



CS 101 - Problem Solving and Structured Programming

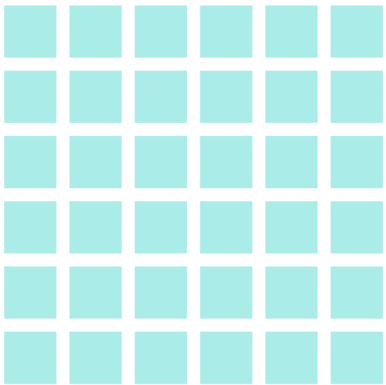
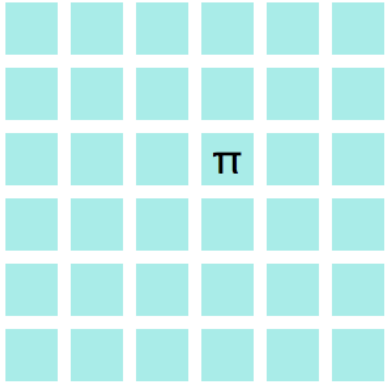
Lab 8 - Concentration

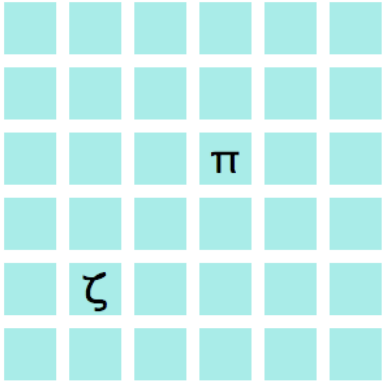
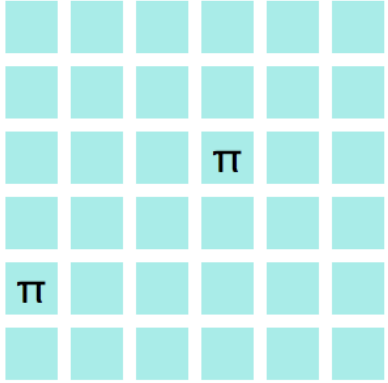
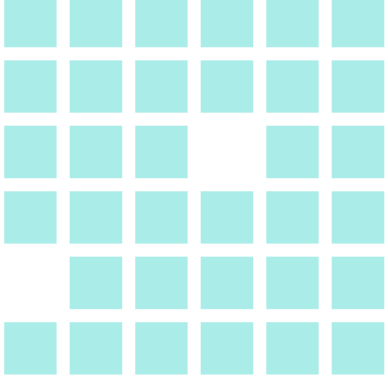
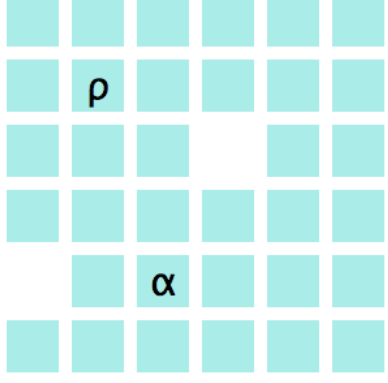
Due: December 2/3

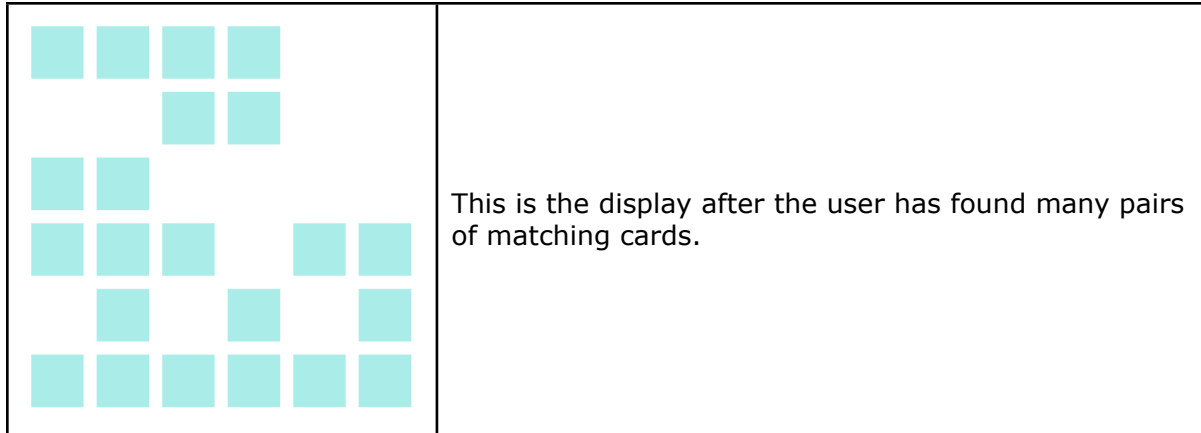
Introduction to the Assignment

In this lab, you will work with arrays. You will have an array of cards. The cards will come in pairs, where each card in a pair contains the same symbol. Your program will shuffle the cards and then display them "face down" so the symbols are not showing. When the user clicks on a card, the symbol is displayed. When the user clicks on a second card, its symbol is also displayed. If the cards have the same symbol, on the next click both cards are removed from the display. If the cards have different symbols, they are both turned face down again. The user's goal is to remove all cards from the display in as few clicks as possible by remembering where they have seen the symbols.

Here are some snapshots from this program:

	<p>This is the original display. All cards are drawn with their symbols hidden.</p>
	<p>When the user clicks on a card, its symbol is shown.</p>

	<p>When the user clicks on a second card, its symbol is also shown. Since the cards have different symbols, the next click anywhere in the canvas will hide both symbols.</p>
	<p>Here the user has found 2 cards with the same symbol.</p>
	<p>When the user clicks anywhere on the canvas after finding 2 matching cards, both cards disappear.</p>
	<p>After 2 more clicks, the user discovers 2 more cards that don't match.</p>



The Program Design

This project will have 3 classes: Concentration, CardCollection and Card.

The Concentration class extends WindowController. It is responsible for defining the begin method and the event handlers.

- begin method - This method should assign symbols randomly to cards and draw the display consisting of the facedown cards.
- onMouseClick method - This method should determine which, if any, card has been clicked on. If the user clicks on a facedown card and there are currently 0 or 1 faceup cards, the card is turned faceup. If the user clicks anywhere with 2 faceup cards, if the cards have the same symbol, they are both removed from the display. If the 2 faceup cards have different symbols, they are both turned facedown.

The CardCollection class manages the entire collection of cards. It is very similar to the ShapeCollection class in the DrawingProgramWithArrays example that we did in class. This class keeps the entire collection of cards in an array. It provides:

- A constructor that creates an empty array large enough to hold the number of cards we expect to have
- A method to add a card to the collection
- A method to find a card at a particular location
- A method to remove a particular card from the collection

The Card class defines an individual card. As with the Button class last week, your program will construct many cards. A Card consists of a rectangle and a symbol. It provides:

- A constructor - This should take parameters for the left and top coordinates of the card, the symbol to display on the card, and the canvas.
- A contains method - onMouseClick should call this method to determine if the user clicked on a card
- A getSymbol method - this method should return the symbol displayed on the card
- A showSymbol method - this method should display the symbol, making it appear that the card is faceup.
- A hideSymbol method - this method should hide the symbol, making it appear that the card is facedown
- A removeFromCanvas method - this method should remove the card and its symbol from the canvas.

Writing the Program

Create a new BlueJ project called Concentration.

Step 1: Displaying the cards

The begin method of the Concentration class should create an empty CardCollection, create the cards and add the cards to the collection. The cards should be laid out in a 6x6 grid.

The CardCollection should be constructed before creating the cards.

The Card constructor will be called within nested loops, similar to the nested loops used in the KnitAScarf example. Each iteration of the loop will calculate the location for the next card. After constructing a Card, you should add the new card to the CardCollection object.

The Card constructor should have the following signature:

```
public Card (char symbol, int left, int top, DrawingCanvas canvas)
```

The symbol is an individual character. In Java, a char constant is simply a letter enclosed in `'`. For example, `'a'` is the character a. For this step, when you construct your cards, you should put the same symbol (like `'a'`) on each card. You will need to create a Text object to display the symbol and center this Text object on the card, like you did with Buttons last week. You will probably also want to increase the size of the font used when displaying the card. You can do this by calling the following method defined in the Text class:

```
public void setFontSize (int size)
```

You should change the font size before you center the text since changing the font size changes the size of the text.

For now, the cards should appear to be faceup, with their symbols showing.

Use a color of your choosing as the color to display for the background of the cards.

Step 2: Turning an individual card faceup when clicked on

You will need an `onMouseClicked` method to do this. It will first need to determine which card the user clicked on, if any. To do this you must define the `getCardAt` method in the CardCollection class. You can model your implementation after the `getShapeAt` method in the ShapeCollection class of the drawing program we looked at in class.

CardCollection's `getCardAt` method will need to call Card's `contains` method to determine if a card was clicked on. You should be familiar with how to write a `contains` method.

You will also need to define the `showSymbol` methods in the Card class. The `showSymbol` method is quite simple. It just calls the `show` method already defined in the Text class.

To see that this method has any effect, you will need to hide the symbols when you create the cards. To do this, add a line to your Card constructor to hide the text after you have created it, by calling the `hide` method already defined in Text. Now, the initial display should have facedown cards. When you click on a card, that should appear to turn faceup.

Step 3: Turning a pair of cards facedown on the 3rd click

When the user plays the game, the first card the user clicks on is turned face up. When the user clicks on a different 2nd card, that card is turned face up. On the third click anywhere on the canvas, either both cards are turned face down or both cards are removed from the canvas and from the collection.

For now, just always turn the pair of cards facedown on the 3rd click. Don't bother checking whether they match or not.

To do this, you will need to introduce 2 instance variables to hold each of the cards in the pair that are turned face up. You will also need to use the special value null as an indicator that a card has not been set. For example, suppose the instance variable that you use to hold the first faceup card is called firstCard, then the following code can be used to determine if there are any faceup cards:

```
if (firstCard == null)
```

This would be true only if firstCard had no value. You can do something similar to see if there is a second faceup card by comparing the second variable to null.

When you turn the cards facedown, you need to set these variables to null, like this:

```
firstCard = null;
```

You should also be sure that the user clicks on 2 different cards.

In addition to setting these variables, you must also change the display. When a card is turned faceup, you call showSymbol on that card as before. When a card is turned facedown, you call hideSymbol on that card. The hideSymbol method will be very similar to the showSymbol method, except that you will call hide instead of show on the text.

Step 4: Putting different symbols on the cards

The program should have pairs of cards with the same symbol on them. That is, you might have 2 cards with 'a', 2 cards with 'b', etc. When you create the cards, you need to use different symbols. The char datatype provided by Java allows you to walk through an alphabet by "adding 1" to a character variable to get the next letter. For example, suppose you have the following code:

```
char c = 'a';  
c++;
```

At this point c will have the value 'b' since that is the letter that follows 'a'. As a result, you can use a loop to put symbols on the cards as long as the symbols you want to use are sequential in the alphabet. With a 6x6 grid, you need 18 different symbols since there are 2 cards with each symbol. That would be the letters 'a' to 'r'.

It is fine for you to use normal English letters. You may have noticed that the demo program uses Greek letters. It is extra credit to use symbols other than letters. How to do that is described later in this document.

Step 5: Deciding when to turn a pair facedown

A pair of cards should only be turned facedown on the 3rd click if the symbols of the cards are different. Add a test for the 3rd click to compare the symbols and only turn the cards facedown if they are different. For now, if the cards are the same, just leave them faceup.

To determine if the cards match, you must compare the symbols on the cards. To do this, add a getSymbol method to the Card class. This should just return the character that is on the card.

Step 6: Removing a pair of cards on the 3rd click

If the pair of cards have the same symbol. You should remove the cards from the card collection and also remove the cards from the canvas. To do this, you should define a removeCard method in the CardCollection class. This method should walk through the card array. When it finds the card to remove, it should set that entry in the array to null.

To remove a card from the canvas, define a removeFromCanvas method in the Card class. This method should remove the card by delegating to the rectangle and text that make up the card.

Step 7: Shuffling the symbols so the cards appear random

If you just put 'a' on the first 2 cards, 'b' on the next 2 cards, 'c' on the next pair, this would be an extremely boring game. Instead, you want to randomly assign symbols to cards but you also need to be sure that each symbol appears on exactly 2 cards.

To do this, you should create an array in the Concentration class that is large enough to hold 36 characters. Initialize this to hold 2 copies of 18 consecutive characters (the letters 'a' to 'r'). Then you will shuffle the symbols.

To shuffle the entries in an array, you will use the following algorithm to repeatedly swap 2 randomly-chosen array entries:

```
create a random integer generator ranging from 0 to 35
loop 100 times
    use the random integer generator to pick 2 random array indexes
    copy the symbol at the first index into a local variable
    copy the symbol at the second index into the first index
    copy the local variable into the second index
```

You should shuffle the symbol array before creating the cards. Then, when you create the cards, walk through the symbol array to determine which symbol to place on each card. The effect will be that the cards are shuffled.

Extra credit: Using symbols other than the English alphabet

Obviously, computers can display symbols other than those on a typical keyboard. When using a word processor, there is generally support for selecting those characters from a palette to put them in your document. If you want to display special characters from a Java program, you must enter those characters by stating their Unicode value. For example, the Unicode value for α is '\u03B1'. This value can be assigned to a variable whose type is char, like this:

```
char c = '\u03B1';
```

If you create a Text object to display this character, it will look like α , not that funny sequence of letters and numbers. When you increment c by saying c++, c will then contain the Unicode value for β . Incrementing will allow you to step through Greek letters the same as you can with English letters.

It turns out that there are many alphabets available by using codes like this. To see the possible codes, go to <http://unicode.org/charts/>. There are also codes available for many common non-alphabetic symbols. To see these codes, go to <http://unicode.org/charts/symbols.html>.

Grading

10	Step 1: Drawing the cards
10	Step 2: Turning a card faceup when clicked on
10	Step 3: Turning a pair of cards facedown
5	Step 4: Putting different symbols on the cards
5	Step 5: Deciding when to turn cards facedown
5	Step 6: Removing a pair of cards
5	Step 7: Shuffling the symbols
10	Comments
5	Constant declarations
5	Indentation
10	Variable names
5	Instance variables / local variables / parameters
5	If statements / loops
5	Method design
5	Arrays
5	Extra credit: non-English symbols

Turning in Your Work

Go to ella. Click on COMSC 101 in the toolbar menu across the top. Then click on Assignments in the left column. Click the submit link for this assignment. Click the Add Attachment button. Use the Browse... button to upload a local file. You should submit your Java files, but not the other files created by BlueJ. Click Continue. Then click the Submit button.