Faculty of Arts & Sciences
Department of Computer Science
CMPS 200—Introduction to Programming
Assignment 8 – Due November 7, 2014

Notes

- You should write the appropriate static methods to show structure in your solution and to help you in writing other methods.
- Give meaningful names to methods and variables in your code.
- Include a comment at the beginning of your program with basic information and a description of the program and include a comment at the start of each method.

Exercise 1

A password is a sequence of characters that can be used for authentication purposes. Passwords are widely used in computer systems and networks; and they are often used to authenticate the identity of an automated data processing (ADP) system user.

A valid password is a personal password that will authenticate the identity of an individual when presented to a password system.

In this problem you are asked to check the validity of passwords through the development of some routines that will play the role of a password system. Basic factors shall be considered in the design, implementation, and use of a password system used to authenticate the identity of a person. In this problem we only focus on two main factors, comprising 4 distinct rules:

- Length Range Factor: Length Range is the set of acceptable lengths of passwords, expressed as a minimum length through a maximum length.
  - Rule 1: acceptable number of characters in a valid password must be more than 6 and less than 15.
- Composition Factor: Composition is the set of acceptable characters which may be used in a valid password.
  - Rule 2: valid password should not be only alphabetic characters
  - Rule 3: valid password should not be only numeric characters
  - Rule 4: valid password should not be a series of alphabetic characters ending with '99'

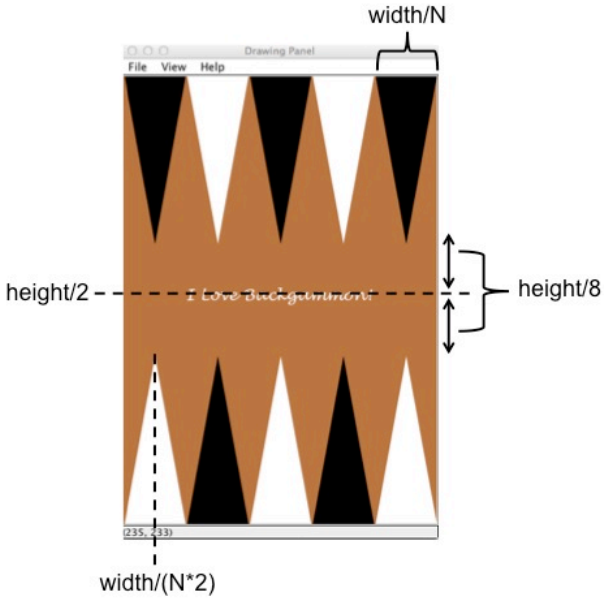
Write a program, `PasswordChecker.java`, which asks the user to enter a password and then checks its validity. The program should print "valid password" if the password follows the rules presented above. If the password violates any of the rules, the program should print "not a valid password" along with the rules that are violated. The program should keep repeating the foregoing process until the user enters a valid password.

Sample Run

```
Please enter your password: CGI
not a valid password (rule(s): 1,2 violated)
Please enter your password: a199
not a valid password (rule(s): 1 violated)
Please enter your password: 99
not a valid password (rule(s): 1,3 violated)
Please enter your password: CGI99
not a valid password (rule(s): 1,4 violated)
Please enter your password: CMPS200
valid password
```

Exercise 2

Write a program `Backgammon.java` that takes as command line input the number of triangles *N*, and the base width and height of the window, and plots the board of the game Backgammon according to the measurements in the following figure.



Example: for a base width: 400 and height: 600, and 5 triangles to be drawn, the corresponding graphics will be as follows:

- Drawing panel is of size 410x600 (410px = 400px base width + 2px times *N*)
- Board is of color (red=191, green=118, blue=73)
- Board is at (0, 0), size 410x600
- WHITE "I Love Backgammon!" text at relative position (82, 300)
    - Font is “Lucida Handwriting”, Bold, size: 20 (i.e., 400 / 20)
- Triangles alternate in either WHITE or BLACK colors (opposite triangles have opposite colors)
- Each triangle is 80px (base width / *N*) wide
- **T**riangles are 82px apart (2 blank pixels in between)

Examples:

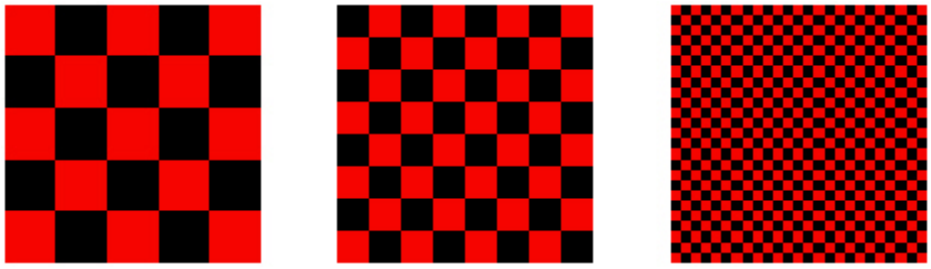
```
> java Backgammon 6 400 600
```

```
> java Backgammon 10 400 600
```


```
> java Backgammon 6 200 300
```

Exercise 3

a) Write a program `CheckerBoard.java` that takes a command line input `N` and plots an `N-by-N` checkerboard. Always color the lower left square red. Draw the squares in red and black. Below is the desired output for N = 5, 10, and 25.



b) Write a variation on the program that reads three command line parameters and draws three corresponding checkerboards.

Exercise 4

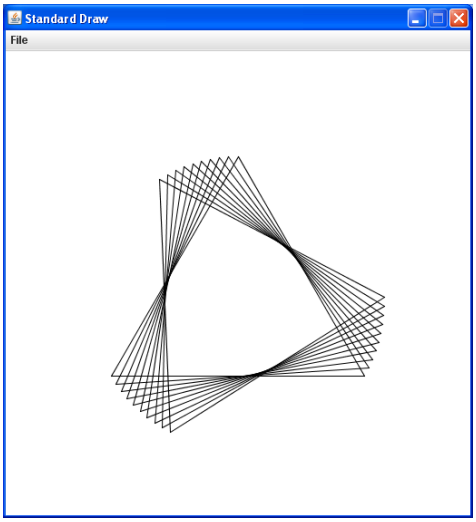
In this problem, `Rotating`, you will implement a rotating triangle ABC around its center of gravity G. Given point A with coordinates `(xa, ya)`, point B with coordinates `(xb, yb)`, and point C with coordinates `(xc, yc)`, the coordinates of the center of gravity G of triangle ABC are given by:

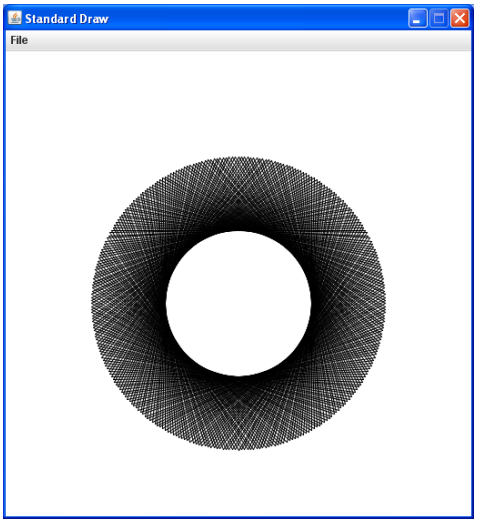
```
xg=(xa+xb+xc)/3
yg=(ya+yb+yc)/3
```

To rotate triangle ABC around its center of gravity G by a certain angle `theta`, you will actually need to rotate the coordinates of A, B, and C by theta producing 3 new points A', B', and C', then drawing triangle A'B'C'. To compute the rotated points, you will have to use the following equations: if `G(xg,yg)` is the center of gravity and `A(xa,ya)` is the point to be rotated by an angle `theta` then the new resulting point `A'(xa',ya')` will have the following coordinates:

```
xa'=(xa-xg)cos(theta)-(ya-yg)sin(theta)+xg
ya'=(xa-xg)sin(theta)+(ya-yg)cos(theta)+yg
```

Write a program, `Rotating`, that takes the following triangle coordinates `A(0.5,0.8)`, `B(0.2,0.28)` and `C(0.8,0.28)` (No need to read the coordinates from the console or the command line; you can use them directly in your source file). Your program should rotate this triangle around its center of gravity G. Your triangle should rotate by `0.02*pi` radians every `50ms`. The result must be a rotating triangle that will never stop as shown below. Here are two snapshots of running the program. Create two methods other than main: (1) `drawTrinagle` that takes the coordinates of three points and draws the corresponding triangle, (2) `rotTriangle` that takes the coordinates of three points, the coordinates of their center of gravity, and an angle `theta`. This method should rotate every point by `theta`, then calls `drawTriangle` to draw the rotated triangle.





Submission Instructions

- Your submission must consist of a zip archive that contains five .java files only (called **`PasswordChecker.java`, `Backgammon.java`, `CheckerBoard.java`, `CheckerBoard1.java`,** and **`Rotating.java`**). No additional files should exist in the .zip archive.
- The name of the zip file must adhere to the following naming convention `asst8_`*`<netid>`*, where *`<netid>`* stands for you AUBnet user name. These zipped files will be processed automatically so please make sure you use this naming convention. The single zipped file must be uploaded to Moodle.