软件设计报告：图形渲染系统的设计与实现

---

1. 需求分析

核心功能需求：

1. **图形渲染**
   - 支持基本图形：圆形、矩形、三角形、椭圆
   - 支持多种渲染引擎：Swing（本地图形界面）、SVG（生成 SVG 标签）、Legacy（旧版适配器）
   - 渲染操作：绘制、移动、撤销/重做
2. **设计模式应用**
   - **创建型**
     - `Factory`：创建基本图形（`ShapeFactory`）
     - `Singleton`：全局配置管理（`GlobalConfig`）
   - **结构型**
     - `Adapter`：适配旧版渲染器（`LegacyRendererAdapter`）
     - `Bridge`：分离渲染接口与实现（`Renderer` 和 `SwingRenderer`）
     - `Proxy`：远程渲染代理（`RemoteRendererProxy`）
   - **行为型**
     - `Command`：图形操作命令（`AddShapeCommand`, `MoveShapeCommand`）
     - `Visitor`：图形导出为 JSON（`JSONExportVisitor`）
     - `Observer`：图形变化监听（`ShapeObserver`）
3. **扩展功能**
   - 序列化/反序列化：通过 Gson 保存/加载图形（`PersistenceManager`）
   - 撤销/重做：操作历史管理（`UndoManager`）

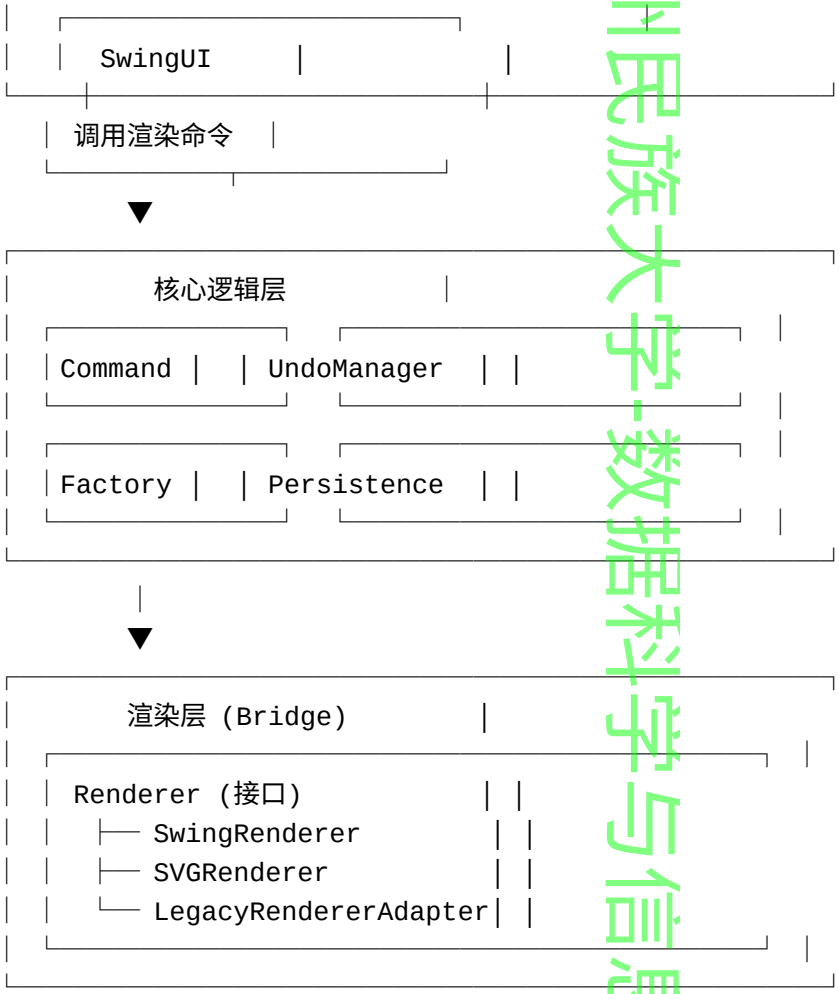
---

2. 总体设计

体系架构图

text

```
┌──────────────────────────────────────────────┐
│              客户端 (UI)                     │
```

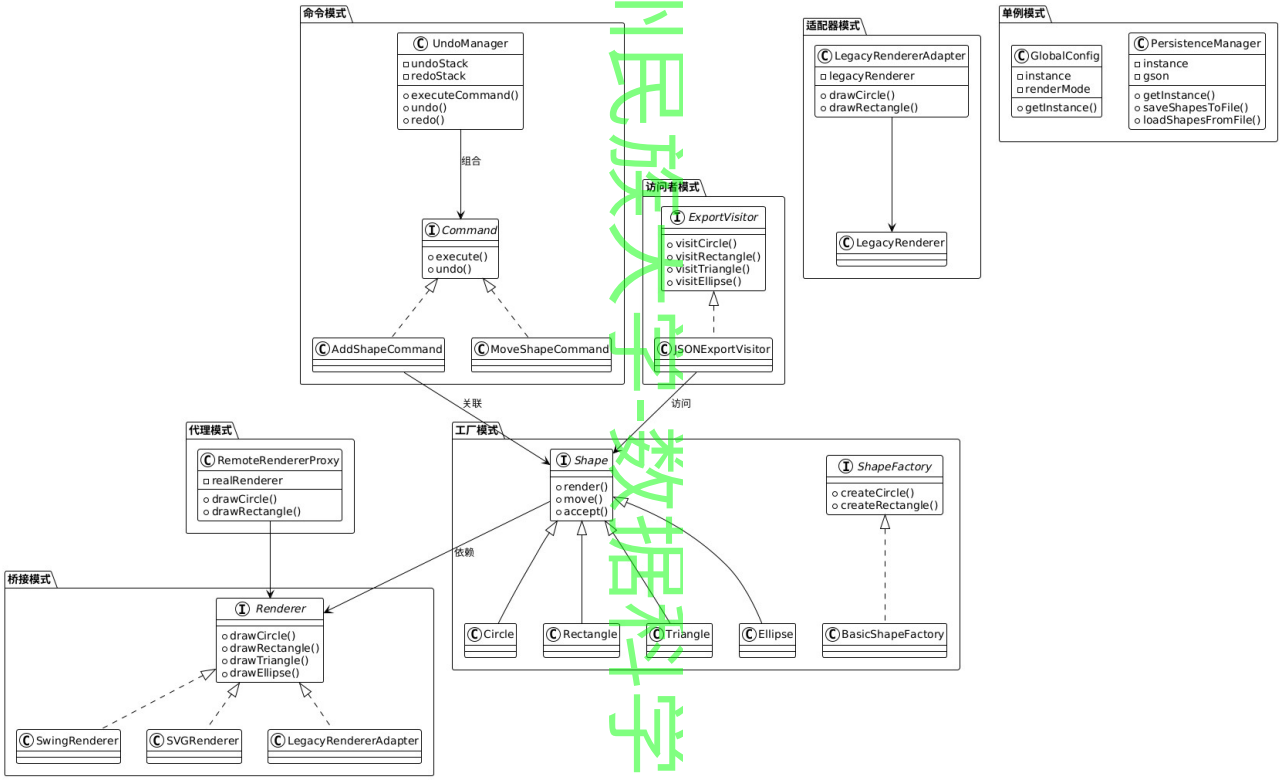


- **UI 层**：`SwingUI` 提供图形界面，触发命令操作。
- **核心逻辑层**：
  - `Command` 执行图形操作，`UndoManager` 管理操作历史。
  - `Factory` 创建图形对象，`Persistence` 处理序列化。
- **渲染层**：`Renderer` 接口统一渲染行为，具体实现由桥接模式分离。

---

3. 详细设计

3.1 类图



plantuml

```
@startuml
!theme plain

package "工厂模式" {
  interface Shape {
    + render()
    + move()
    + accept()
  }
  class Circle
  class Rectangle
  class Triangle
  class Ellipse
  Shape <|-- Circle
  Shape <|-- Rectangle
  Shape <|-- Triangle
  Shape <|-- Ellipse

  interface ShapeFactory {
    + createCircle()
    + createRectangle()
  }
  class BasicShapeFactory
  ShapeFactory <|.. BasicShapeFactory
}

package "桥接模式" {
```

```
  interface Renderer {
    + drawCircle()
    + drawRectangle()
    + drawTriangle()
    + drawEllipse()
  }
  class SwingRenderer
  class SVGRenderer
  class LegacyRendererAdapter
  Renderer <|.. SwingRenderer
  Renderer <|.. SVGRenderer
  Renderer <|.. LegacyRendererAdapter
}

package "命令模式" {
  interface Command {
    + execute()
    + undo()
  }
  class AddShapeCommand
  class MoveShapeCommand
  Command <|.. AddShapeCommand
  Command <|.. MoveShapeCommand

  class UndoManager {
    - undoStack
    - redoStack
    + executeCommand()
    + undo()
    + redo()
  }
}

package "访问者模式" {
  interface ExportVisitor {
    + visitCircle()
    + visitRectangle()
    + visitTriangle()
    + visitEllipse()
  }
  class JSONExportVisitor
  ExportVisitor <|.. JSONExportVisitor
}

package "代理模式" {
  class RemoteRendererProxy {
    - realRenderer
    + drawCircle()
    + drawRectangle()
  }
  RemoteRendererProxy --> Renderer
```

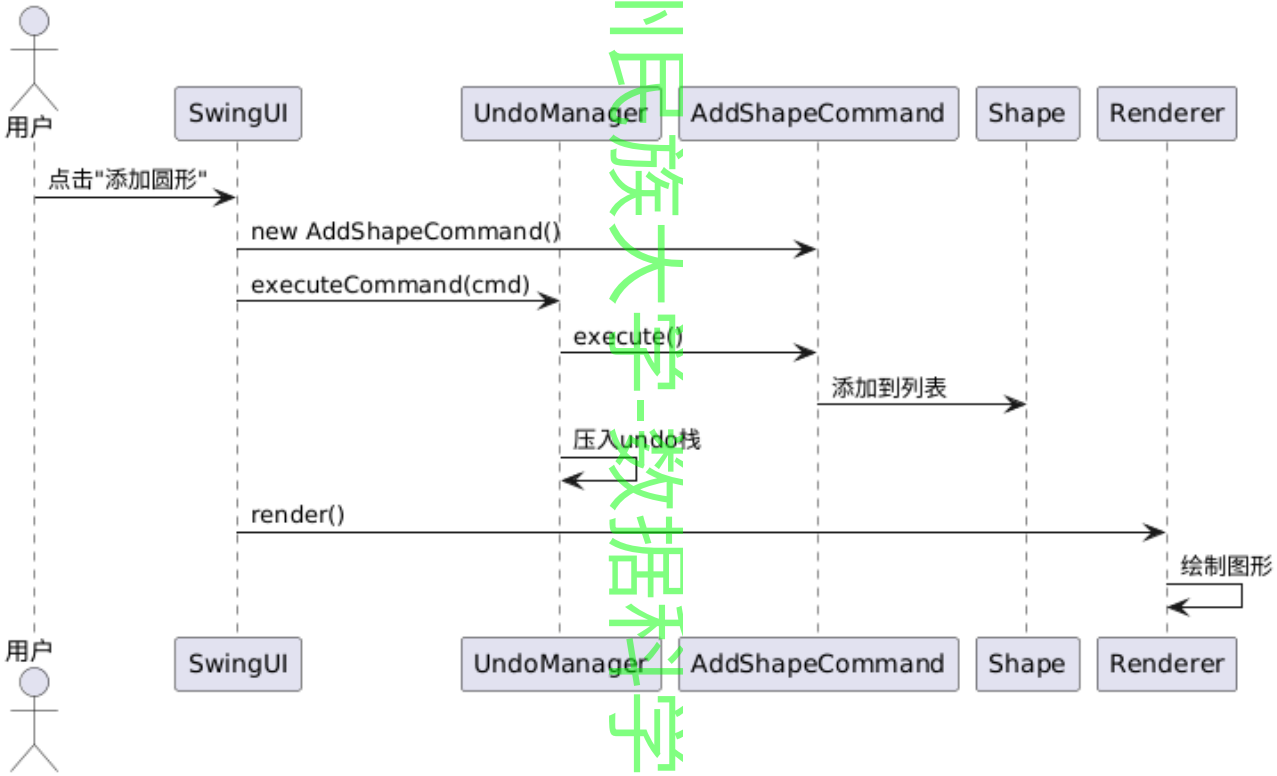
```
}

package "适配器模式" {
  class LegacyRendererAdapter {
    - legacyRenderer
    + drawCircle()
    + drawRectangle()
  }
  LegacyRendererAdapter --> LegacyRenderer
}

package "单例模式" {
  class GlobalConfig {
    - instance
    - renderMode
    + getInstance()
  }
  class PersistenceManager {
    - instance
    - gson
    + getInstance()
    + saveShapesToFile()
    + loadShapesFromFile()
  }
}

Shape --> Renderer : 依赖
AddShapeCommand --> Shape : 关联
UndoManager --> Command : 组合
JSONExportVisitor --> Shape : 访问
@enduml
```

3.2 顺序图：添加图形并渲染

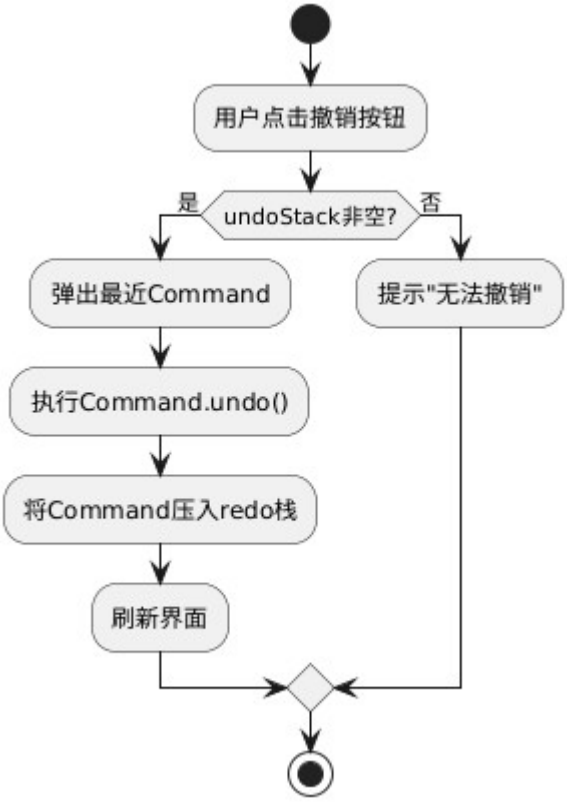


plantuml

```
@startuml
actor 用户 as User
participant "SwingUI" as UI
participant "UndoManager" as Undo
participant "AddShapeCommand" as Cmd
participant "Shape" as Shape
participant "Renderer" as Renderer

User -> UI : 点击"添加圆形"
UI -> Cmd : new AddShapeCommand()
UI -> Undo : executeCommand(cmd)
Undo -> Cmd : execute()
Cmd -> Shape : 添加到列表
Undo -> Undo : 压入undo栈
UI -> Renderer : render()
Renderer -> Renderer : 绘制图形
@enduml
```

3.3 活动图：撤销操作流程



plantuml

```
@startuml
start
:用户点击撤销按钮;
if (undoStack 非空?) then (是)
  :弹出最近 Command;
  :执行 Command.undo();
  :将 Command 压入 redo 栈;
  :刷新界面;
else (否)
  :提示"无法撤销";
endif
stop
@enduml
```

---

4. 设计模式应用总结

模式	实现类	作用
Factory	`BasicShapeFactory`	解耦图形创建过程
Singleton	`GlobalConfig`	全局配置统一访问点
Adapter	`LegacyRendererAdapter`	兼容旧版渲染器
Bridge	`Renderer` + `SwingRenderer`	分离抽象（渲染接口）与实现（具体引擎）
Proxy	`RemoteRendererProxy`	控制远程渲染访问
Command	`AddShapeCommand`	封装图形操作，支持撤销/重做
Visitor	`JSONExportVisitor`	扩展图形导出功能
Observer	`ShapeObserver`	监听图形变化（预留扩展点）

---

5. 测试与部署

单元测试覆盖:

- `Command` 模式：测试撤销/重做逻辑正确性。
- `PersistenceManager`：验证图形序列化/反序列化。
- `Renderer`：检查各引擎渲染输出是否符合预期。

部署说明：

1. 环境要求：Java 11+
2. 运行入口：`MainUI.java`
3. 数据保存：图形列表默认保存为 JSON 文件

---

6. 用户手册

操作流程：

1. 启动应用：运行 `MainUI` 类。
2. 添加图形：点击按钮（圆形/矩形/三角形/椭圆）。
3. 撤销/重做：顶部工具栏按钮。
4. 保存/加载：通过文件菜单操作。

渲染切换：
修改 `GlobalConfig` 的 `renderMode` 属性（支持 `swing`/`svg`）。

我们目前正在开发一个图形渲染系统，并且已经完成了大部分代码。现在需要生成软件设计报告，包括需求分析、总体设计和详细设计（类图、顺序图、活动图）。根据提供的代码，我们已经实现了以下设计模式：

1. 工厂模式（Factory）：在`ShapeFactory`和`BasicShapeFactory`中用于创建图形对象。

2. 抽象工厂（Abstract Factory）：虽然未显式实现抽象工厂，但工厂接口可以扩展。

3. 单例模式（Singleton）：`PersistenceManager`和`GlobalConfig`中使用了单例。

4. 适配器模式（Adapter）：`LegacyRendererAdapter`将旧版渲染器适配到新的渲染器接口。

5. 桥接模式（Bridge）：`Renderer`接口和其实现（如`SwingRenderer`、`SVGRenderer`）分离了抽象和实现。

6. 代理模式（Proxy）：`RemoteRendererProxy`为远程渲染提供了代理。

7. 命令模式（Command）：`Command`接口和具体命令（如`AddShapeCommand`）以及`UndoManager`用于实现撤销/重做。

8. 访问者模式（Visitor）：`ExportVisitor`和`JSONExportVisitor`用于导出图形数据。

9. 观察者模式（Observer）：`ShapeObserver`和`ShapeObservable`（虽然实现简单，但体现了观察者模式的思想）。

接下来，我们将编写软件设计报告的主要内容。

软件设计报告

1. 需求分析（具体功能）

- **图形渲染**：支持绘制圆形、矩形、椭圆、三角形。

- **多渲染引擎**：支持 Swing（本地图形界面）、SVG（生成 SVG 标签）、Legacy（旧版渲染器适配）和远程代理渲染（通过代理模式模拟）。

- **图形操作**：可以添加图形，并支持移动图形（通过命令模式实现撤销和重做）。

- **序列化与持久化**：可以将当前图形列表保存到文件，并从文件加载（使用 Gson 和 RuntimeTypeAdapterFactory 处理多态）。

- **导出功能**：通过访问者模式将图形导出为 JSON 格式。

- **观察者模式**：当图形发生变化时，通知观察者（目前代码中未完全实现，但有接口和基本结构）。

- **全局配置**：使用单例模式管理全局配置，如渲染模式。

2. 总体设计（体系架构图）

采用分层架构:

- **表示层（UI）**：`SwingUI`类，提供用户界面，包括按钮和绘图面板。

- **业务逻辑层**:

- 命令管理：`UndoManager`管理命令的执行、撤销和重做。

- 图形管理：维护图形列表，执行添加、移动等操作。

- 工厂：通过工厂创建图形对象。

- 持久化：`PersistenceManager`负责图形的序列化和反序列化。

- **渲染层**：通过桥接模式，将图形渲染抽象（`Renderer`接口）与具体实现（Swing、SVG、Legacy 适配器、远程代理）分离。

- **工具层**：包括访问者（导出 JSON）、观察者（图形变化通知）等。

体系架构图（文本描述）：

```

+------------------+ +--------------------+ +------------------+

| 表示层 | | 业务逻辑层 | | 渲染层 |

| SwingUI |---->| UndoManager |---->| Renderer |
```

| DrawingPanel | | Command | | SwingRenderer |

+------------------+ | ShapeFactory | | SVGRenderer |

| PersistenceManager| | LegacyRendererAdapter|

+--------------------+ | RemoteRendererProxy|

+------------------+

```

#### 3. 详细设计

##### 类图（主要类及其关系）

由于无法直接绘制 UML 图，这里用文字描述关键类和关系：

- **`Shape`接口**：所有图形的统一接口，定义了`render`、`move`和`accept`方法。
- **具体图形类**：`Circle`、`Rectangle`、`Ellipse`、`Triangle`，实现`Shape`接口。
- **`Renderer`接口**：定义了绘制各种图形的方法。
- **具体渲染器**：`SwingRenderer`（使用 Swing 绘制）、`SVGRenderer`（输出 SVG 标签）、`LegacyRendererAdapter`（适配旧版渲染器）、`RemoteRendererProxy`（远程渲染代理）。
- **`Command`接口**：定义`execute`和`undo`方法。
- **具体命令**：`AddShapeCommand`（添加图形）、`MoveShapeCommand`（移动图形）。
- **`UndoManager`**：管理命令栈，实现撤销和重做。
- **`ShapeFactory`接口和`BasicShapeFactory`**：创建图形对象。
- **`ExportVisitor`接口和`JSONExportVisitor`**：访问图形并导出为 JSON。
- **`ShapeObserver`和`ShapeObservable`**：观察者模式相关（但当前实现较为简单，仅预留接口）。
- **`PersistenceManager`**：单例，负责图形的保存和加载。
- **`GlobalConfig`**：单例，全局配置。

##### 顺序图（以添加图形为例）

1. 用户点击添加图形按钮（如圆形）。
2. `SwingUI`创建`Circle`对象，并创建`AddShapeCommand`命令（传入图形列表和图形对象）。
3. 调用`UndoManager`的`executeCommand`方法。
4. `UndoManager`执行命令，命令将图形添加到图形列表，并将命令压入撤销栈。
5. `SwingUI`重绘画布，在画布上调用所有图形的`render`方法。
6. 每个图形调用当前设置的`Renderer`（例如`SwingRenderer`）的绘制方法。

##### 活动图（撤销操作）
```

1. 用户点击撤销按钮。

2. `UndoManager`检查撤销栈是否为空，如果非空则弹出顶部命令。

3. 执行该命令的`undo`方法（例如从图形列表中移除图形）。

4. 将命令压入重做栈。

5. 刷新界面。

4. 设计模式应用总结

| 设计模式 | 应用点 |

|---|---|

| 工厂模式 | `BasicShapeFactory`创建具体图形 |

| 单例模式 | `PersistenceManager`和`GlobalConfig` |

| 适配器模式 | `LegacyRendererAdapter`适配旧版渲染器 |

| 桥接模式 | `Renderer`接口与具体渲染实现分离 |

| 代理模式 | `RemoteRendererProxy`代理远程渲染 |

| 命令模式 | `Command`接口及具体命令，`UndoManager`管理命令栈 |

| 访问者模式 | `ExportVisitor`接口及`JSONExportVisitor`实现图形导出 |

| 观察者模式 | `ShapeObserver`和`ShapeObservable`（预留，未完全实现） |

5. 其他

- **测试**：使用 JUnit 对命令模式、持久化等核心功能进行单元测试。

- **文档**：使用 JavaDoc 生成文档。

- **部署**：打包为可执行 JAR，用户可通过`MainUI`启动。

由于时间限制，我们无法在此处绘制 UML 图，但可以根据上述描述使用 PowerDesigner 或 PlantUML 等工具绘制。

注意：观察者模式在代码中有一个接口`ShapeObserver`和一个简单实现`ShapeObservable`，但未在主要逻辑中使用，属于预留扩展。

以上即为软件设计报告的主要内容，可以根据需要进一步细化和补充。

liying@huawei-B730E:~/IdeaProjects/GraphRenderRestructured$ tree

.

├── GraphRenderRestructured.iml

├── pom.xml

├── runtest.bash

```
├── src
│   ├── main
│   │   └── java
│   │       └── com
│   │           └── example
│   │               └── renderer
│   │                   ├── adapter
│   │                   │   ├── LegacyRendererAdapter.java
│   │                   │   └── LegacyRenderer.java
│   │                   ├── bridge
│   │                   │   ├── Renderer.java
│   │                   │   └── SwingRenderer.java
│   │                   ├── command
│   │                   │   ├── AddShapeCommand.java
│   │                   │   ├── Command.java
│   │                   │   └── UndoManager.java
│   │                   ├── config
│   │                   │   └── GlobalConfig.java
│   │                   ├── factory
│   │                   │   ├── Circle.java
│   │                   │   ├── Ellipse.java
│   │                   │   ├── Rectangle.java
│   │                   │   ├── ShapeFactory.java
│   │                   │   ├── Shape.java
│   │                   │   └── Triangle.java
│   │                   ├── observer
│   │                   │   ├── ShapeObservable.java
│   │                   │   └── ShapeObserver.java
│   │                   ├── proxy
│   │                   │   └── RemoteRendererProxy.java
```

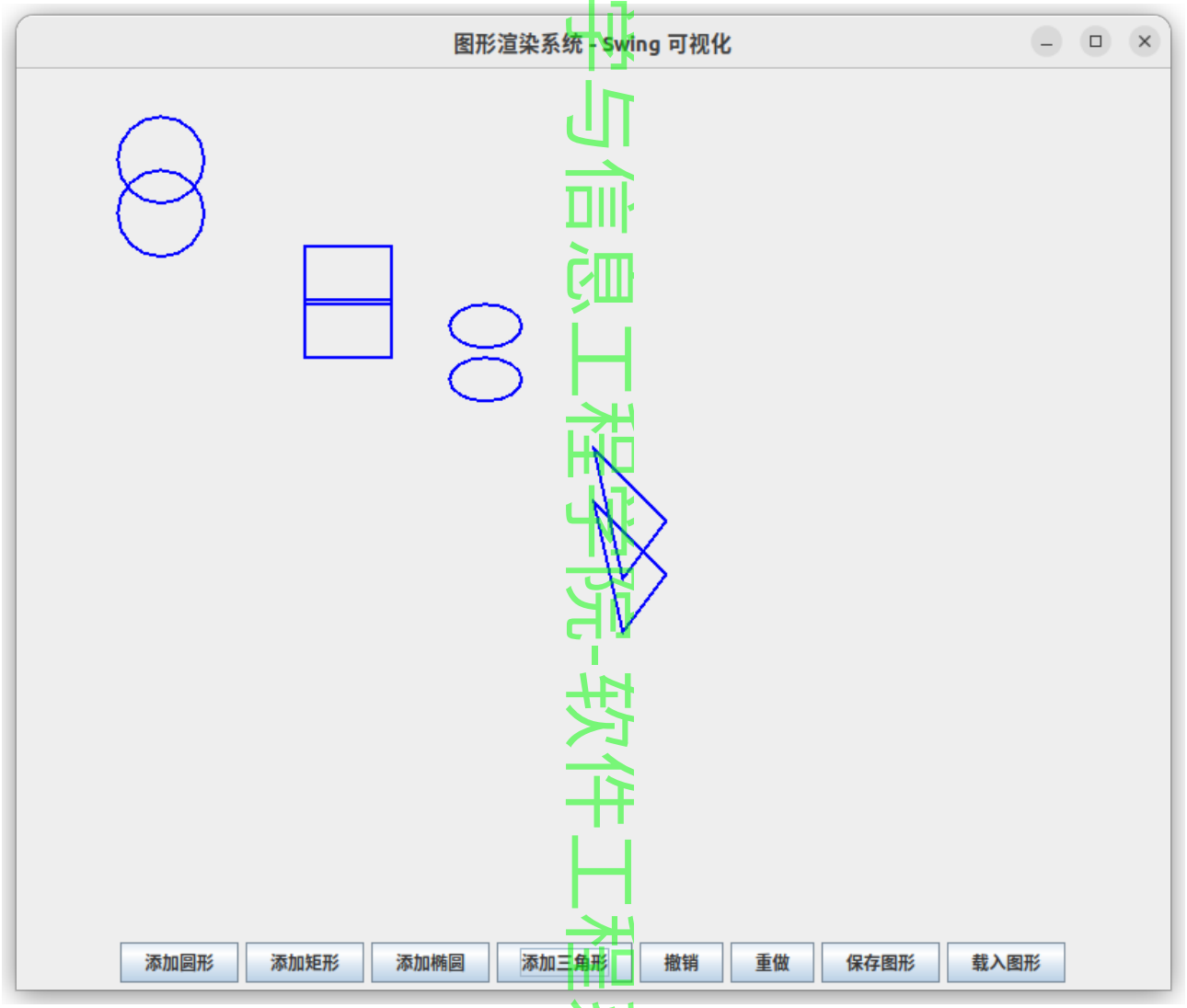
```
│   │   ├── singleton
│   │   │   └── PersistenceManager.java
│   │   ├── ui
│   │   │   ├── DrawingPanel.java
│   │   │   ├── MainUI.java
│   │   │   └── SwingUI.java
│   │   ├── util
│   │   │   └── RuntimeTypeAdapterFactory.java
│   │   └── visitor
│   │       ├── ExportVisitor.java
│   │       └── JSONExportVisitor.java
│   └── test
│       └── java
│           └── com
│               └── example
│                   └── renderertest
│                       ├── adapter
│                       │   ├── LegacyRendererAdapterTest.java
│                       │   └── LegacyRendererTest.java
│                       └── bridge
│                           └── SwingRendererTest.java
├── target
│   ├── classes
│   │   └── com
│   │       └── example
│   │           └── renderer
│   │               ├── adapter
│   │               │   └── LegacyRendererAdapter.class
│   │               ├── bridge
│   │               │   ├── Renderer.class
```

```
│   │   │   ├── ShapeObserver.class
│   │   │   ├── SVGRenderer.class
│   │   │   └── SwingRenderer.class
│   │   ├── command
│   │   │   ├── AddShapeCommand.class
│   │   │   ├── Command.class
│   │   │   ├── MoveShapeCommand.class
│   │   │   └── UndoManager.class
│   │   ├── config
│   │   │   └── GlobalConfig.class
│   │   ├── factory
│   │   │   ├── BasicShapeFactory.class
│   │   │   ├── Circle.class
│   │   │   ├── Ellipse.class
│   │   │   ├── Rectangle.class
│   │   │   ├── Shape.class
│   │   │   ├── ShapeFactory.class
│   │   │   └── Triangle.class
│   │   ├── legacy
│   │   │   └── LegacyRenderer.class
│   │   ├── observer
│   │   │   ├── ShapeObservable.class
│   │   │   └── ShapeObserver.class
│   │   ├── proxy
│   │   │   └── RemoteRendererProxy.class
│   │   ├── singleton
│   │   │   └── PersistenceManager.class
│   │   ├── ui
│   │   │   ├── DrawingPanel.class
│   │   │   ├── MainUI.class
```

```
│   │   │   └── SwingUI.class
│   │   ├── util
│   │   │   ├── RuntimeTypeAdapterFactory$1.class
│   │   │   └── RuntimeTypeAdapterFactory.class
│   │   └── visitor
│   │       ├── ExportVisitor.class
│   │       └── JSONExportVisitor.class
│   ├── generated-sources
│   │   └── annotations
│   ├── generated-test-sources
│   │   └── test-annotations
│   ├── maven-status
│   │   └── maven-compiler-plugin
│   │       ├── compile
│   │       │   └── default-compile
│   │       │       ├── createdFiles.lst
│   │       │       └── inputFiles.lst
│   │       └── testCompile
│   │           └── default-testCompile
│   │               ├── createdFiles.lst
│   │               └── inputFiles.lst
│   ├── surefire-reports
│   │   ├── bridge.SwingRendererTest.txt
│   │   ├── com.example.renderertest.adapter.LegacyRendererAdapterTest.txt
│   │   ├── com.example.renderertest.adapter.LegacyRendererTest.txt
│   │   ├── TEST-bridge.SwingRendererTest.xml
│   │   ├── TEST-com.example.renderertest.adapter.LegacyRendererAdapterTest.xml
│   │   └── TEST-com.example.renderertest.adapter.LegacyRendererTest.xml
│   └── test-classes
│       ├── bridge
```

```
|       |   └── SwingRendererTest.class
|       └── com
|           └── example
|               └── renderertest
|                   └── adapter
|                       ├── LegacyRendererAdapterTest.class
|                       └── LegacyRendererTest.class
└── test1.json

58 directories, 73 files
```



图形可以保存为 JSON 文件：test1.json

0
x 100
y 100
radius 30
type "Circle"
1
x 200
y 160
width 60
height 40
type "Rectangle"
2
x 300
y 200
width 50
height 30
type "Ellipse"

可以载入图形。

实现了 Undu+Redo

X

李莹
教师