**Project 3**

**Due date: Friday, October 18**

**Pair project**

Use[**this Google Form**](https://forms.gle/j5EeV5LZshEV612r6)to identify a Project 3 Pair partner by September 24; if you can't identify a partner by then, your instructor will assign a Pair member for you for this project.

**Do not create a GitHub repository for Project 3 until you have been assigned to a Pair. At that time only one of the Pair will create the repository and share it with their pair mate and with me at** [**ryacobellis313@msn.com**](mailto:ryacobellis313@msn.com) **and our TA.**

**Objectives**

An understanding of

* Requirements analysis
  + functional requirements
  + nonfunctional requirements
* Modeling
  + UML class diagrams
  + UML sequence diagrams
* Design principles and patterns
  + Dependency inversion principle
  + Composite pattern
  + Decorator pattern
  + Visitor pattern
* Java coding
  + final-correctness
  + @Override correctness
  + anonymous inner classes
  + basic Android graphics
  + mock-based testing using Mockito

**Description**

In this project, you will complete the implementation of the simple API for drawing graphical shapes. The API design is based on the requirements from an in-class group activity.

See also these examples: [expressions](https://github.com/LoyolaChicagoCode/misc-java/blob/main/src/main/java/expressions/SimpleExpressions.java) (how to process simple arithmetic expressions) and [vexpressions](https://github.com/LoyolaChicagoCode/misc-java/blob/main/src/main/java/vexpressions/VisitorExpressions.java) (how to do the same with visitors / the Visitor design pattern).

**Functional requirements**

Start with this code skeleton: [**https://github.com/LoyolaChicagoCode/shapes-android-java/**](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2FLoyolaChicagoCode%2Fshapes-android-java%2F&sa=D&sntz=1&usg=AOvVaw0ltcwzDP9iC1ZBz5GLZ1ky). The functional requirements are embodied in the JUnit tests in the **test** folder; instructions for running the tests are included in the **README** file and in the Notes below. When your code passes all the tests, you will have fulfilled the functional requirements for grading purposes. If some of the tests do not pass, you will receive partial credit. In addition, your app should produce an output that looks like this screenshot: [**http://t.co/gaYmzRki**](http://www.google.com/url?q=http%3A%2F%2Ft.co%2FgaYmzRki&sa=D&sntz=1&usg=AOvVaw1qTmGVw_wjbwHain_3rOO6)when run on an emulator or Android device. Note that the color of the polygon has changed slightly to blue.

**Notes**

* It's unlikely this will happen, but just in case ... be sure to click any link that comes up to connect the Project to Gradle, otherwise it may not be connected to Gradle properly. On the other hand, **do not click any link that says you should update your Gradle version!**
* See below under **How to submit** for how to name your imported GitHub repository (this uses the standard course naming convention).
* The **README** file tells you to run **./gradle testDebug** in an IntelliJ IDEA **Terminal** **Window** to see the test results. You should also be able to change to the **Android** view and then under **Java** right-click the **test** item to run the tests.

**Nonfunctional requirements**

Specifically, complete the code in the various Java source files within the **src** folder. Look in the IntelliJ IDEA **TODO** view for sections marked as **TODO** or **FIXME** and use the test cases as your guide. Besides other minor tasks, the main implementation tasks are:

* Implementing the **Count**, **BoundingBox**, and **Draw** visitors
* Implementing the missing classes **StrokeColor**, **Outline**, **Point**, and **Polygon**
  + The **StrokeColor** decorator indicates the foreground color for drawing its **Shape**.
  + The **Outline** decorator does the opposite of the **Fill** decorator: it indicates that its **Shape** should be drawn in outline (default) mode.
  + A **Point** is a **Location** without a **Shape**. You should implement it using a **Circle** with radius 0 as its **Shape** and override any methods as needed.
  + A (closed) **Polygon** is a **Shape** defined by a list of **Points**; the last **Point** should be connected to the first one to close the **Polygon**. Implement it as a special case of **Group**.

**You must not make any other changes to the code skeleton or the test cases.**

**Go on to the next page for Grading and other information.**

**Grading**

* 0.5 code skeleton intact
* 1.0 Count
* 2.0 BoundingBox
* 2.0 Draw
* 1.0 StrokeColor
* 1.0 Outline
* 0.5 Point
* 1.0 Polygon
* 1.0 output from running your app on an emulator or real device should match the drawing shown in the link in the Functional Requirements section above (note: the **Polygon’s** outline color has been changed from the color in that drawing to make it more visible)

**10 points total**

**How to submit**

As the first step in working on this project, you or your pair-mate will import the code skeleton into a private GitHub repository shared between the two of you and with your instructor and TA. The name of the ***repository*** should be cs313413f24pairNp3, ***where N is your Pair number found in Sakai***, and the p3 part refers to Project 3. When your work is ready to be graded, please notify your instructor and TA **via a Sakai Assignment comment or a Piazza post, including your GitHub repo HTTPS URL**.

**IntelliJ IDEA-related changes – important for Android!**

**Read the additional Word document which describes key changes to IntelliJ IDEA in order for IntelliJ to successfully run Android projects. You may have done some of those steps already when you set up IntelliJ for Project 1, so you won’t need to repeat any that are already done.**