Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC 205.01 Fundamentals of CS II 20 Pts Fall 2012**

**LAB 1**

**Objectives: Revision of OOP terminology. Documentation standards, Software design: Use Cases, CRC Cards, Code Skeletons, Test Plan Preparation, Coding and Program Testing, Learning to use Eclipse IDE.**

**COMMON JAVA PROGRAMMING ERRORS**

**1. Using a variable before it is given a value**

This is a common error found in both object-oriented and procedural languages. In Java, primitive variables must be initialized to zero or some default value so there will be no doubt as to what is stored in that variable.

For Example, the following code:

**int x;   
x = x + 1;   
System.out.println("X = " + x);**

 This would have unpredictable results (the value of x is some unknown number)

To fix the problem, initialize the value stored in x to a known value (like 0) ...

**int x = 0;   
x = x + 1;   
System.out.println("X = " + x);**

**2. Misplaced Semi-colon (usually with a loop or if statement)**

This is a common error that is done by beginning Java programmers.

For example:

**for (i=1; i<=10; i++) ; {   
  System.out.println("Number is " + i);   
}**

**if ( x > y) ;  {   
  System.out.println("X is bigger");   
}**

**3. Accessing a non-static variable from a static environment:**

**public class StaticDemo**

**{**

**public String myStr = "somedata";**

**public static void main (String args[])**

**{**

**// Access a non-static member from static method**

**System.out.println ("This generates a compiler error" +**

**myStr );**

**}**

**}**

**How to fix it?**

**public class NonStaticDemo**

**{**

**public String my\_member\_variable = "somedata";**

**public static void main (String args[])**

**{**

**NonStaticDemo demo = new NonStaticDemo();**

**// Access member variable of demo**

**System.out.println ("This WON'T generate an error" +**

**demo.my\_member\_variable );**

**}**

**}**

**4. Mistyping the name of a method when overriding;**

**public class MyWindowListener extends WindowAdapter {**

**// This should be WindowClosed**

**public void WindowClose(WindowEvent e) {**

**// Exit when user closes window**

**System.exit(0);**

**}**

**});**

**5. Using assignment where comparison is needed (for primitive types)**

**if (a = b) //This is not correct**

**if (a == b) //This is correct**

**6. Comparing two objects ( == instead of .equals)**

**if (object1 == object2) //This is incorrect**

**if (object1.equals(object2)) //This is correct**

**7. Confusion over passing by value, and passing by reference**

Primitive types are passed by value, and objects are passed by reference.

**public class CommonMistakes {**

**private static class SomeClass**

**{**

**public int myInt;**

**public SomeClass(int x)**

**{**

**myInt = x;**

**}**

**}**

**public static void swapInt(int a, int b)**

**{**

**int temp = a;**

**a = b;**

**b = temp;**

**}**

**public static void swapSomeClass(SomeClass a, SomeClass b)**

**{**

**int temp = a.myInt;**

**a.myInt = b.myInt;**

**b.myInt = temp;**

**}**

**public static void main(String[] args)**

**{**

**int x = 10; int y = 20;**

**System.*out*.println("Before Swap: x = " + x + " y = " + y);**

***swapInt*(x, y);**

**System.*out*.println("After Swap: x = " + x + " y = " + y);**

**SomeClass p = new SomeClass(10);**

**SomeClass q = new SomeClass(20);**

**System.*out*.println("Before Swap: p = " + p.myInt +**

**" q = " + q.myInt);**

***swapSomeClass*(p, q);**

**System.*out*.println("After Swap: p = " + p.myInt +**

**" q = " + q.myInt);**

**}**

**}**

**8. NULL Pointer Exceptions**

**String[] names = new String[5];**

**for (int k = 0; k < names.length; k++)**

**{**

**//Just print the first letter of each name**

**System.out.println(names[k].charAt(0));**

**// java.lang.NullPointerException at the line above**

**}**

**LAB #01: Partner1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Partner2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PART A. Locating and Correcting Programming Errors (5)**

**(Write answers and print this page)**

For each of the code segments below, first predict the output of the program. Type the code into a class called **lab01A** which just contains the main program and run it and show what the actual output is.

(a) **int x;   
x = x + 1;   
System.out.println("X = " + x);**

Expected output: X = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Computed output: X = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(b) **//print numbers 1 to 10**

**for (i=1; i<=10; i++) ; {   
  System.out.println("Number is " + i);   
}**

Expected output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Computed Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Is it correct? (Does it do what the comment says?) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If NOT correct, how do you correct the code (Write the corrected code below)?

**(c) if ( x > y) ;  {   
  System.out.println("X is bigger");   
}**

What is the output if x = 10, and y = 5? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the output if x = 5, and y = 10? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Are the answers correct? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If NOT correct, how do you correct the code (Write the corrected code below)?

(d) Refer to the program in #7 (Common Errors part) above.

Without running the program, guess the output of the program?

Output \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(e) Run the program in #8 (Common Errors part) above. Explain why the error occurred?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LAB #01: Partner1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Partner2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PART B**

**Programming Exercise: (USE Eclipse IDE) 15 pts**

**(AC-B1, IS-B1, AC-B2, IS-B2, AC-C2, AC-C3, IS-C3, AC-C4,IS-C4, AC-D1, IS-D1,**

**AC-E1 to E4, IS-E1 to E4, AC-F1-F2, IS-F1-F2)**

**Note:** Study the given a sample program called **TelephoneListConverter** and all theartifacts that came with it: use cases, CRC cards, skeleton code, test plan, and the actual code. In this lab you will be creating similar artifacts and writing software for another, very similar, problem (it belongs to the same family).

**Your Problem**: You have been given list of words (call it ***originalList***) each of which can contain letters, digits or any other characters. The encoding of these words is done as follows:

Characters are mapped as follows: (A maps to C, etc.)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |

Write a class called **EncodeDecode** that has

The following instance variables:

* originalList – This is an array of Strings, it is assigned a value by the constructor
* encodedList – For each word of the original list, this will contain the corresponding encoded word
* decodedList – For each word in the encodedList this will contain its decoded word, which should be the same as the corresponding word in the original list.

The following methods:

* Constructor: The original list is passed to the constructor as a parameter -oL. The constructor assigns oL to originalList. It will then call the encode and decode methods.
* String encode (String originalWord) – this maps every character in original word to 2 positions forward, with wraparound.
* String decode (String codedWord) – this maps every character in coded word to 2 positions back, with wraparound.
* char forwardMap(char ch) – supporting method, it maps the given ch to 2 positions forward (if the ch is not a letter or digit, it maps to itself)
* char backMap(char ch) – supporting method, it maps the given ch to 2 positions back (if the ch is not a letter or digit, it maps to itself)
* getEncodedList, getDecodedList – just get methods

**Use the sample artifacts provided to you. You should create (in this order) and turn in the following documents.**

1. Use Case
2. CRC Card
3. Code Skeletons
4. Test Plan: Make a table showing the input data and the expected output. Ensure that all parts of your code are properly tested.
5. Actual Code (*flesh-out* the skeleton code)
6. Terminal Window showing the results.

encodes and decodes, and a class called **EncodeDecodeTester** that does the testing.

**USE THSE TEMPLATES TO CREATE YOUR USE CASE AND CRC CARD**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |

|  |  |
| --- | --- |
| **Use Case Name:** | Decode |
| **Description:** | |
| Given a list of encoded words, decode each of them into the original word by mapping each character to the original character. | |
| **Preconditions:** | |
| /\* What data should be known? \*/  encodedList: List of encoded words.  mapping: Standard mapping of characters.  originalList: List of original words after decoding. | |
| **Workflow:** | |
| /\* What do you do with the data you are given? \*/  Create an empty originalList  For each encodedWord in the encodedList do   * Map each of the characters in this encodedWord to create a originalWord * Characters are mapped from encoded to original using the table above. Non-letters and non-digits map to themselves. * Add the originalWord to originalList | |
| **Results:** | |
| /\* What do you get back after doing this thing? \*/  originalList: a list of original words after decoding. | |
| **Alternates:** | |
| /\* What can go wrong? \*/ | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | a | b |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |

|  |  |
| --- | --- |
| **Use Case Name:** | Encode |
| **Description:** | |
| Given a list of words, encode each of them by mapping each character to the corresponding encoded character. | |
| **Preconditions:** | |
| /\* What data should be known? \*/  originalList: List of original words.  mapping: Standard mapping of characters.  encodedList: List of encoded words after encoding. | |
| **Workflow:** | |
| /\* What do you do with the data you are given? \*/  Create an empty encodedlList  For each originalWord in the originalList do   * Map each of the characters in this originalWord to create a encodedWord * Characters are mapped from original to encoded using the table above. Non-letters and non-digits map to themselves. * Add the encodedWord to encodedList | |
| **Results:** | |
| /\* What do you get back after doing this thing? \*/  encodedList: a list of encoded words after encoding. | |
| **Alternates:** | |
| /\* What can go wrong? \*/ | |

|  |  |  |
| --- | --- | --- |
| **Class Name:** | **EncodeDecode** | |
| **Description:** | | |
| Encodes and decodes a list of words | | |
| **Stereotype:** | | |
|  | | |
| **Subclass:** | | |
|  | | |
| **Super-class:** | | |
|  | | |
| **Attribute:** | | **Type:** |
| OFFSET  originalList  encodedList  decodedList | | Int  String[]  String[]  String[] |
| **Method:** | | **Collaborating Classes:** |
| EncodeDecode(String[] originalList)  String encode(String originalWord)  String decode(String codedWord)  char forwardMap(char originalCharacter)  char backMap(char codedCharacter)  String[] getEncodedList()  String[] getDecodedList() | |  |

**Test Data:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Word** | **Expected Encoded Word** | **Encoded Word** | **Pass?** |
| ABCDEFGHIJKLMNOPQRSTUVWXYZ | CDEFGHIJKLMNOPQRSTUVWXYZAB | CDEFGHIJKLMNOPQRSTUVWXYZAB | Passed |
| abcdefghijklmnopqrstuvwxyz | cdefghijklmnopqrstuvwxyzab | cdefghijklmnopqrstuvwxyzab | Passed |
| 123456789 | 2345678901 | 2345678901 | Passed |
| ABCDEFGHIJKLMnopqrstuvwxyz | CDEFGHIJKLMNOpqrstuvwxyzab | CDEFGHIJKLMNOpqrstuvwxyzab | Passed |
| abc012 | cde234 | cde234 | Passed |
| !,.?'[]{}()<> | !,.?'[]{}()<> | !,.?'[]{}()<> | Passed |
|  |  |  | Passed |
| AZaz09 | CBcb21 | CBcb21 | Passed |
| abcXYZ012789.!? | cdeZAB234901.!? | cdeZAB234901.!? | Passed |
| A Z | C B | C B | Passed |
| a z | c b | c b | Passed |
| abc xyz | cde zab | cde zab | Passed |
| 123+123=123+123 | 345+345=345+345 | 345+345=345+345 | Passed |
| 567 MNO mno | 789 OPQ opq | 789 OPQ opq | Passed |
| a,b,c,A,B,C,1,2,3 | c,d,e,C,D,E,3,4,5 | c,d,e,C,D,E,3,4,5 | Passed |

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Word** | **Expected Decoded Word** | **Decoded Word** | **Pass?** |
| ABCDEFGHIJKLMNOPQRSTUVWXYZ | ABCDEFGHIJKLMNOPQRSTUVWXYZ | ABCDEFGHIJKLMNOPQRSTUVWXYZ | Passed |
| abcdefghijklmnopqrstuvwxyz | abcdefghijklmnopqrstuvwxyz | abcdefghijklmnopqrstuvwxyz | Passed |
| 123456789 | 123456789 | 123456789 | Passed |
| ABCDEFGHIJKLMnopqrstuvwxyz | ABCDEFGHIJKLMnopqrstuvwxyz | ABCDEFGHIJKLMnopqrstuvwxyz | Passed |
| abc012 | abc012 | abc012 | Passed |
| !,.?'[]{}()<> | !,.?'[]{}()<> | !,.?'[]{}()<> | Passed |
|  |  |  | Passed |
| AZaz09 | AZaz09 | AZaz09 | Passed |
| abcXYZ012789.!? | abcXYZ012789.!? | abcXYZ012789.!? | Passed |
| A Z | A Z | A Z | Passed |
| a z | a z | a z | Passed |
| abc xyz | abc xyz | abc xyz | Passed |
| 123+123=123+123 | 123+123=123+123 | 123+123=123+123 | Passed |
| 567 MNO mno | 567 MNO mno | 567 MNO mno | Passed |
| a,b,c,A,B,C,1,2,3 | a,b,c,A,B,C,1,2,3 | a,b,c,A,B,C,1,2,3 | Passed |

**CSC 205: SYSTEMATIC SOFTWARE DEVELOPMENT:**

**SAMPLE PROGRAM**

**Problem:** We have a list of telephone numbers some of which contain letters as well as numbers (for example: 395-COLD). We want to convert all of them into pure numbers using a standard transformation from letters to numbers:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ABC | DEF | GHI | JKL | MNO | PQRS | TUV | WXYZ |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Write a program to do this.

**PROBLEM ANALYSIS**

|  |  |
| --- | --- |
| **Use Case Name:** | Letters-To-Numbers |
| **Description:** | |
| Given a list of telephone numbers, possibly containing letters, convert each of them into purely numbers by mapping each letter to the corresponding number. | |
| **Preconditions:** | |
| /\* What data should be known? \*/  mixedTelNumList: List of telephone numbers containing letters and numbers mixed  listSize: Number of Telephone Numbers  mapping: Standard mapping of letters to numbers  pureTelNumList: list of telephone numbers after mapping | |
| **Workflow:** | |
| /\* What do you do with the data you are given? \*/  Create an empty pureTelNumList  For each mixedTelNum in the mixedTelNumList do   * Map each of the characters in this mixedTelNum to create a pureTelNum * Letters are mapped using the table above. Digits and the hyphen (‘-‘) map to themselves. All other characters map to ‘$’ * Add the pureTelNum to pureTelNumList | |
| **Results:** | |
| /\* What do you get back after doing this thing? \*/  pureTelNumList: A list of pure (numbers only) telephone number list | |
| **Alternates:** | |
| /\* What can go wrong? \*/ | |

**CRC Card (CLASS-RESPONSIBILITY-COLLABORATION Card)**

|  |  |  |
| --- | --- | --- |
| **Class Name:** | TelephoneListConverter | |
| **Description:** | | |
| /\* General Description of the class \*/  Coverts a list of mixed (letter/number) telephone numbers to pure numbers | | |
| **Subclass:** | | |
| /\* List any subclasses here \*/  None | | |
| **Superclass:** | | |
| /\* List any class that this is a subclass of \*/  None | | |
| **Attribute:** | | **Type:** |
| /\* List attributes and their types \*/  mixedTelNumList  pureTelNumList | | Array of Strings  Array of Strings |
| **Method:** | | **Collaborating Classes:** |
| /\* List Methods \*/  //Constructor  TelephoneListConverter(String[] mixedList)  //Maps the mixedNumber string to pureNumber //string and returns it.  String mapString(String mixedNumber)  //returns the pureNumberList  String[] getPureTelNumList()  //Maps one char from mixed to pure. Used by //mapString method  char mapChar(char thisChar) | | String  String  String |

**Skeleton Code:**

/\*\*

\* Converts a list of mixed letter-digit telephone numbers

\* into pure telephone numbers using the standard mapping.

\*

\* @author T.M. Rao

\* @version

\*/

**public class TelephoneListConverter**

**{**

**//-----------------------------------------------------**

**//Input list of mixed numbers**

**private String[] mixedTelNumList;**

**//-----------------------------------------------------**

**//Resulting list of pure numbers**

**private String[] pureTelNumList;**

**//-----------------------------------------------------**

**public TelephoneListConverter(String[] mixedList)**

**{**

**}**

**//-----------------------------------------------------**

**/\*\***

**\* The constructor**

**\* @param mixedNumber: A string object**

**\* @return String: result of the mapping**

**\*/**

**String mapString(String mixedNumber)**

**{**

**}**

**//-----------------------------------------------------**

**/\*\***

**\* This maps one characeter to one number using the standard**

**\* mapping. Any chars other than numbers, uppercase or lowercase**

**\* letters, will be mapped to $**

**\*/**

**char mapChar(char ch)**

**{**

**}**

**//-----------------------------------------------------**

**String[] getPureTelNumList()**

**{**

**}**

**}**

**Test Plan:**

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Output | Computed Output | Do they match? |
| 777-HOME | 777-4663 |  |  |
| 777-home | 777-4663 |  |  |
| 777-SAVE | 777-7283 |  |  |
| 777-save | 777-7283 |  |  |
| 1-800-TALK-LAW | 1-800-8255-529 |  |  |
| 1-800-talk-law | 1-800-8255-529 |  |  |
| 123-4567 | 123-4567 |  |  |
| 123#4567 | 123$4567 |  |  |
| 12$$%%&& | 12$$$$$$ |  |  |
| ABC DEF GHHH | 222$333$4444 |  |  |
| IJK-LMN-OPQR | 455-566-6777 |  |  |
| STU-VWX-YZZZ | 788-899-9999 |  |  |
| abc def ghhh | 222$333$4444 |  |  |
| ijk-lmn-opqr | 455-566-6777 |  |  |
| stu-vwx-yzzz | 788-899-9999 |  |  |

**TESTER CLASS**

**package lab01;**

**/\*\***

**\* A Tester class for the TelephoneListConverter**

**\* @author TMR**

**\* @version May 2012**

**\*/**

**public class TelephoneListConverterTester**

**{**

**//--------------------------------------------------------------**

**public static void main(String[] args)**

**{**

**//Create test data**

**String[] mixedNumbers = {**

**"777-HOME", "777-home", "777-SAVE",**

**"777-save", "1-800-TALK-LAW", "1-800-talk-law",**

**"123-4567", "123#4567", "12$$%%&&",**

**"ABC DEF GHHH", "IJK-LMN-OPQR", "STU-VWX-YZZZ",**

**"abc def ghhh", "ijk-lmn-opqr", "stu-vwx-yzzz"**

**};**

**String[] expectedAnswers = {**

**"777-4663", "777-4663", "777-7283",**

**"777-7283", "1-800-8255-529", "1-800-8255-529",**

**"123-4567", "123$4567", "12$$$$$$",**

**"222$333$4444", "455-566-6777", "788-899-9999",**

**"222$333$4444", "455-566-6777", "788-899-9999"**

**};**

**//Create the converter object**

**TelephoneListConverter tlc =**

**new TelephoneListConverter(mixedNumbers);**

**//Get the mapped list**

**String[] pureTelNumList = tlc.getPureTelNumList();**

**//Print the table**

**System.out.println("Table of Test Results: "+**

**"\n------------------------------------------------" +**

**"\nInput Value \tExpected Result \tComputed Result ");**

**for (int index = 0; index < pureTelNumList.length; index++)**

**{**

**System.out.println(mixedNumbers[index] + "\t" +**

**expectedAnswers[index] + "\t" +**

**pureTelNumList[index]);**

**}**

**}**

**}**

**//--------------------COMPLETED CODE------------------------**

**package lab01;**

**/\*\***

**\* Converts a list of mixed letter-digit telephone numbers**

**\* into pure telephone numbers using the standard mapping.**

**\***

**\* @author T.M. Rao**

**\* @version May 2012**

**\*/**

**public class TelephoneListConverter**

**{**

**//-----------------------------------------------------**

**//Input list of mixed numbers**

**private String[] mixedTelNumList;**

**//-----------------------------------------------------**

**//Resulting list of pure numbers**

**private String[] pureTelNumList;**

**//-----------------------------------------------------**

**public TelephoneListConverter(String[] mixedList)**

**{**

**//Assign parameter to instance variable**

**mixedTelNumList = mixedList;**

**//Create another string array of the same size**

**pureTelNumList = new String[mixedTelNumList.length];**

**//Do the mapping**

**for (int index = 0; index < mixedTelNumList.length; index++)**

**{**

**pureTelNumList[index] = mapString(mixedTelNumList[index]);**

**}**

**}**

**//-----------------------------------------------------**

**/\*\***

**\* @param mixedNumber: A string object**

**\* @return String: result of the mapping**

**\*/**

**private String mapString(String mixedNumber)**

**{**

**//Start with empty result string**

**String result = "";**

**//for each character in the mixedNumber**

**for (int index = 0; index < mixedNumber.length(); index++)**

**{**

**//map it to number and concat it to the result**

**result = result + mapChar(mixedNumber.charAt(index));**

**}**

**//return result string**

**return result;**

**}**

**//-----------------------------------------------------**

**/\*\***

**\* This maps one character to one number using the standard**

**\* mapping. Any chars other than numbers, uppercase or lowercase**

**\* letters, will be flagged as errors and left unchanged**

**\*/**

**private char mapChar(char ch)**

**{**

**//map the ch to upper case**

**char thisChar = Character.toUpperCase(ch);**

**if (thisChar >= 'A' && thisChar <= 'C')**

**return '2';**

**if (thisChar >= 'D' && thisChar <= 'F')**

**return '3';**

**if (thisChar >= 'G' && thisChar <= 'I')**

**return '4';**

**if (thisChar >= 'J' && thisChar <= 'L')**

**return '5';**

**if (thisChar >= 'M' && thisChar <= 'O')**

**return '6';**

**if (thisChar >= 'P' && thisChar <= 'S')**

**return '7';**

**if (thisChar >= 'T' && thisChar <= 'V')**

**return '8';**

**if (thisChar >= 'W' && thisChar <= 'Z')**

**return '9';**

**if (thisChar >= '0' && thisChar <= '9')**

**return thisChar;**

**if (thisChar == '-')**

**return thisChar;**

**//Any other character is invalid. We return $**

**return '$';**

**}**

**//-----------------------------------------------------**

**String[] getPureTelNumList()**

**{**

**return pureTelNumList;**

**}**

**}**

**TERMINAL WINDOW SHOWING OUTPUT**

**Table of Test Results:**

**------------------------------------------------**

**Input Value Expected Result Computed Result**

**777-HOME 777-4663 777-4663**

**777-home 777-4663 777-4663**

**777-SAVE 777-7283 777-7283**

**777-save 777-7283 777-7283**

**1-800-TALK-LAW 1-800-8255-529 1-800-8255-529**

**1-800-talk-law 1-800-8255-529 1-800-8255-529**

**123-4567 123-4567 123-4567**

**123#4567 123$4567 123$4567**

**12$$%%&& 12$$$$$$ 12$$$$$$**

**ABC DEF GHHH 222$333$4444 222$333$4444**

**IJK-LMN-OPQR 455-566-6777 455-566-6777**

**STU-VWX-YZZZ 788-899-9999 788-899-9999**

**abc def ghhh 222$333$4444 222$333$4444**

**ijk-lmn-opqr 455-566-6777 455-566-6777**

**stu-vwx-yzzz 788-899-9999 788-899-9999**