

Project 2 Specs

(updated November 4th, 1:20PM - new sections *highlighted in green*)

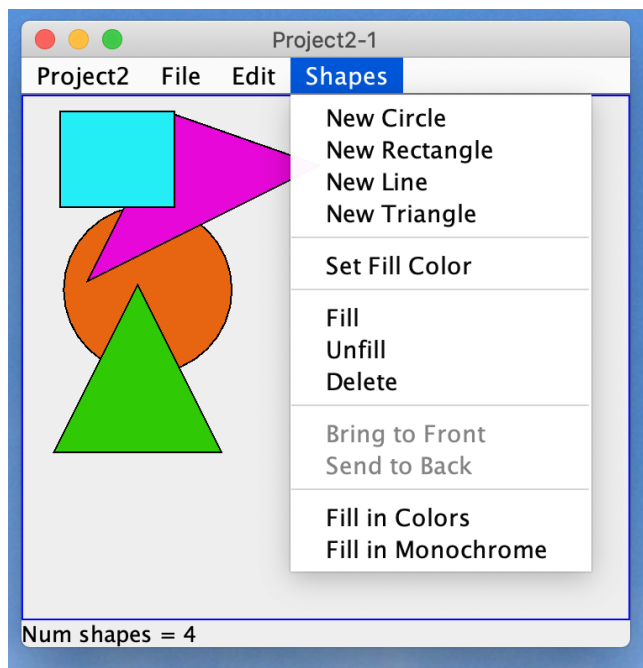
(updated November 4th, 10:50AM - new sections *highlighted in red*)

Goals

- Gain experience creating actions accessed through menus
- Gain experience managing selection

Description

In this project you will have to manage the selection of graphical shapes in the canvas and you will be able to implement commands that apply to the currently selected object(s). The image below shows a screenshot of the application.



Menus



The application will implement the following commands, all available in the **Shapes** menu as shown in the picture above.

- Install an action in Project2 > Quit to terminate the program, much like you did in Project1.
- Install an action in the File > New menu that creates a new window (**Project2**).
- Install an action in the File > Close that closes the current window by calling **closeWindow()**.
- New Circle - install an action that creates a new circle (**SCircle**)
- New Rectangle - install an action that creates a new rectangle (**SRectangle**)
- New Line - install an action that creates a new line (**SLine**)
- New Triangle - install an action that creates a new triangle (**STriangle**)

- Set Fill Color - install an action that opens a **JColorChooser** and let the user pick a default color. Store the resulting color.
- Fill - install an action that fills the selected shape(s) with the current default color.
- Unfill - install an action that changes the selected shape(s) to hollow (unfilled).
- Delete - install an action that deletes the selected shape(s)
- Bring to Front - install an action that brings the selected objects to the front (they display on top of the others)
- Send to Back - install an action that moves the selected objects to the back (they display behind the others)
- (Bonus) Fill in Colors - install an action that randomly assigns a color to each shape in the window (more on this below)
- (Bonus) Fill in Monochrome - same as color but use colors whose values are the same for Red, Green, and Blue.

The graphical shapes have been given to you already, all you need to do is to use them.

Selection

You must implement the bulk of the selection management. Study the **SFramework** examples added to Canvas (classes called `shapes1.java`, `shapes2.java`, etc.) for how to manage the selection. In particular, look at [shape5.java](#)  and [shape6.java](#) . through [shape9.java](#)

Basically you should create an **ArrayList<SShape>** where you will store all the shapes created. Note that the shapes classes mentioned above are all subclasses of **SShape**.

Functional Requirements

Menus

Note: the names of the menu items must be just as shown here, the testing code will look for those menu items by name and if you name them differently, it will cause the testing program to fail, thus delaying feedback.

1. Install a new menu called **Shapes**.
2. Install a new menu item under **Shapes** called “New Circle” and install an action that creates a new instance of **SCircle** and add it to the list of objects (**ArrayList<SShape>**). You should repaint the window and update the menus (**more on this below, in the Selection section**) after this is done. Repeat for all the shapes in the program, including “New Rectangle”, “New Triangle”, and “New Line”.
3. Install a new menu item under **Shapes** called “Set Fill Color” and install an action that calls the **JColorChooser**. If the user selects a color and clicks ok, then save the color in an instance variable. This becomes the “default color” from here on until a new default color is specified by the user.
4. Install a new menu item under **Shapes** called “Fill” and install an action that sets the fill attribute for the currently selected shape(s) to true and sets the fill color to be the default color (i.e., color selected with the **JColorChooser**). Make sure you force a repaint of the window after you set the fill attribute for the shape.
Note that **SShape** has defined **isFilled()** which returns true is the same has the attribute fill set to true; **fill(Boolean)** sets the fill attribute to the argument passed; and **fill(Boolean, Color)** sets the fill attribute to the argument passed and sets the color to be used for filling the shape. Note that the fill color can also be set independently via **setFillColor(Color)**.
5. Install a new menu item under **Shapes** called “Unfill” and install an action that clears the fill attributed (set it to false) for the currently selected shape(s). Make sure you force a repaint of the window.

6. Install a new menu item under **Shapes** called “Delete” and install an action that deletes the selected shape. Make sure you force a repaint of the window.
7. Install a new menu item under **Shapes** called “Bring to Front” and install an action that brings the selected shape(s) to the front. More on this command below in the Selection section. Make sure you force a repaint of the window.
8. Install a new menu item under **Shapes** called “Send to Back” and install an action that sends the selected shape to the back. More on this command below in the Selection section. Make sure you force a repaint of the window.
9. **Add a method `public int numShapes()` to your `Project2` class. This routine should return the number of shapes you have in your `ArrayList` list. This is required for the Web-CAT grader to grade your project.**

Selection

Study the **shapes** examples in the **SFramework** documentation pages for more details. In particular, look at [shape5.java](#)  and [shape6.java](#) , through [shape9.java](#)


You must store all of the shapes created via menus (above) into an `ArrayList<SShape>`. When you draw the contents of the array, you will be drawing them in a particular order so that some shapes occlude (cover) others. This is dictated by the order in which the shapes are stored in the array. In the screen shot shown above, the circle appears behind the other objects. Thus it is drawn first.

Shapes that are selected are drawn with a handle (small black square). This is already implemented in the **SShape** class and its subclasses. When the user clicks on a shape, all you need to do is set that shape to be selected. See [shape4.java](#) for an example.

For commands that apply to the selected object(s) (e.g. **Fill**, **Unfill**, and **Delete**), you must loop over your array and apply the command to each object that is selected. The **SShape** class stored the select property for each object. You can manage it via `isSelected()` which returns true if the object is selected, and via `select(Boolean)` which sets the selected attribute to the value passed in the call.

If the object at the beginning of the array is drawn first, and thus appears behind all the other objects, then the **Bring to Front** command should move an object to the end of the array. The opposite is true with the **Send to Back** command. It should move the selected object(s) to the front of the array because the element in position 0 is drawn first and appears behind all the other drawn objects.

One last point worth noting. You cannot modify an `ArrayList` while iterating over it. But there are three commands that require you to do just that in the project (**Bring to Front**, **Send to Back**, and **Delete**). One way to implement this is to iterate over the array, identify the objects that have to be moved and save them in a second `ArrayList`. Then iterate over this second list and make the changes on the selection list.

You should enable/disable menu items based on the selection of objects. For example, if you have no selection (0 objects selected), then you need to disable menu commands that act on selected objects. The Delete menu item, for example, deletes the selected object. If you have no selection, the Delete command should be disabled. Some commands are available all the time (e.g., New...) so they should always be enabled. Some require a selection. Study [shape6.java](#)  for an example of how to update menus.

Submission

You must submit to Web-CAT the **Project2.java** file and any other files needed to run the project.

You do not need to submit the **SFramework.jar**. It is already stored in Web-CAT.

1. Document your code, it makes it easier to read, easier to grade, and easier for you when you revisit it for future projects. Follow the [CCI Style Guide for Java](#).
2. Do not put your code in any user-defined packages (you can use existing Java packages). The projects in this class are not big enough to merit the use of packages. Using packages in these projects complicates the testing unnecessarily. Note this is not true for commercial software development, but certainly true for me trying to grade your code and finding that everybody used a different name for the main package.
3. Stick to the names mentioned here. Following the right naming conventions is essential for me to be able to give you feedback quickly.

Bonus Points

1. (Bonus - optional) Install a new menu item under **Shapes** called “Fill in Colors” and install an action that generates random colors and sets each shape to a different randomly selected color. You can use **Random** class to generate random numbers. Just generate a random number the **nextInt(upperbound)** method call. If you call this with 255 as parameter, you are guaranteed to get a value between 0 and 254. Generate three such values and use them for red, green, and blue in a **new Color()** object.
2. (Bonus - optional) Install a new menu item under **Shapes** called “Fill in Monochrome” and install an action that generates random monochrome colors and sets each shape to a different randomly selected color. Use the **Random** class as in explained above, but set RGB to the same randomly generated integer between 0 and 255. This will give you different shades of gray (i.e., monochrome).

TLDR

- Make sure you use [this version of the SFramework.jar](#)
- [Project 2 Spec](#)
 - Video of Running Project 2 & Automated Testing of Project 2 available on the [assignment page](#).
- [CCI Style Guide for Java](#)
- [JGrasp](#)
- [\(Links to an external site.\)](#)
-
- [JGrasp Video for 4440](#)
- [Video with examples of SFramework](#)
 - Newer videos with examples of SShapes (COMING SOON)
- [Document \(PDF\) with examples of SFramework](#)
- [Browse the Javadoc for SFramework](#)

the end