

## Chapter 05 Buggy Homework - Due 11:55 PM Saturday, March 29, 2014

Read the chapter 05 and other chapters during debugging as needed. Read this hand-out thoroughly.

1. Get a copy of the program **Buggy.java** from Piazza. Add it into your IDE environment (Eclipse), and follow the instructions below. You may need to refactor your class name into different one as requested such as **Ch05_StudentID_MyName.**
2. Post your source file(**.java**) and answer file(**.doc or .hwp**) in Piazza.
   Refer to the checklist and how to submit at the end of this file.

## Background

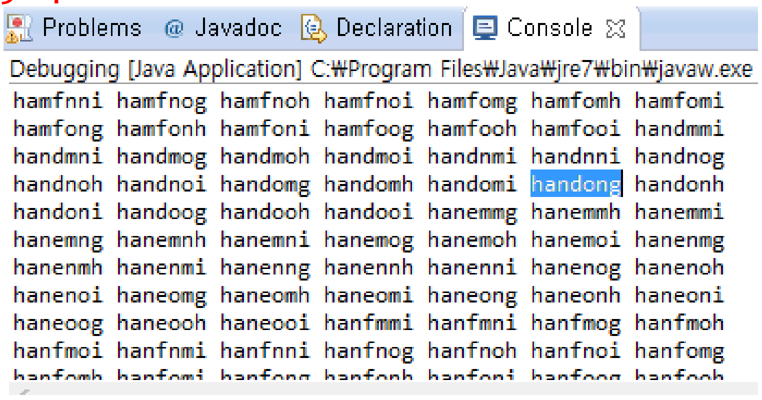


A Mobile Telephone keypad

Follow the instructions outlined in the assignment in order to debug the provided program and answer the questions throughout the assignment on a separate piece of paper. You will be required to present **<u>the corrected code and answers to the questions</u>** in this assignment. The questions in this assignment are meant to be thought-provoking. They illustrate important aspects of the Java language. If you experience problems, do not hesitate to post them on **Piazza** for help!

The provided program is to turn regular phone numbers into "words." For example, we may have a number 1-800-**426-3664** that translated into 1-800-**HANDONG**. (When you try **356-9377** with **five** consonants, you will find "**flowers**", **232-6459** with **four** consonants "**afamily**".) The program, when working properly, will provide all possible "words" that have no more than a user-specified number of consonants. Below is a transcript of how the program should work, with user input appearing in boldface:

> This program finds words that can be associated with phone numbers.
> What is the maximum number of consonants you would like in the generated
> Word to possess?(note: 0's and 1's are considered consonants)
> **5**
> Please enter a 7 digit phone number with dashes removed
> **4263664**



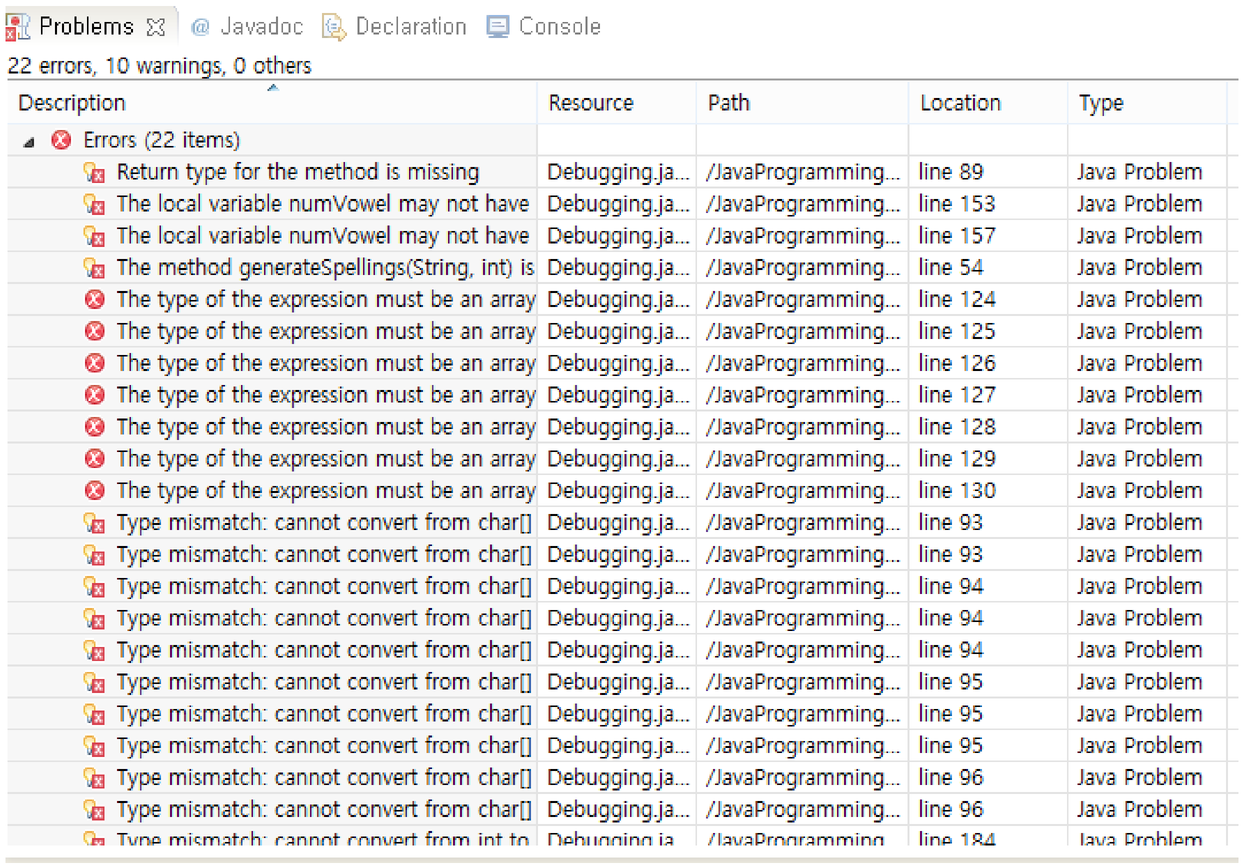
```
Debugging [Java Application] C:₩Program Files₩Java₩jre7₩bin₩javaw.exe
hamfnni hamfnog hamfnoh hamfnoi hamfomg hamfomh hamfomi
hamfong hamfonh hamfoni hamfoog hamfooh hamfooi handmmi
handmni handmog handmoh handmoi handnmi handnni handnog
handnoh handnoi handomg handomh handomi handong handonh
handoni handoog handooh handooi hanemmg hanemmh hanemmi
hanemng hanemnh hanemni hanemog hanemoh hanemoi hanenmg
hanenmh hanenmi hanenng hanennh hanenni hanenog hanenoh
hanenoi haneomg haneomh haneomi haneong haneonh haneoni
haneoog haneooh haneooi hanfmmi hanfmni hanfmog hanfmoh
hanfmoi hanfnmi hanfnni hanfnog hanfnoh hanfnoi hanfomg
hanfomh hanfomi hanfong hanfonh hanfoni hanfoog hanfooh
```

> Would you like to try another number? (Y/N)
> **n**

Before continuing on with this assignment, understand how the program is supposed to work.

## Assignment

The first step in the debugging process is the compilation of the program. In the process of compiling code, useful error messages are provided that identify problematic parts of the code that are not syntactically correct. Very often, these mistakes are typos. You should see something similar to the following (your output may not be the same anyway…):



22 errors, 10 warnings, 0 others

| Description | Resource | Path | Location | Type |
|---|---|---|---|---|
| Errors (22 items) | | | | |
| Return type for the method is missing | Debugging.ja... | /JavaProgramming... | line 89 | Java Problem |
| The local variable numVowel may not have | Debugging.ja... | /JavaProgramming... | line 153 | Java Problem |
| The local variable numVowel may not have | Debugging.ja... | /JavaProgramming... | line 157 | Java Problem |
| The method generateSpellings(String, int) is | Debugging.ja... | /JavaProgramming... | line 54 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 124 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 125 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 126 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 127 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 128 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 129 | Java Problem |
| The type of the expression must be an array | Debugging.ja... | /JavaProgramming... | line 130 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 93 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 93 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 94 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 94 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 94 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 95 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 95 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 95 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 96 | Java Problem |
| Type mismatch: cannot convert from char[] | Debugging.ja... | /JavaProgramming... | line 96 | Java Problem |
| Type mismatch: cannot convert from int to | Debugging.ja | /JavaProgramming | line 184 | Java Problem |

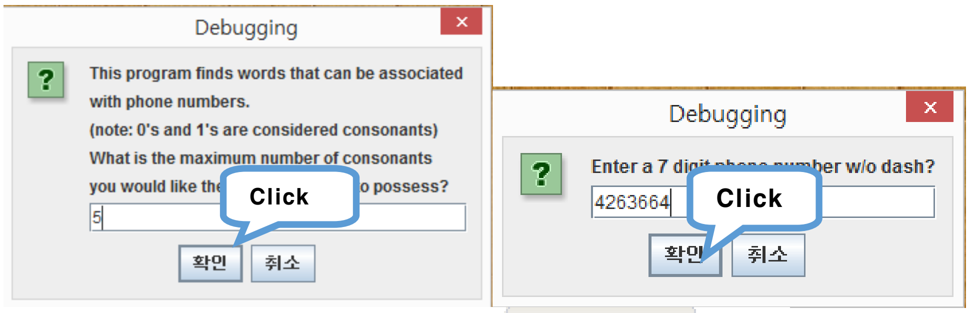
The numbers after the filename refer to the numbers of lines that contain errors. It is important to resolve the **first** error message (in red) first. "Parse errors" typically signify something that is syntactically incorrect. "Warnings" are issued for code that the C compiler believes to be problematic but that is syntactically correct. Warnings will not prevent your code from compiling, but they can tip you off to some rather serious problems.

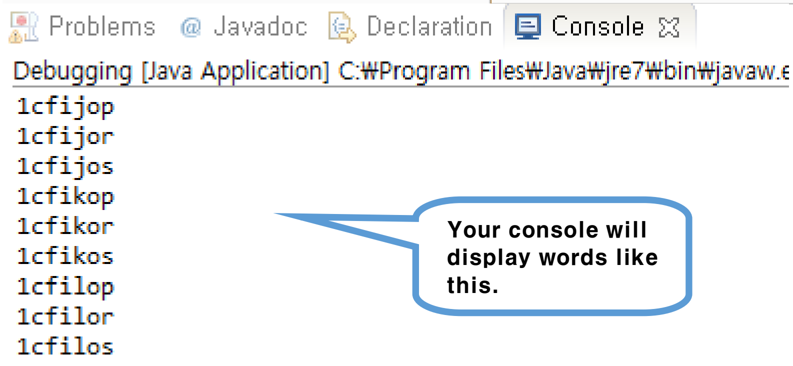
1. Correct the compile errors. Make a note of what changes you made. You may need to make small changes and recompile to see if those errors are fixed. This may take several iterations. It is okay not to fix all the warnings but all of the non-warning problems will need to be fixed.

   Your program has compiled successfully. Don't try to figure out what's wrong with this line yet. We'll figure it out eventually.

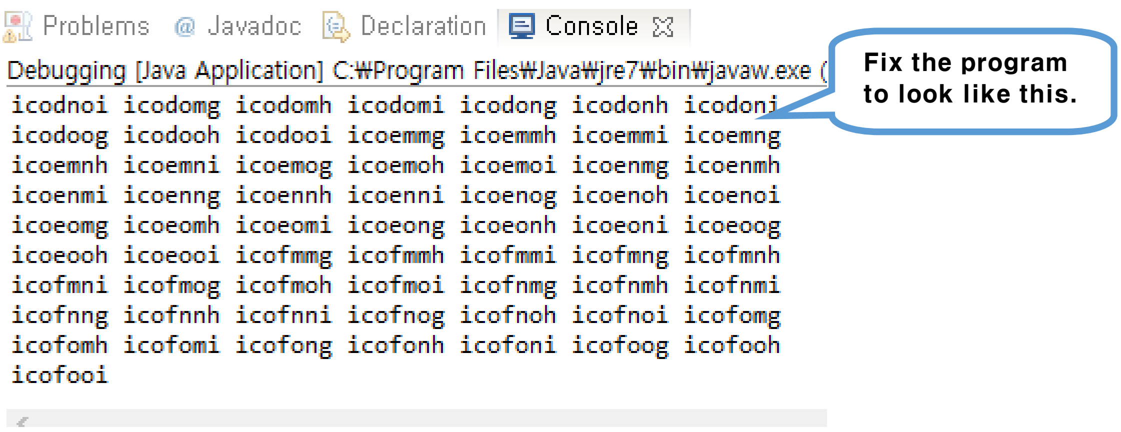
Let's see if the program works. To run it

First Click Run menu → Run (Ctrl+F11) and you will see the following Panel


Debugging
This program finds words that can be associated with phone numbers.
(note: 0's and 1's are considered consonants)
What is the maximum number of consonants you would like the words to possess?
5
Click
확인
취소
Debugging
Enter a 7 digit phone number w/o dash?
4263664
Click
확인
취소



```
Debugging [Java Application] C:₩Program Files₩Java₩jre7₩bin₩javaw.e
1cfijop
1cfijor
1cfijos
1cfikop
1cfikor
1cfikos
1cfilop
1cfilor
1cfilos
```



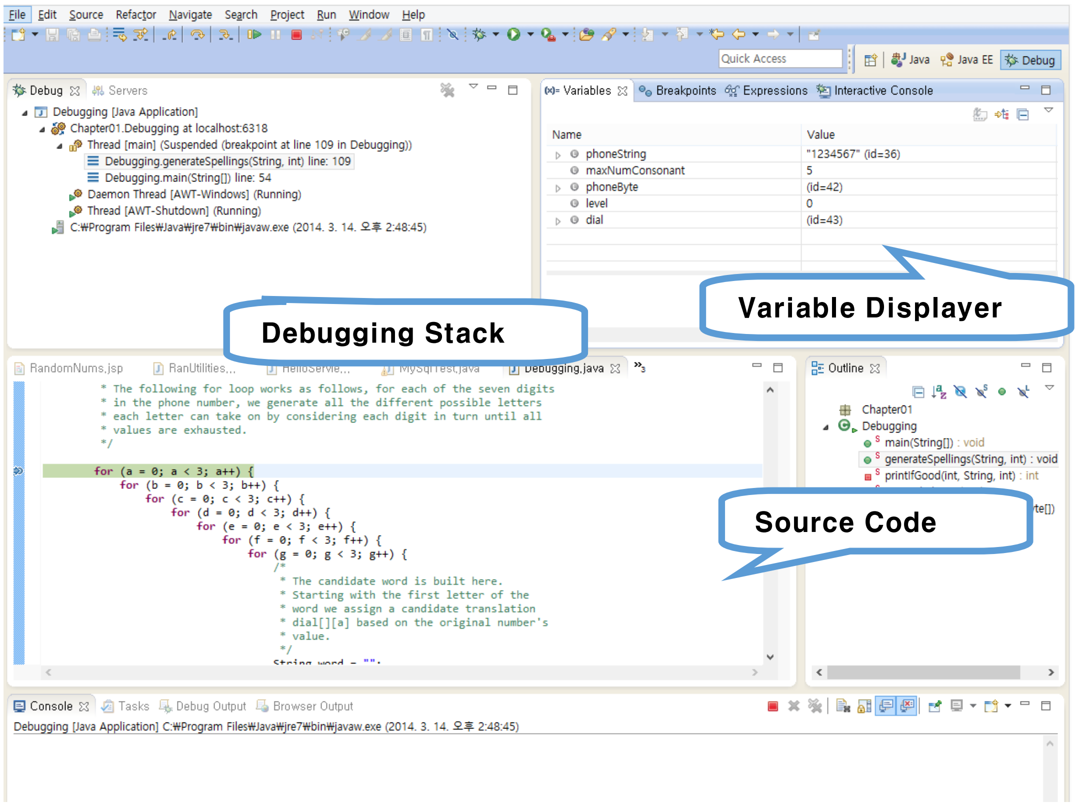
```
Debugging [Java Application] C:₩Program Files₩Java₩jre7₩bin₩javaw.exe (
icodnoi icodomg icodomh icodomi icodong icodonh icodoni
icodoog icodooh icodooi icoemmg icoemmh icoemmi icoemng
icoemnh icoemni icoemog icoemoh icoemoi icoenmg icoenmh
icoenmi icoenng icoennh icoenni icoenog icoenoh icoenoi
icoeomg icoeomh icoeomi icoeong icoeonh icoeoni icoeoog
icoeooh icoeooi icofmmg icofmmh icofmmi icofmng icofmnh
icofmni icofmog icofmoh icofmoi icofnmg icofnmh icofnmi
icofnng icofnnh icofnni icofnog icofnoh icofnoi icofomg
icofomh icofomi icofong icofonh icofoni icofoog icofooh
icofooi
```

**Now, let's start debugging this program.**

Before we begin running the program, we will need to set a **breakpoint**. A breakpoint is a programmer-specified point in the code where we would like the program to "pause" at while the program is running. Why would we want to do this? Using a debugger, we may examine variables in this paused state which will give us important information about whether or not the program is working correctly.

You may set the breakpoint at the beginning of the program when you don't know where the program is wrong, or where to start. The program will run and stop where the first break point is. Then on, you use different controls such as step-in step-over etc. To set a breakpoint in Eclipse, you **click the source code line** to highlight and **<Ctrl + Shift + B>** or you may click a left side of source code window frame (it may depend on your IDE). The breakpoint toggles on/off.

Then select **Run → Debug (F11)** to start and click "Yes" when the Confirm Perspective Switch Panel pops up. Once it stops at the breakpoint, you may start using <step into> <step return> <step over> etc.



```
Debugging [Java Application]
  Chapter01.Debugging at localhost:6318
    Thread [main] (Suspended (breakpoint at line 109 in Debugging))
      Debugging.generateSpellings(String, int) line: 109
      Debugging.main(String[]) line: 54
    Daemon Thread [AWT-Windows] (Running)
    Thread [AWT-Shutdown] (Running)
  C:₩Program Files₩Java₩jre7₩bin₩javaw.exe (2014. 3. 14. 오후 2:48:45)
```

| Name | Value |
|---|---|
| phoneString | "1234567" (id=36) |
| maxNumConsonant | 5 |
| phoneByte | (id=42) |
| level | 0 |
| dial | (id=43) |

```
         * The following for loop works as follows, for each of the seven digits
         * in the phone number, we generate all the different possible letters
         * each letter can take on by considering each digit in turn until all
         * values are exhausted.
         */

        for (a = 0; a < 3; a++) {
            for (b = 0; b < 3; b++) {
                for (c = 0; c < 3; c++) {
                    for (d = 0; d < 3; d++) {
                        for (e = 0; e < 3; e++) {
                            for (f = 0; f < 3; f++) {
                                for (g = 0; g < 3; g++) {
                                    /*
                                     * The candidate word is built here.
                                     * Starting with the first letter of the
                                     * word we assign a candidate translation
                                     * dial[][a] based on the original number's
                                     * value.
                                     */
                                    String word = "";
```

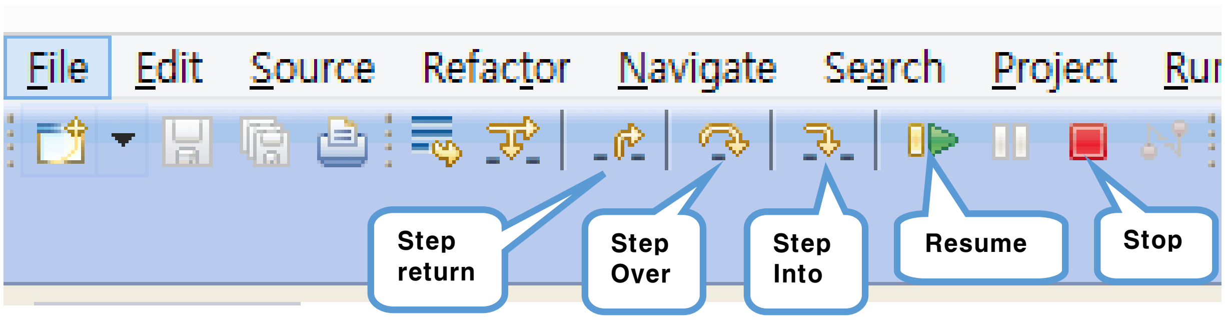
```
Chapter01
  Debugging
    main(String[]) : void
    generateSpellings(String, int) : void
    printIfGood(int, String, int) : int
```

```
Debugging [Java Application] C:₩Program Files₩Java₩jre7₩bin₩javaw.exe (2014. 3. 14. 오후 2:48:45)
```

## ** If (You See Difference windows)

Click Window menu → Reset Perspective…

It will reset the perspective to default view



- STEP RETURN **(F7)** : Steps out of the function of current context.
- STEP OVER **(F6)** : Steps over the function to the next statement.
- STEP INTO **(F5)** : Steps into the function call to inspect inside of the function.
- RESUME **(F8)** : Resume the program.
- STOP **(Ctrl + F2)** : terminate the debugging process.

Continue executing the code **line by line** (use <step over> usually, if you come across a strange code sections, use <step out> to get out of the pit.) until the following is reached.

generateSpellings (phoneString, maxNumConsonant);
generateSpellings is a defined function within this program. We may be interested in watching what happens within the generateSpellings function. If we select <step over> at this point, we will go to the next instruction instead of going inside the generateSpellings function. In order to go inside generateSpellings, we can use the <step into command>

Now, look at the "**Variables**" window again. You may see another new world display all about generateSpelling stuff.

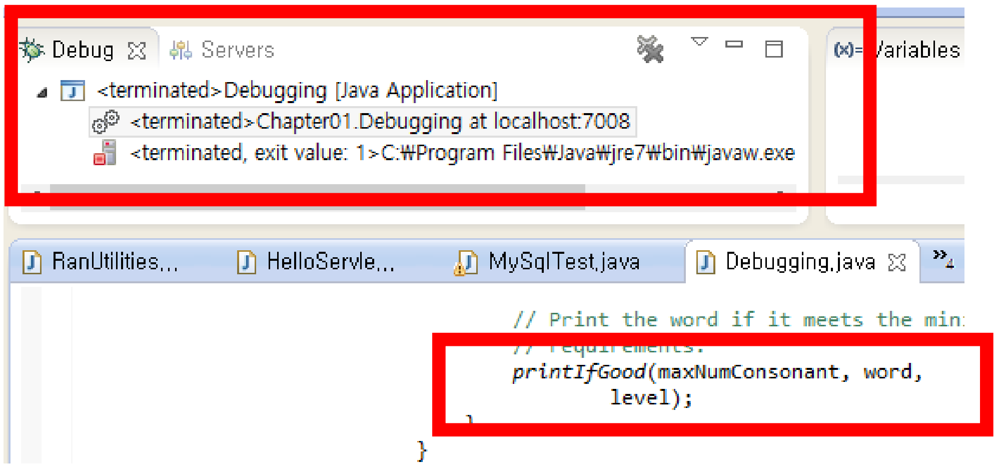
2. Using the techniques you have learned so far,
   Explain what phoneByte[i] = (byte) (phoneString.charAt(i) - '0') does.

Use <step into> to reach convertStringToByte and use <step over> to find out what it is.

3. After calling convertStringToByte method, explain what values are in phoneByte variable.

Look at "Variables" View to find out what values are in phoneByte.

4. Capture a screen shot of Debug window, after calling printIfGood method and explain it.
   What are the argument values when printIfGood() is invoked at the very first time with user's inputs of (5 consonants and 4263664 handong)?



```
<terminated>Debugging [Java Application]
  <terminated>Chapter01.Debugging at localhost:7008
  <terminated, exit value: 1>C:₩Program Files₩Java₩jre7₩bin₩javaw.exe
```

```
        // Print the word if it meets the min
        // requirements.
        printIfGood(maxNumConsonant, word,
                level);
    }
}
```

5. Starting on the above line, a series of nested loops is encountered. How does each nested loop work? Make sure you step through the loop and demonstrate that you've actually examined how the loop works.

6. Starting on the above line, a series of nested loops is encountered. How does each nested loop work? Make sure you step through the loop and demonstrate that you've actually examined how the loop works.
7. What is role of printIfGood method?
   What is the very first ***word*** that printIfGood() prints with user's inputs of (5 consonants and 4263664 handong)?
8. What is the role of level variable?
9. Why the printIfGood method is returning the value of level variable?
10. Finally, explain what the problem is in this section of code. Fix the problem in the code, and recompile the program.

## Checklist – 1.5 points for debugging and comments, 2.0 points for the report

A satisfactory grade on this assignment requires the following:

1. Even though this is a group assignment, each one of you is responsible for submitting the result as directed below. In your files (report and source files), clearly identify yourself as well as your partner.
2. In your report, ***include questions** and answers* to boxed question 1 through 10 in a homework/project report format. You may not have Section 1 ~ 6 described in "HowToWriteReprot" a since you have not programmed this code much, but you should refer to "HowToWriteReport".
   Include a section called ***"Group work:"*** that describes what you did and what your partner did for this assignment, how you contributed or get helped each other. Also include what surprised you the most about this homework. This section is the only part of the report that could be different from that of your partner.
3. Submit **Ch05_StudentID_Name.java** debugged and the questions and answers file called **Ch05_StudentID_Name.doc or .hwp file.**
4. Post your source file and answer file into **hw5** folder in Piazza board. (Make sure that you post the files such that all instructors can see them all, but not to the public or students. And **Make sure that you stick to the submission Guide Line posted on Piazza.**)
5. Each source code must be documented properly that includes author(s), date, description, parameters/input/output and example(s). Since you debugged this program, add your comments where necessary and the change history accordingly.
6. **Note**: a significant portion of the grade for this homework depends on what you write in response to those questions.

## Submitting or posting your solution

- Make sure your code **compiles** and **runs** right before you submit it. Every semester, we get dozens of submissions that don't even compile. Don't make "a tiny last-minute change" and assume your code still compiles. You will not receive sympathy for code that "almost" works.
- If you only manage to work out the homework problem partially before the deadline, you still need to turn it in. However, don't turn it in if it does not compile and run.
- After submitting, if you realize one of your programs is flawed, you may edit or modify your posting as long as it is **before the deadline**. You will have to resubmit any related files together, even if you only change one. You may modify your posting as often as you like. **Only the last version** you submit before the deadline will be graded. **Never modify/edit your post after its deadline.** Then yours will **not** be graded.(period)