The lab is a series of problems for you to complete in NetBeans on a cs linux computer (or your own computer). Each problem corresponds to a NetBeans project included in the files for the exam. The instructions for each problem are included in this document, below. In solving each problem you should make appropriate changes to the corresponding NetBeans project. This practice exam is open-book and open notes. You may access the Internet only for the purpose of consulting the following sites:

Java Documentation: http://docs.oracle.com/javase/6/docs/api/
ACM Student Package Documentation: http://jtf.acm.org/javadoc/student/
Java Tutorial On Line: http://docs.oracle.com/javase/tutorial/
Intro Java Text, David J. Eck: http://math.hws.edu/javatext

When you are done with your practice exam, you should make a zip file of all the NetBeans projects and give the zip file a name similar to Jones-Tom-Midterm.zip (that is, yourlastname-yourfirstname-Midterm.zip) and upload it to the Lab 4 Practice Exam Drop Box on our Moodle Site. You should not look at other students’ code. Be sure to get checked off by a coach or your professor when you are done.

1. Profile: Extend this program to print profiles of people enrolled on an Internet dating site. First create 5 instance variables for the Profile class: String username, int age, char gender, String city, and String state. Next, create a 5-parameter constructor for the Profile class. The constructor should give values to all the instance variables in the class. Create 2 objects of type Profile in the main method. Define an instance method printProfile that takes no input parameters, returns void, and prints the object it is called on in the bold text shown below. Invoke printProfile twice in the main method on the two objects of type Profile that you instantiated.

   User Name: savvySuzy
   Age: 51
   Gender: Female
   Home Town: Little Rock, AK

   User Name: coolDude
   Age: 31
   Gender: Male
   Home Town: Baton Rouge, LA

2. DrawShapes: This program entails the rendering of ASCII blocks to form squares and lower triangles of a square. The main method should ask if the user wants to print a SQUARE (0), LOWER-LEFT(1), OR UPPER-LEFT(2) block. The input and output should look like this (input in bold, output in normal text in this document, not in your program):

   Please enter the shape you want to create (0 for square, 1 for lower-left triangle, or 2 for upper left triangle): 2
   upper left
   * * * * * * * *
   * * * * * * *
   * * * * * *
   * * * * *
   * * * *
   * * *
   * *
   *

"
Each shape is 8 characters wide by 8 characters high (with a space before and after each consecutive character on the same line). The number 8 is declared as a constant called SIZE. Other shapes that your program should be able to make are:

- **square**
  
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *

- **lower left**
  
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *
  * * * * * * *

3. **CycleArray**: This program demonstrates the method: `int[] rightShift(int[] array)`. The CycleArray class has one instance variable of type `int[]` called `myArray`. It has one constructor that puts the values 0…3 in the array `myArray`. The `rightShift` instance method takes an array of type `int[]` and returns a new integer array of the same length. The values in the output array returned by `rightShift` are a cyclic permutation of the values in the input array, i.e., each entry is shifted one step toward the beginning of the array, and the first array entry is placed at the end. Complete the definition of the `rightShift` instance method so that the program prints the following (the code to print is already in the CycleArray class):

```
Initial Array: < 0 1 2 3 >
Shifted 1 Times: < 1 2 3 0 >
Shifted 2 Times: < 2 3 0 1 >
Shifted 3 Times: < 3 0 1 2 >
Shifted 4 Times: < 0 1 2 3 >
```

Read through the file and note the static and non-static members. Do not change any of the non-static members to static in your solution.

4. **StringMask**: This program demonstrates the method: `String maskString(String source)`, which accepts a `String` as input and returns a new `String`. The new string is the same as the original one except that randomly selected characters are replaced with an asterisk. Each character in `source` appears in the corresponding place in the new string, with probability given by a static constant: `PROBABILITY`. Otherwise, the character is replaced by an asterisk. The `maskString` method should avoid unnecessary copying of string fragments. Complete the definition of the `maskString` method so that the program prints out Strings like the following:

```
Enter a line of text:
Who let the dogs out?
Wh* e**the*d*gs **t?
```