NAVIGATION

Home
- My home
  - My profile
  - Current course
    - CS 2114 Spr 2016
      - Participants
      - January 25 – January 31
        - Reading Quiz 2 (These prepare you for the followin...
        - Pre-lab 02 Activity: Equals vs ==
        - Instructional Design Survey
        - Lab 02: Polymorphism
        - Lab 02: Post-Lab Activity Overriding vs Overloadin...
        - CS2114 Lecture01 DesigningClasse 2016S
        - CS2114 Lecture02 OOGenericsExce 2016S
        - 1/25 Practice Session Problems
        - Project 1: Collection Display
  - My courses

SETTINGS

Course administration

My profile settings

PROJECT 1: COLLECTION DISPLAY

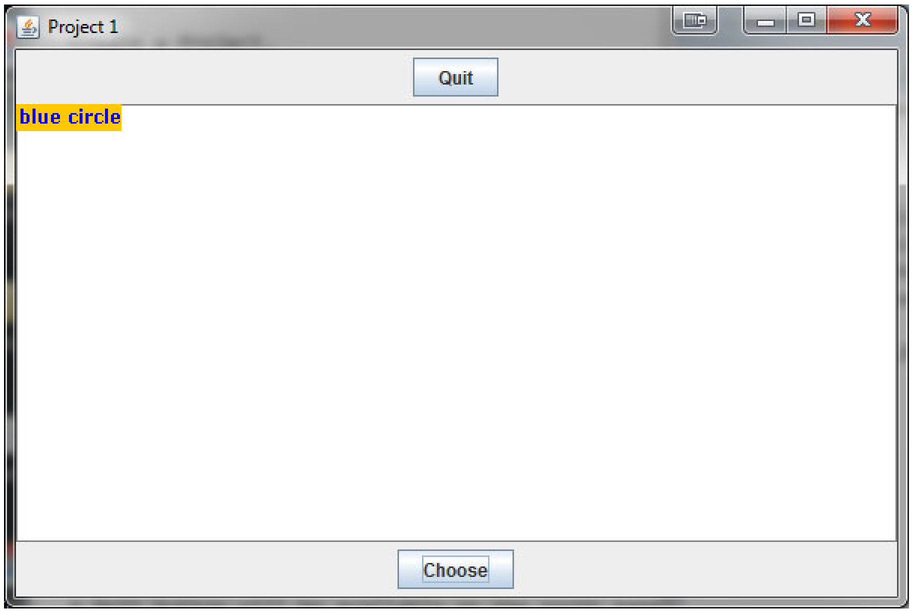
Project 1: Collection Display

Due Date: Monday, Feb 8. 8am.

Project 1: Collection Display

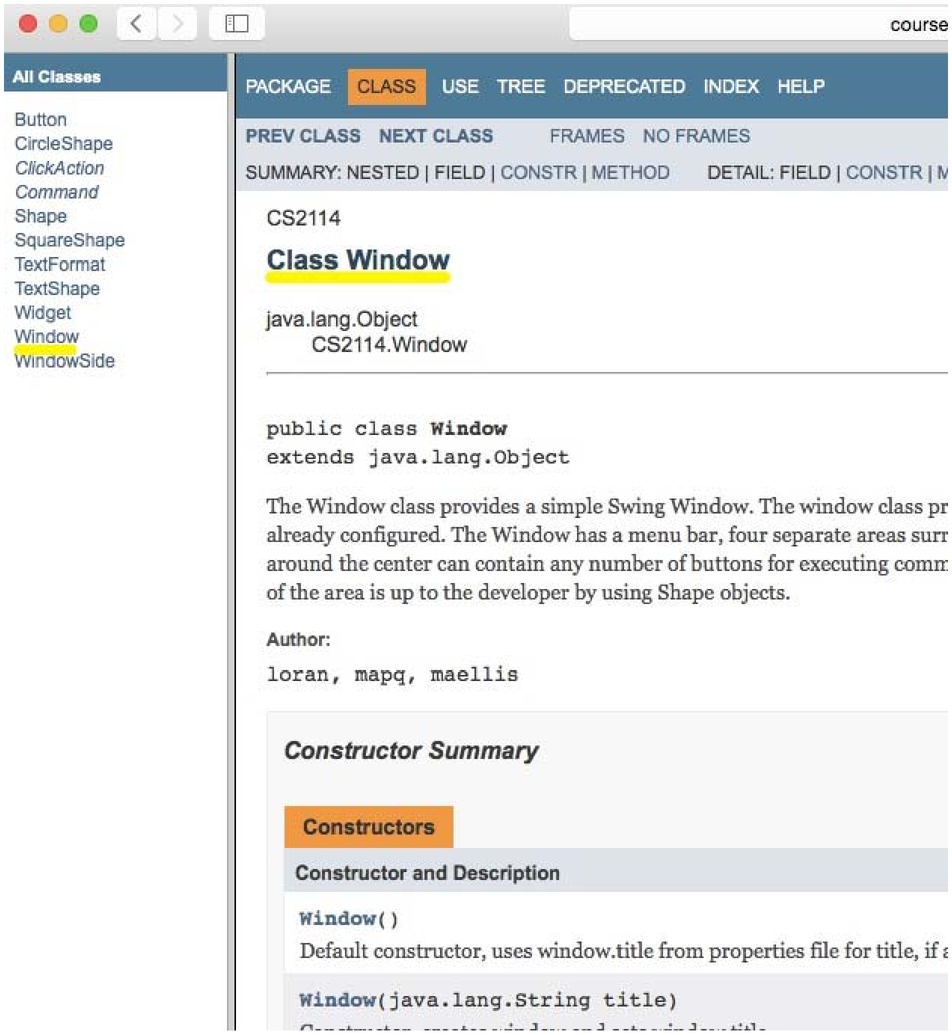
Introduction

The first data structure we are studying is a bag, it is introduced in Chapter 1 of your Carrano textbook. The goal of this project is to display the contents of a bag of strings with a Graphical User Interface (GUI). You will use a `Window` as your GUI, which will have two buttons: a "Quit" `Button`, which will close the `Window`, and a "Choose" `Button`, which will select and display a new string from your `Bag`. The strings can be: “red circle”, “blue circle”, “red square”, or “blue square”. The `Bag` will randomly contain 5 to 15 strings. When the `Bag` is empty there will be a centered message stating “No more items.” It will roughly look like the image below. Don't worry if it isn't exact.



What's a Window/Button/Bag?

On the moodle homepage, click on the Window API or the CarranoDataStructures API links to access the JavaDocs of their Application Programming Interface (API). Use these JavaDocs to familiarize yourself with the classes and their methods. Here you will find references for the `Window`, `Button`, `TextShape`, and `Bag` classes, which we will ask you to use. Navigate the classes in the API in the sidebar:


course
All Classes
Button
CircleShape
ClickAction
Command
Shape
SquareShape
TextFormat
TextShape
Widget
Window
WindowSide
PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | M

CS2114

Class Window

java.lang.Object
    CS2114.Window

```
public class Window
extends java.lang.Object
```

The Window class provides a simple Swing Window. The window class pr
already configured. The Window has a menu bar, four separate areas surr
around the center can contain any number of buttons for executing comm
of the area is up to the developer by using Shape objects.

Author:

loran, mapq, maellis

Constructor Summary

Constructors

Constructor and Description
`Window()` Default constructor, uses window.title from properties file for title, if a
`Window(java.lang.String title)`

Overview of Steps

1. Build a Window you will set up your project and classes and have a working window.

2. Create the Buttons you will have clickable Quit and Choose buttons.

3. Fill a Bag you build the back end so the bag is filled with strings.

4. Test your Bag you write and run your JUnit tests of your bag.

5. Display Bag's Contents your Choose button will display a string from the bag.

Classes

The above steps will be accomplished by three classes you will have to implement: `ShapeWindow`, `DisplayCollection`, and `ProjectRunner`. We describe their purpose here, then show you how to make them later.

The `ShapeWindow` class contains the Graphical User Interface (GUI) functionality for this project. The class contains five fields: a `Window`, two `Buttons`, a `TextShape`, and the `Bag` it is supposed to display. Its constructor sets up these fields, creates the basic layout of the GUI, and assigns the methods reacting to the buttons being clicked. The class also contains two separate methods containing the functionality for each button (their "event handlers").

The `DisplayCollection` class contains the functionality to create a randomly generated `Bag`. The class contains a constant reference array of strings that can serve as content of a `Bag`. The class's `Bag` field is filled with random content from the reference array in its constructor. This `Bag` can be accessed through the `getItemBag()` method (called a "getter").

The `ProjectRunner` class can be used to demonstrate the functionality of both the

`ShapeWindow` and `DisplayCollection` classes.

1. Build a Window (and start coding!)

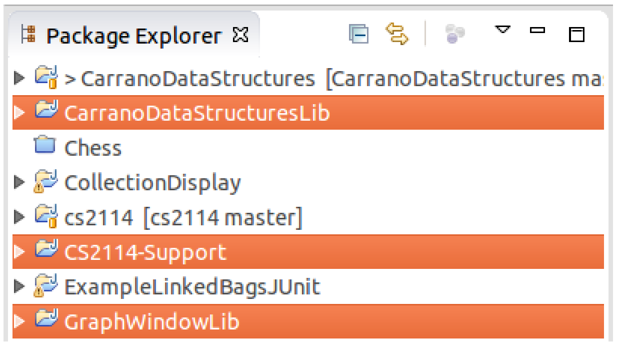
Initialize your CollectionDisplay Project.

In Eclipse, Click on File → New→ Java Project, and put `CollectionDisplay` in Project name.

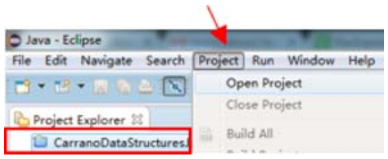
Download CarranoDataStructuresLib, CS2114-Support, and GraphWindowLib Packages.

If you don't already see CarranoDataStructuresLib, CS2114-Support, and GraphWindowLib listed in Package Explorer, download them under Project → Download Assignment... , where you'll find them under the Support folder.

Important: Those projects should be open as indicated by a grey triangle and open folder similar to the highlighted projects shown in the image below:



If the icon next to these projects appears to be a closed folder as shown in the image below, right click on the project tab and select "Open Project" or simply double-click on each project.



Add the downloaded projects to your new Project

Right click on your CollectionDisplay, and select Build Path → Configure Build Path... to open a window. Select the projects tab, click Add..., and select CarranoDataStructuresLib, CS2114-Support, and GraphWindowLib. Click OK. Click OK again.

Create a package in your new Project

Right click on your CollectionDisplay, and select New → Package... and name it "project1"

Build the ShapeWindow Class

Make your new ShapeWindow class by right-clicking on the package project1 and then selecting New → Class. Name it ShapeWindow and select the generate comments checkbox.

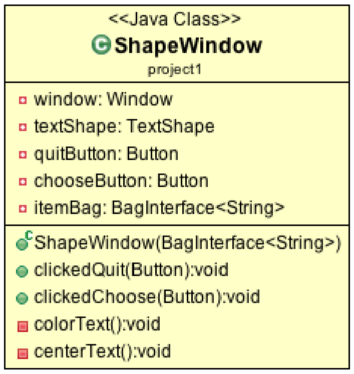
Copy import statements

The classes you write will require the following code. Copy and paste them after the package statement at the top of your class file.

```
import bag.Bag;
import bag.BagInterface;
import CS2114.TextShape;
import java.util.Random;
import CS2114.Window;
import CS2114.Button;
import CS2114.WindowSide;
```

This will give you access to all the classes you will need to implement the `ShapeWindow` class.

Fill out the fields and methods to match the class diagram – don't worry about implementing them yet.



Implement ShapeWindow's constructor

Initialize the window field inside the `ShapeWindow` constructor. To do this, call the default constructor of the `Window` class.

Set the title of window to "Project 1", using it's `setTitle(String title)` method.

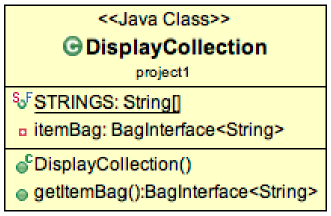
Initialize the itemBag field to point to your constructor's parameter.

Don't worry about initializing the remaining fields yet. We will get to that later!

Build the DisplayCollection Class

Create the `DisplayCollection` class the same way you created the `ShapeWindow` class.

Fill out the fields and methods to match the class diagram.



The STRINGS constant

We want the STRINGS field to be a constant that is available to any other class. To be able to do that, we have to do three things:

First, we have to make the field `public`, so that anyone can access the field.

Second, to avoid having to create a new instance of the `DisplayCollection` class every time we want to access the STRINGS constant, we need to make the field "belong" to the `DisplayCollection` class itself. To do that, we have to add the keyword `static` in front of the field. (You can do the same to any method for the same effect!)

Third, in order to prevent anyone from changing the content of the STRINGS constant, we have to add the `final` keyword in front of the field as well. The `final` keyword means that the variable can only be initialized once. This also means that we have to initialize the field immediately!

Initialize the STRINGS field to be an array that contains all of the possible strings in the bag. The strings can be: "red circle", "blue circle", "red square", and "blue square".

Note: When accessing the STRINGS constant, you need to write `DisplayCollection.STRINGS`.

Implement DisplayCollection's constructor

Initialize the itemBag field by using the default constructor of the `Bag` class. Since the `Bag` class is a "generic" class, you will have to use the "diamond operator":
`this.itemBag = new Bag<>();`

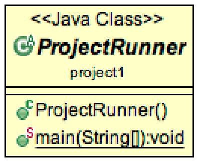
Implement the getItemBag method

Make `getItemBag()` return the itemBag field.

Build the ProjectRunner Class

Create the `ProjectRunner` class the same way you created the other two classes.

Fill out the fields and methods to match the class diagram. (You don't need to provide the default constructor for this class)



Implement ProjectRunner's main method

There is an example of using a `main` method in the `HokieClass` example from the first day of class.

Inside your `main(String[] args)` method for this project, create a new instance of the `DisplayCollection` class. Then, create a new `ShapeWindow` instance, passing the itemBag of your `DisplayCollection` instance into the constructor.

Now, click on Run → Run As → Java Application. If a window with the name Project 1 appears, you're ready to continue. You've just completed the first step of your project!

2. Create buttons

In this section we will create and implement buttons. Similar examples are available from our Class Downloads, under the Examples Folder.

Create the Quit Button for your Window

Now we're going to add the Quit Button.

In the ShapeWindow constructor:

1. Instantiate the quitButton field by using the `Button(String title)` constructor. Give it the String "Quit" for its name parameter.
2. Use the `onClick(Object object, String methodName)` method to set what method will be called when the quit Button is pressed. The arguments should be `this` for the object parameter and "clickedQuit" for methodName. This tells the quitButton to call your `clickedQuit(Button button)` method when it is clicked.
3. Use `addButton(Button button, WindowSide side)` on the window field to make the quitButton appear in the window. The quitButton should be on `WindowSide.NORTH`.

Try running your project again. You should now see a button on the upper side of your Window which says "Quit." Since we haven't implemented the `clickedQuit(Button button)` method yet, clicking the button will not work yet.

Implement your Quit Button

Implement the `clickedQuit(Button button)` method by calling `System.exit(int status)`. This method expects an integer parameter. Use zero as the parameter to indicate that no errors occurred. The quitButton should now close the window when clicked.

Create the Choose Button for your Window

In your constructor, initialize the chooseButton field with the title "Choose" the same way you initialized quitButton. Use the `onClick(Object object, String methodName)` method to specify that "clickedChoose" is called when the button is clicked. Add the chooseButton to the window on the `WindowSide.SOUTH`.

The choose button won't work yet, but you should now see it in your window when you run your program.

3. Fill a Bag

DisplayCollection's constructor

Make a new Random object variable named random and initialize it by using the default constructor of the `Random` class. You will use this variable to randomly generate numbers.

Use the `nextInt(int upperBound)` method to randomly generate a number. Generate a number from 5 to 15 (inclusive) which will determine your bag's size. Save that size in a variable which will be used next. Look at the API for Random to see how to use the `nextInt(int upperBound)` method. The code below gives you a starting point to work with.

```
Random random = new Random();
int count = random.nextInt(x) + y;
```

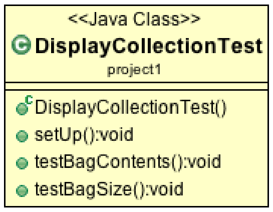
Build a `for` loop that adds new strings to the itemBag field based on the random number you generated. Randomly select one of the strings in the STRINGS array to add to the itemBag. Place each string into the itemBag by calling the `add(T item)` method.

4. Test that Bag!

Setup your DisplayCollectionTest

To check your itemBag's contents, set up a new test class named `DisplayCollectionTest` by right-clicking on the CollectionDisplay project and then following the guidelines from here. You will need to `import` packages similarly to how you did for `ShapeWindow` and `DisplayCollection`.

You must test two things: That only the four strings in the STRINGS constant are used in the bag and that the size of the bag varies correctly. Fill out your test class to match the diagram.



Note:

Since the `DisplayCollectionTest` class does not contain any fields, you can leave the `setUp()` method empty! However, make sure to provide a comment stating that the method was intentionally left empty!

Test your bag's size

In your `testBagSize()` method, you need to create an instance of the `DisplayCollection` class. You can call `getItemBag()` to get a reference to your `DisplayCollection`'s bag, and store that in a local variable.

Now that we have our bag, we need to check that its size is between 5 and 15. Call the `getCurrentSize()` method of the `Bag` class to get its size, and test that this number is between 5 and 15. This means that size 5, and size 15, are legal, but sizes 4 and 16 are not.

To test whether an Integer value like size is greater or less than some value, you can use the `assertTrue(boolean condition)` method. This method takes one parameter, and checks that it is `true`. To give this method a `boolean` parameter, you can use `size <= 15`. This comparison operator returns a `boolean` value.

To test that a bag isn't generated with too few or too many strings, generate a `DisplayCollection` several times. About 20 should do. You can use a for loop to do this. And every time the `DisplayCollection` is created, test its bag's size as mentioned above.

If this doesn't pass, make sure that your random number generator has been given the correct values. You should run your test a few times to make sure that it passes for multiple random values.

Test your bag's contents

For the `testBagContents()` method, create a `DisplayCollection` the same way you did for the `testBagSize()` method. In this test, we want to ensure that every `String` the bag contains is one of the four legal strings: "blue square", "red square", "blue circle", or "red circle" in the STRINGS constant. Make sure to use the STRINGS constant for testing!

Create a loop which iterates over the bag's content. Inside the loop, call the bag's `remove()` method to pull out a `String`. Assert that every `String` is one of the four legal strings using `assertTrue(boolean condition)`. When all of your tests pass, you're ready to move on to the final phase!

5. Display the Bag

Initialize the textShape field

Initialize the textShape field inside of the `ShapeWindow` constructor. `TextShape(int x, int y, String text)` takes three parameters. The first two are the `TextShape`'s coordinates in the `Window` and the third is the text it should display. Give your new `TextShape` an empty string for now and position it in the left-upper corner of the screen, at coordinates `(0, 0)`.

Use `addShape(Shape shape)` method of the window field to make the textShape appear in the window.

Implement ShapeWindow's clickedChoose method

Implement the `clickedChoose(Button button)` method to display the current item in the bag by calling `remove()` on the itemBag field. To put that `String` in our `TextShape`, call the `setText(String text)` method.

Try running your program. Your choose button should now work and display a new string from your bag every time it is clicked. (On some machines, calling the `setText` method will not make the text show up. In this case, try adding a call to `window.repaint()` at the end of the `clickedChoose` method. Remember to remove it once you change the color or position of the text, since those actions correctly trigger a repaint themselves.)

However, when the bag is empty, the program will crash! We will fix that next.

Handle the empty bag

You need to check that the bag is not empty before you call `remove()` on it! (Remember you can reference CarranoDataStructures API). Otherwise, you will get a `NullPointerException` when the bag is empty. This is because the bag's `remove()` method returns `null` once the bag has nothing to remove.

When the bag is empty, change the text of the textShape to "No more items."

Display red or blue text

Now that we have some text to display we can check if the text contains the words "red" or "blue". If so, pass in the appropriate `Color.BLUE`, or `Color.RED`, to our textShape field's `setForegroundColor(Color color)` method. If the text doesn't contain either word, remember to reset the color to `Color.BLACK`.

Implement this functionality in the `colorText()` helper method and call it from within the `clickedChoose` method.

Center the TextShape

Next use the `getGraphPanelWidth()` and `getGraphPanelHeight()` methods of the window field and the `getWidth()` and `getHeight()` of the textShape field to center the `TextShape` in the `Window`. Look through the JavaDoc of the Window API to figure out how to relocate the `TextShape`. Remember to check the class hierarchy for the different classes, as some functionality might be in a super class!

Implement this functionality in the `centerText()` helper method and call it from within the `clickedChoose` method.

Submission

Finally, Submit to WebCat! Visit office hours if you're still having trouble. Focus on getting these in order...

- Test Coverage (make your tests pass)
- Problem Coverage (Address hints given)
- Code Coverage(Test more code to get more hints)

Submit your work to Web-CAT by the due date and time. You can use the Submit Assignment feature in Eclipse to submit your project directly without leaving the Integrated Design Environment (IDE). You may submit as many times as you'd like before the project is due.

Grading Rubric

Your GUI checklist

- check for color and display, red or blue
- overwrite previous contents
- center the string
- end correctly
- Choose button
- Quit button

Web-CAT

- 15 pts Style on web cat
- 35 pts Test for bag contents on web cat, should be the four types, random on across multiple iterations, within the correct range.

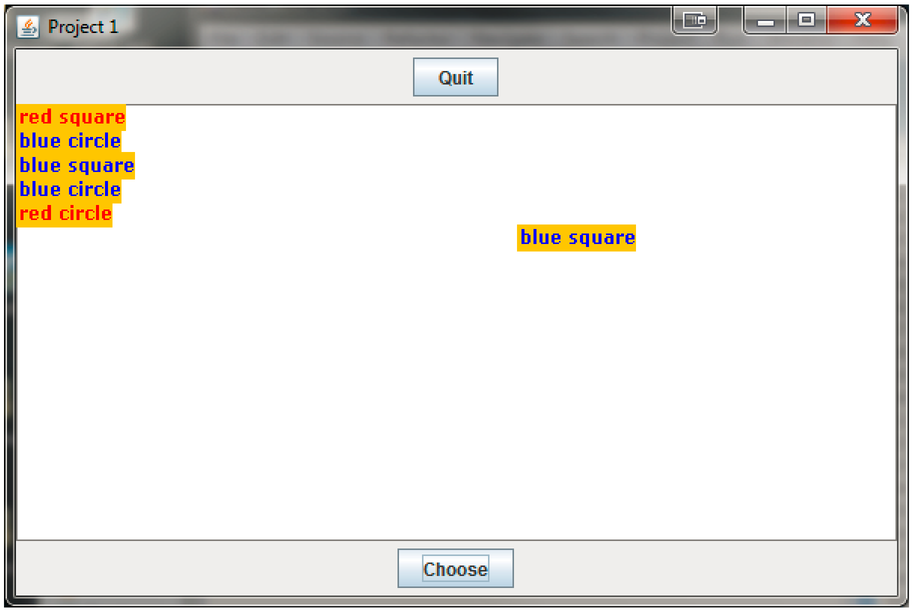
Program Grading Rubric

- 50 pts according to [Program Grading Rubric](put new URL here)

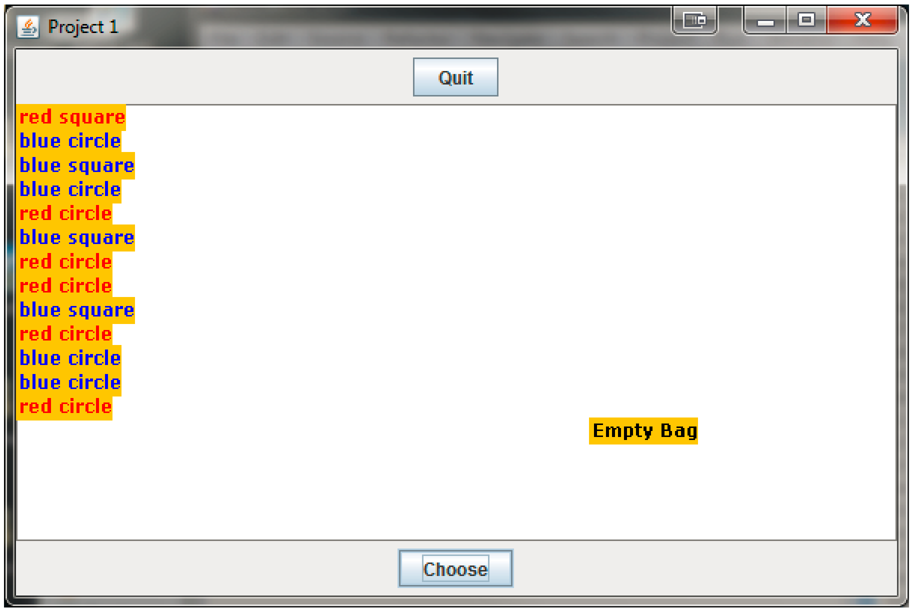
Optional: Cool Ideas

Display each removed String from the bag below the last removed String.

For even more fun, slide the TextShape from right to left.



```
red square
blue circle
blue square
blue circle
red circle
                                   blue square
```



```
red square
blue circle
blue square
blue circle
red circle
blue square
red circle
red circle
blue square
red circle
blue circle
blue circle
red circle
                                        Empty Bag
```

Submission status

Submission status	This assignment does not require you to submit anything online
Grading status	Not graded

You are logged in as Mykayla Fernandes (Logout)

CS 2114 Spr 2016