Assignment List vs Set with GUI

CSIS-1410

Learning Objectives :

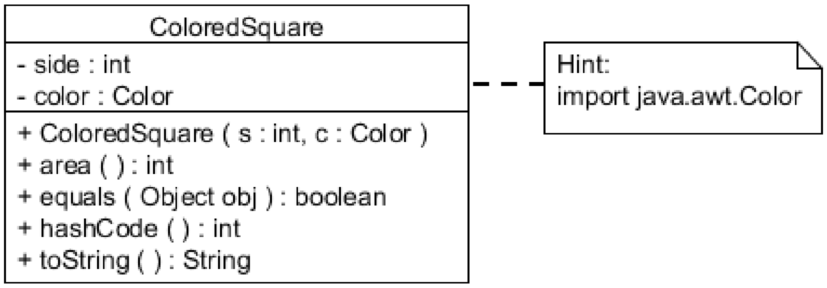
- Implement equals and hashCode to compare objects of a custom class.
- User the internet to learn about the interfaces List and Set, and class HashSet
- Describe the differences between the interfaces List and Set
- Create, initialize, and display a HashSet
- Modify an existing GUI application based on specific instructions
- Demonstrate the benefits of separating functionality and display by using class ColoredSquare both in a console application and in a GUI application

Description:

Download the starter project and import it into Eclipse.
Import > Existing Projects Into Workspace
Implement the classes `ColoredSquare` and `ListVsSetDemo` as described below. Uncomment the code in class `ListVsSetConsoleApp` to test your classes. This test client should **not** be changed.
Once you verified that the classes `ColoredSquare` and `ListVsSetDemo` work as expected modify the Gui Application according to the description below.

Ad ColoredSquare:

- Implement ColoredSquare according to the UML diagram. No class members should be added or removed.
  Important: ColoredSquare should **NOT** include any print methods



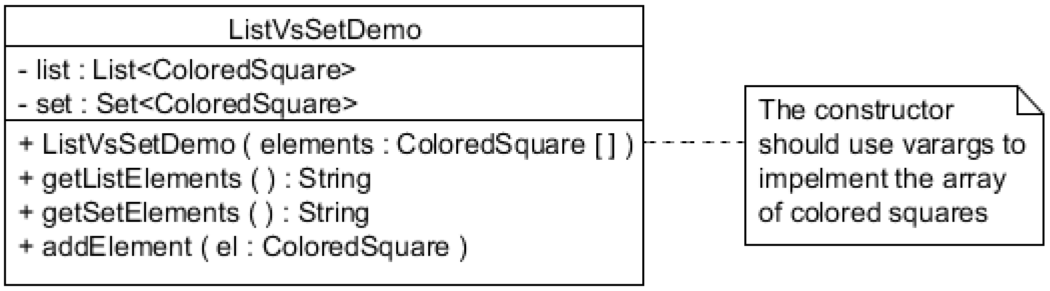
- **equals:**
  2 colored squares should be considered equal if and only if they have the same size and color. Use Eclipse to auto-implement equals and hashCode
- **toString:**
  The method toString should return a string as shown on the right.
  e.g. `side:14 #0000FF   side:12 #FFFF00   etc.`
  Notice: the toString method of Color displays the rgb values in decimal format. However, in our implementation we use a hexadecimal format like in HTML.
  Hint 1: To obtain the individual red, green, blue values check out class Color .
  Hint 2: Info on how to format an integer as a 2-digit hexadecimal value

Console Output:

```
List:
side:14 #0000FF
side:18 #FF0000
side:12 #FFFF00
side:18 #FF0000
side:16 #00FF00

Set:
side:14 #0000FF
side:12 #FFFF00
side:16 #00FF00
side:18 #FF0000


Adding a new element:
List:
side:14 #0000FF
side:18 #FF0000
side:12 #FFFF00
side:18 #FF0000
side:16 #00FF00
side:10 #FFC800

Set:
side:14 #0000FF
side:12 #FFFF00
side:10 #FFC800
side:16 #00FF00
side:18 #FF0000


Adding a duplicate
element:
List:
side:14 #0000FF
side:18 #FF0000
side:12 #FFFF00
side:18 #FF0000
side:16 #00FF00
side:10 #FFC800
side:10 #FFC800

Set:
side:14 #0000FF
side:12 #FFFF00
side:10 #FFC800
side:16 #00FF00
side:18 #FF0000
```

Ad ListVsSetDemo:

- Implement ListVsSetDemo as specified in the UML diagram.
  Important: ListVsSetDemo should **NOT** include any print methods


ListVsSetDemo
- list : List<ColoredSquare>
- set : Set<ColoredSquare>
+ ListVsSetDemo (elements : ColoredSquare [ ])
+ getListElements ( ) : String
+ getSetElements ( ) : String
+ addElement (el : ColoredSquare)
The constructor should use varargs to impelment the array of colored squares

- Notice: even though the parameter of the constructor is defined as an array of type ColoredSquare, I want you to implement it as **varargs**. This is a good review and it provides a convenient way to call the constructor. Here is a short video in case you'd like a refresher on varargs:

- **constructor**:
  The constructor uses the elements passed to initialize both the list and the set

- **getListElements, getSetElements:**
  return a formatted string containing all the elements of the collection - one element per line. (see Console Output)
  Hint: use StringBuilder to create that string

- **addElement:**
  Adds the element passed to both the list and the set

Ad ListVsSetGuiApp:

Just as the console app calls methods of class ListVsSetDemo I want you to call methods of ListVsSetDemo to generate the text that will be displayed in ListVsSetGui when the ***Demo*** tab is selected.

The starter project includes two radio buttons. Add a third radio button that allows adding a new element. Just like in the console app it needs to be an element that has not been included in the original list. Every time this third radio button is selected, that same element is displayed and added to both the list and the set.
Important: Make sure that at any time only one of the radio buttons can be selected.
Also: Every time the List Elements or Set Elements radio button is selected the updated list / set should be displayed.

When the user selects the ***ListVsSet*** menu item the two main differences between the interfaces List and Set should be displayed. Hint: Refer to the Java API to learn about the differences between interface `List` and `Set`.

When the user selects the ***Exit*** menu item, the application should close.

Turning in:

Ensure to include your name on top of each java file that you modified.
Create a runnable jar file and submit in via Canvas