COSC 211 Project #1 Mini-concentration game

Distributed: 7 January 2008
Due: 23 January 2008

This project is slightly modified from Savitch's Programming Project #2 in Chapter 6. The intent of this assignment is to cement your recollection of Java I techniques and knowledge and to demonstrate an understanding of OO programming, 2D arrays, random number generation and screen output. It's a fairly straightforward project also intended to give you time to assimilate Eclipse.

A common memory matching game played by young children is to start with a deck of cards that contain identical pairs. For example, given six cards in the deck, two might be labeled 1, two labeled 2, and two labeled 3. The cards are shuffled and placed face down on the table. A player then selects two cards that are face down, turns them face up, and if the cards match they are left face up. If the two cards do not match they are returned to their original face down position. The game continues until all cards are face up.

Write a program that plays the memory matching game (randomly shuffled cards, single play, fixed array size – you can add functionality if you want). Use 16 cards that are laid out in a 4 X 4 square and are labeled with pairs of numbers from 1 to 8. Your program should allow the player to specify the cards that he would like to select through a coordinate system. Also keep track of, and display, the attempt number.

For example, in the following layout:

```
    0  1  2  3
   ------------
0 | 8  *  *  *
1 | *  *  *  *
2 | *  8  *  *
3 | *  *  *  *
```

All of the cards that are face down are indicated by *. The pairs of 8 which are face up are at coordinates (0, 0) and (2, 1).

Use a single 2D array where each element is an object that stores both the card's value and face (showing or not-showing).

To generate a uniformly distributed (double) random number *x*, where $0 < x < 1$, use `x = Math.random();` In order to obtain specific integer numbers in a range [min .. max], scale by *(max – min+1)* then translate by *min*.

For example, to generate uniformly distributed x where
`x ε { 1, 2, 3, 4, 5, 6 }`
use `x = (int) (6 * Math.random() + 1)`

More elaborate random number values can be generated in a similar way using an intermediate array. Suppose you want to randomly generate a single card in a standard deck. E.g., "Ace" of "Spades". Here is one approach (in pseudo-code):

Initialize an array `suites = { "Spades", "Hearts", "Clubs", "Diamonds" }`. An index in the array `suites` ranges between 0 and 3.
Initialize an array `values = { "Ace", "Deuce", "Trey", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King" }`. An index in the array `values` ranges between 0 and 12

To generate a random card, generate two random numbers:
```
x = (int) Math.random() * 4;  // i.e., 0, 1, 2, or 3
y = (int) Math.random() * 13;
```

The randomly generated card is `suites[x ] + values[y ];  // concatenate`

Alternative, you could initialize a single array `cards = {"ace of spades", "deuce of spades", "trey of spades", … "jack of diamonds", "queen of diamonds", "king of diamonds"};`
There are 52 elements in the array `cards` (index ranging from 0 to 51).

Then a randomly generated card is `cards[(int)Math.random()*52 ];`

To make the concentration game interesting, you have to create shuffled arrays. There are two simple ways of doing this. First, initialize an array containing each card. E.g., for the concentration game, you would use {1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8}.
Second, do (1) or (2).

(1) In a fairly large loop (the upper bound is very important – deciding a good upper bound is beyond our scope), say `for (i=0; i<array.length*1000; i++)`, randomly choose 2 cards,
```
      x = (int) Math.random() * array.length;
      y = (int) Math.random() * array.length;
```
and swap the cards in the array. Say x == 2 and y == 15. The new array is {1, 1, 8, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 2}. After a large number of swaps the array has been shuffled.

(2) Make a "hand" of cards by drawing out one at a time. The first card you draw out of the deck is randomly chosen:
```
      x = (int) Math.random() * array.length; // x is the index!
```

That card cannot be chosen again. So replace the value of `cards[x ]` with the value of `cards[array.length-1 ]`.
Now, chose your second random card index
```
      x = (int) Math.random() * (array.length - 1).
```

An example will help. Say you want to draw a poker hand for 5-card stud.

```
declare and initialize cards[] with all 52 cards.
declare hand[5 ];
declare int x;

for (int i=0; i<5; i++) {
     x =  (int) Math.random()*(cards.length – i) ;
     hand[i ] = cards[x ];
     cards[x ]= cards[cards.length - i – 1]);
     }
```

Turn in:

Hardcopy of code
Screen shots of 5 consecutive moves of a single game
Give demo.

Grade based on:

Meets specifications:	70%
Code elegance:	10%
Meets coding standards:	20%

My advice:
Write this up for array size 1 X 1. Debug.
Then expand to 2 X 2. Debug.
Then expand to 4 X 4.