A simple paint application

Student: Grecu Razvan-Stefan
Group: 30232

Table of Contents

.....1

Table of Contents.....2

1. Requirements Analysis 3
    1.1Assignment Specification.....3
    2 Non-functional Requirements.....3

2. Use-Case Model.....3

3. System Architectural Design.....4

4. UML Sequence Diagrams.....5

5. Class Design.....6

.....8

6. Data Model 8

7. System Testing.....9

8. Bibliography.....10

1. Requirements Analysis

1.1 Assignment Specification

Use appropriate Java/C# classes for Drawing and Drag & Dropping to design and implement an application similar with Paint. The application should implement the following actions: (1) draw complex shapes using simple shapes and (2) drag and drop simple shapes from a toolbar.

Choose and use an architectural pattern in designing the application. Use at least two design patterns: the Composite design pattern and another one at your choice (not an already used one). As a second design pattern, I have chosen the Command pattern, as most of my actions within the paint application could be thought out as commands.

1.2 Functional Requirements

The implemented functionalities of this small application are the selection and drawing of different primitive shapes, such as circle, rectangle, line and also a special shape, that can be formed by playing with the primitive ones. In addition there have been created two buttons with the soul purpose of increasing functionality, their purpose being undo-ing and redo-ing the last drawing commands of the user.

2 Non-functional Requirements

Non-functional requirements impose constraints on the services and functionalities of the system; such constraints could be : temporal constraints, development constraints, imposed standards etc. Depending on the situation, non-functional requirements may be more critical than functional ones. If these requirements are not met, the system will not be useful for the purpose it was developed.

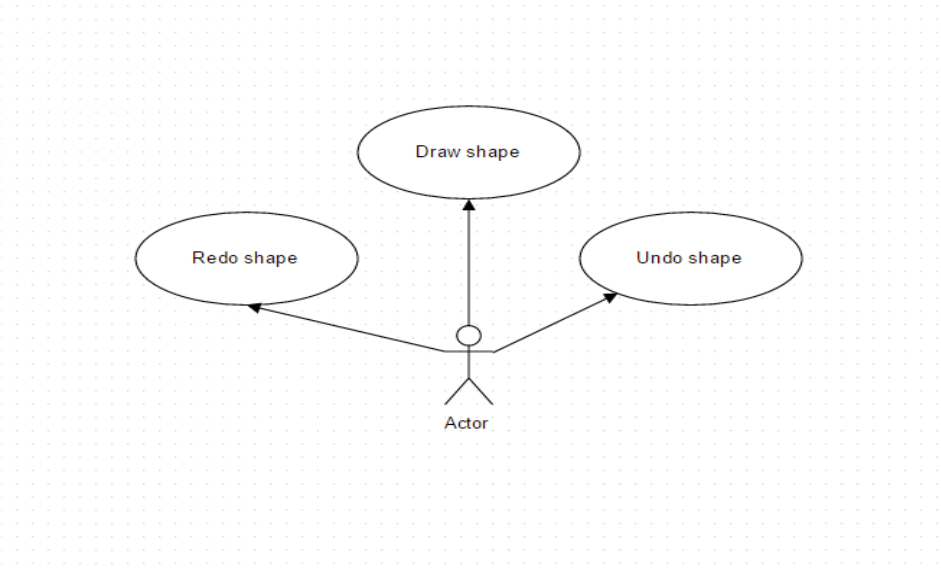
The conventional I/O mechanism for the application must have:

- Java compliant device, to run the application
- Basic I/O devices, such as a keyboard and/or mouse, as the user must offer data as input, and a monitor as he expects an outcome.

The availability of an application is tightly coupled to its feasibility. If an application is not readily-available when it is needed, it is unlikely that it fulfills the role for which it was created.

2. Use-Case Model

Use case: Draw a complex shape
Level: any
Primary actor: any user
Main success scenario: one clicks on the design complex shape button; a new frame appears on which the actor draws his complex shape and adds it to the original canvas.
Use case: Undo a (complex) shape
Level: any
Primary actor: any user
Main success scenario: the actor wants to create a house; his actions are interrupted and accidentaly creates an extra figure. He will press undo to destroy it.
Extension : if the user removes the last figure, he will not be able to recreate it.



3. System Architectural Design

3.1 Architectural Pattern Description

The concept of Model-View-Controller was originally invented by Smalltalk programmers (TrygveReenskaug). In his very own words, he describes the components of these patterns as follows:

Models

- they represent knowledge
- could be a single object/structure of objects
- There should be a 1-to-1 correspondence between the model and its

parts and the represented world as perceived by the owner of the model.

- It is considered confusing and a bad form, mixing problem-oriented nodes with implementation details

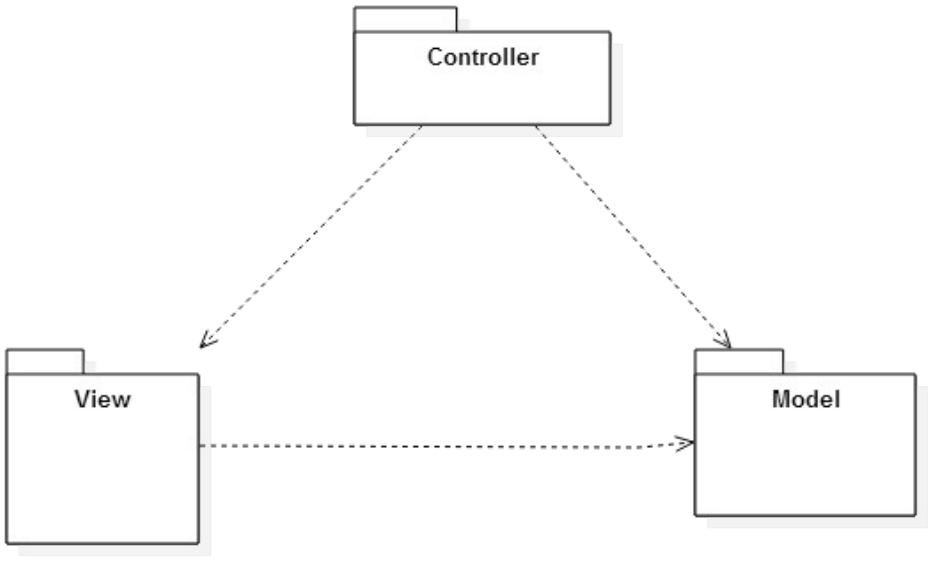
Views

- A visual representation of the model.
- Highlights certain attributes of the model and suppresses others
- *Presentation filter*
- It is attached to a model and gets data by asking it questions

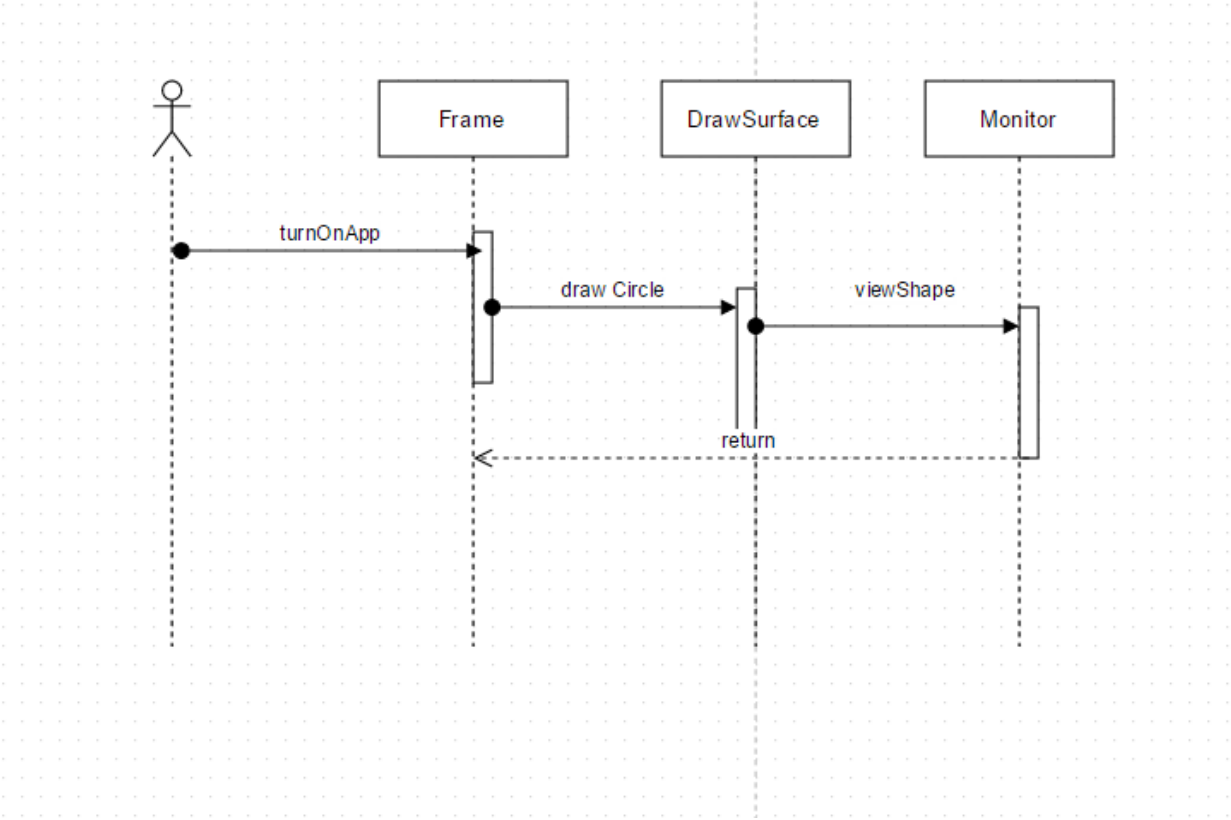
Controllers

- Link between a user and the system
- Provides the user with the input by arranging for relevant views to present themselves in appropriate places on the screen
- Provides means for user output by presenting the user with menus or other means of giving commands and data.
- It should never supplement views.

3.2 Diagrams



4. UML Sequence Diagrams



5. Class Design

5.1 Design Patterns Description

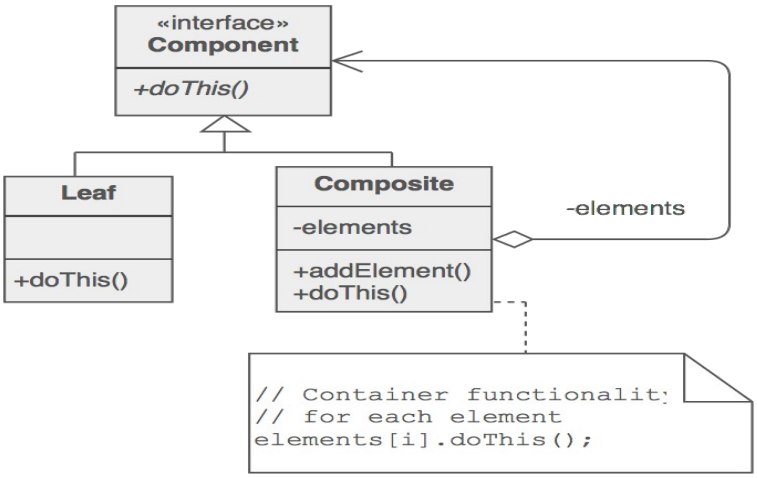
Composite pattern is used where we need to treat a group of objects in similar way as a single object. Composite pattern composes objects in term of a tree structure to represent part as well as whole hierarchy. This type of design pattern comes under structural pattern as this pattern creates a tree structure of group of objects.

This pattern creates a class that contains group of its own objects. This class provides ways to modify its group of same objects.

Compose objects into tree structures to represent whole-part hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

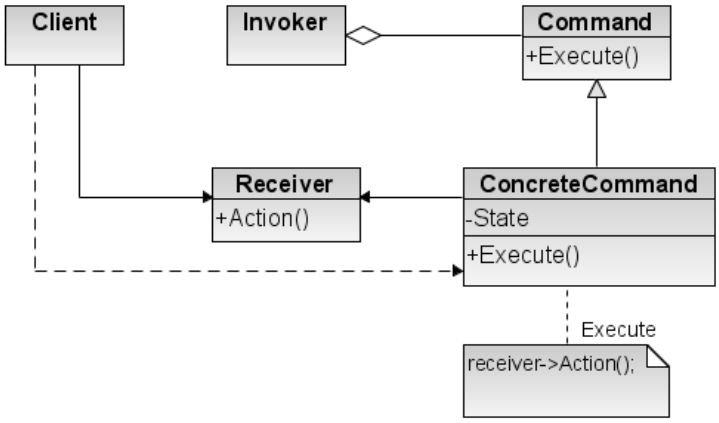
- Recursive composition

- Directories contain entries, each of which could be a directory."
- 1-to-many "has a" up the "is a" hierarchy

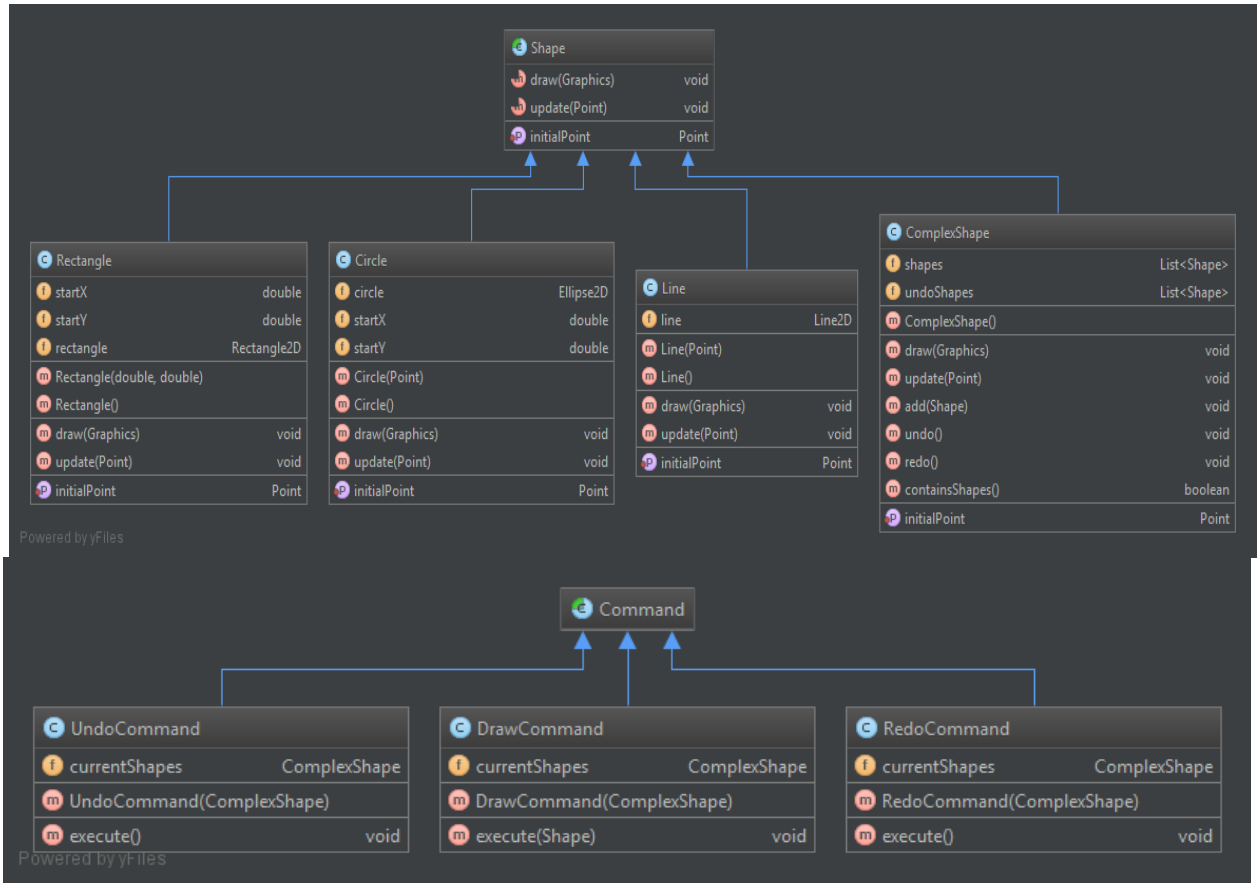


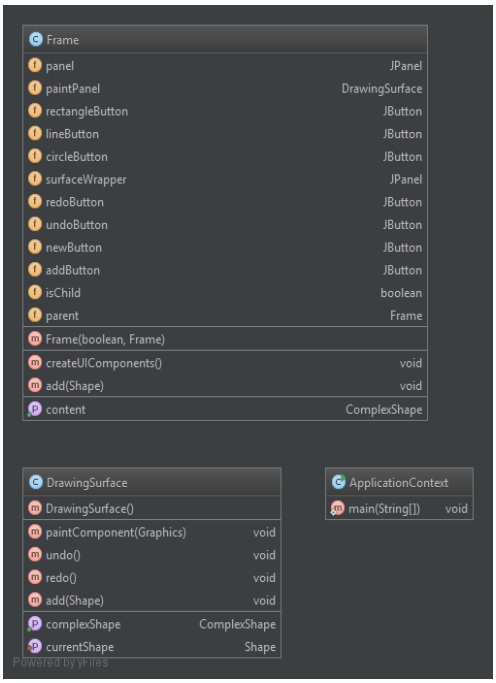
Command pattern is a data driven design pattern and falls under behavioral pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

- Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.
- Promote "invocation of a method on an object" to full object status
- An object-oriented callback

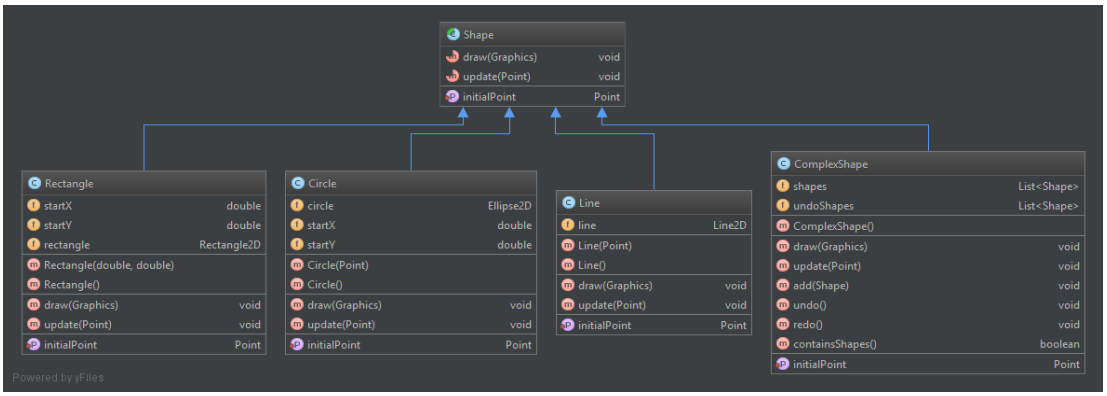


5.2 UML Class Diagram





6. Data Model



7. System Testing

[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]

8. Bibliography

1. http://java.sun.com/docs/books/tutorial/uiswing/
2. http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html
3. http://www.exampledepot.com/egs/?
4. http://msdn.microsoft.com/en-us/library/54xbah2z(VS.80).aspx
5. http://msdn.microsoft.com/en-us/library/system.drawing.graphics.aspx
6. http://download.oracle.com/javase/tutorial/uiswing/dnd/index.html
7. http://msdn.microsoft.com/en-us/library/aa984430%28v=vs.71%29.aspx