实验3 继承与多态

1 实验目的

（1）熟悉面向对象编程中的继承特性；
（2）掌握基于函数重写的多态实现和应用。

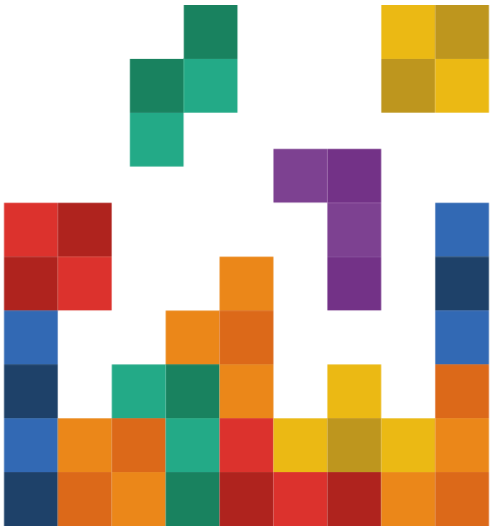
2 实验环境

开发环境：JDK 8.0（或更高版本）+JavaFX
开发工具：Eclipse
设计工具：StarUML（或PlantUML等其他工具）

3 实验内容

3.1 方块游戏

问题描述：试玩俄罗斯方块游戏，尝试用面向对象的思维分析和实现一个游戏原型。

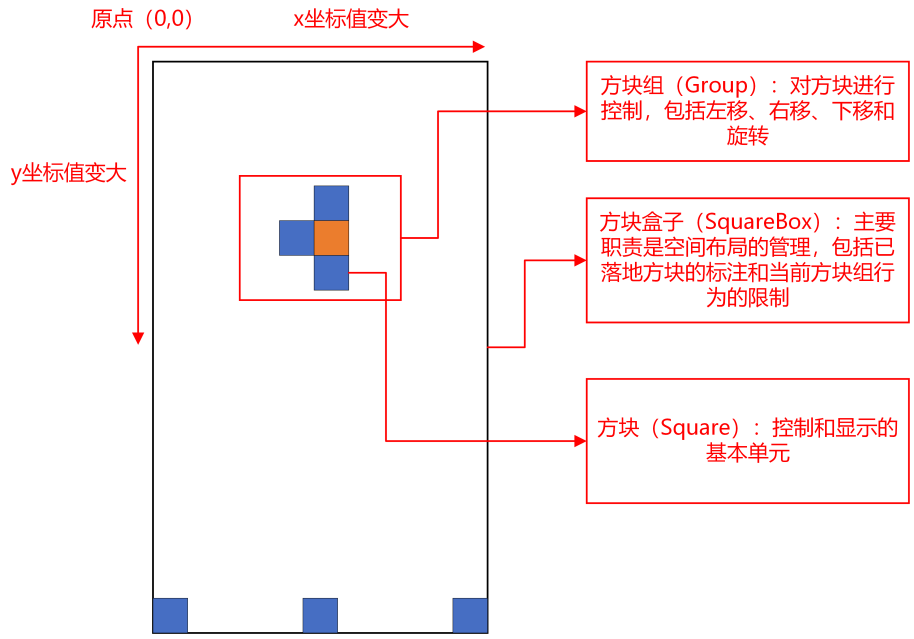


1 识别对象

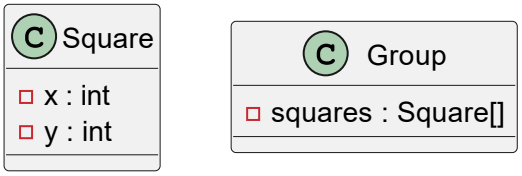
分析游戏场景中可能存在的对象和基本职责。如下图所示，游戏包含三种基本对象：

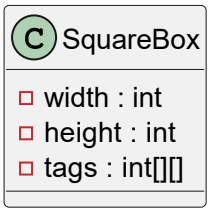
- 方块（Square）：游戏中被控制和显示的基本单元，可以移动；
- 方块组（Group）：对方块的移动方式进行控制，包括特定阵型的组织、各方向的移动、转动（顺时针）和方块的放置；

- 方块盒子（SquareBox）：负责空间布局管理，包括已落地方块的标记和方块组行为的限制。



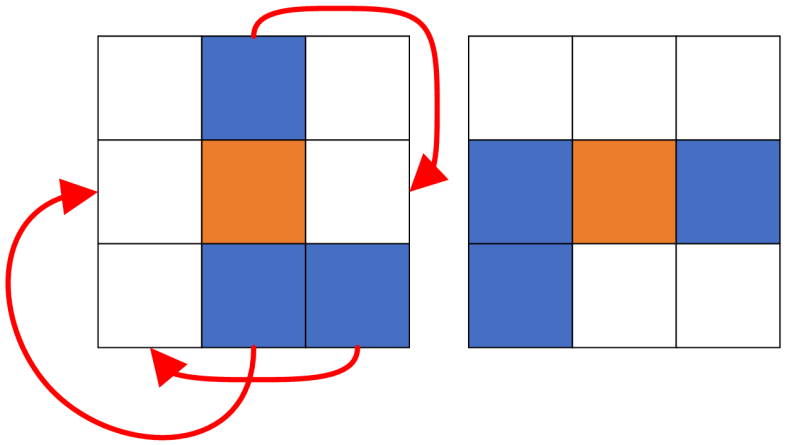
分析得到基本类图：



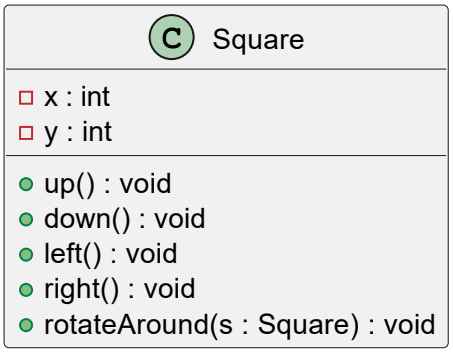


2 方块特征分析

对方块的特征进一步分析。方块能上下左右移动，单次移动距离为1个单位。如下图所示，一个方块组的旋转可以看成所有边缘方块围绕中心方块顺时针旋转，因此这里可以给方块增加一个绕中心点旋转职责。

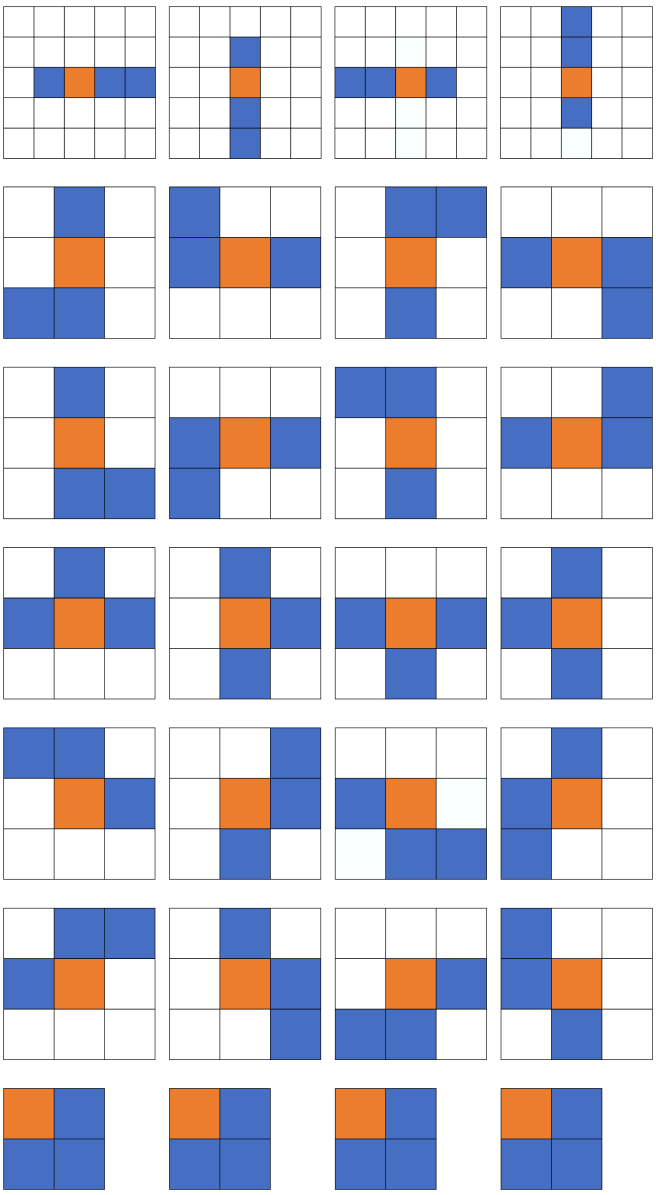


分析后类图细化为：

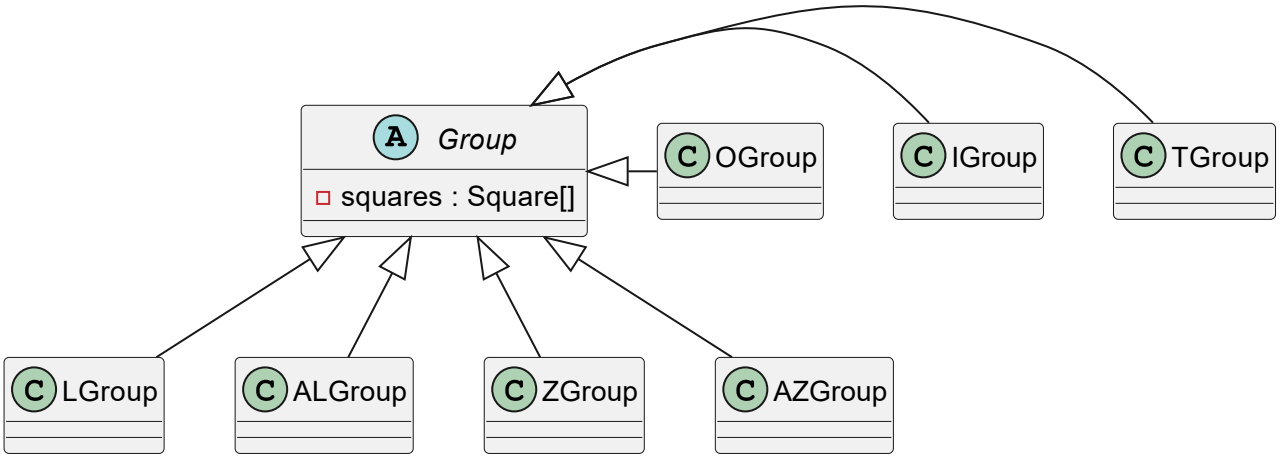


3 方块组特征分析

（3）对方块组的特征进一步分析。如下图所示，方块组有七种类型，包括正反L型、正反Z型、I型、T型和O型，可以为每个方块组设置一个中心点用于旋转逻辑的实现。

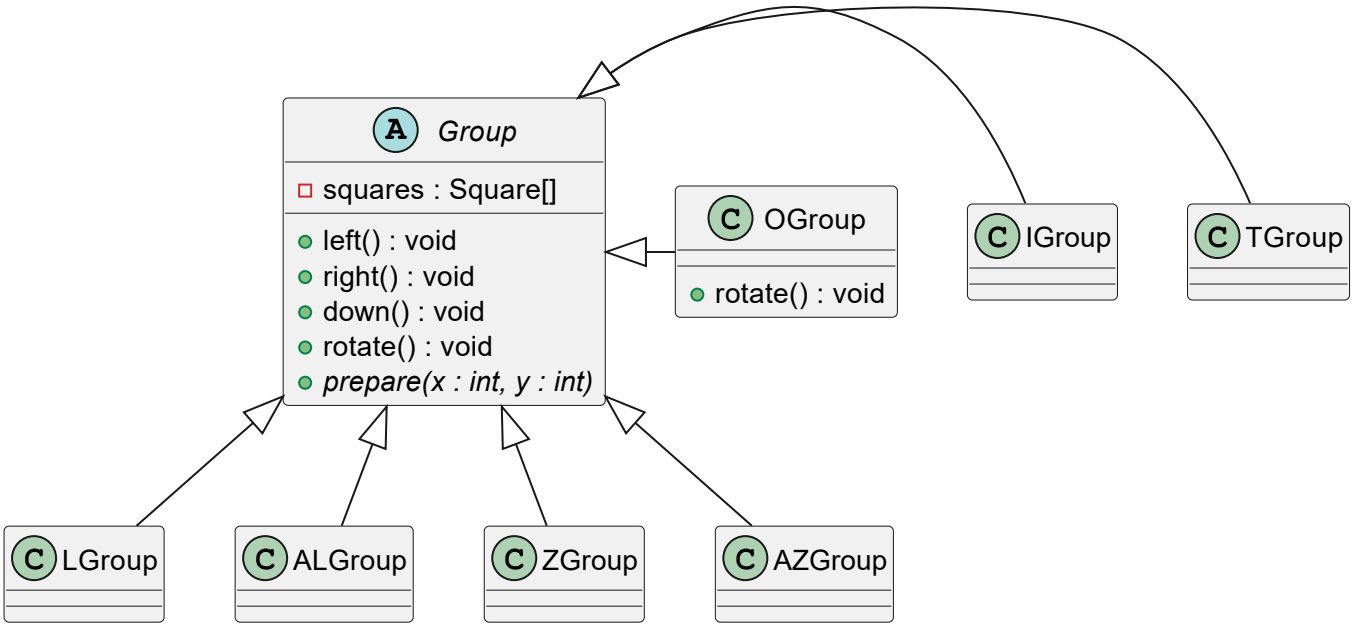


每个方块组的形状和行为都存在一定差异，只有确定了形状才能创建方块组。因此，这里定义几种对应的子类：



方块组的行为特征分析：

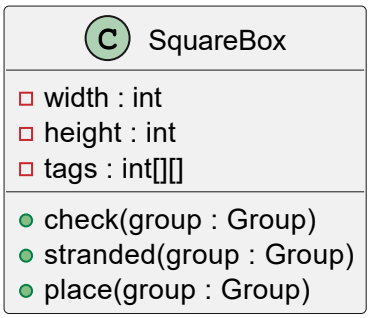
- 每个方块组都需要指定一个中心，这里指定第一个方块为中心方块squares[0]，当然也可以通过其他方式表示，例如声明一个中心索引字段；
- 各类方块组的平移（左、右、下）逻辑基本一致，属于共同特征，可以统一定义在`Group`类中；
- 由于方块定义了绕目标方块转动的行为，各类方块组的转动逻辑也可以统一定义在`Group`类中；但转动逻辑存在一个例外，就是`OGroup`是不转动的，因此要进行函数重写；
- 顶层方块组无法直接创建，需要围绕某个中心点进行队形准备（prepare），而各类方块组的队形准备实现逻辑又不同，因此可以设计为抽象函数。



4 方块盒子特征分析

对方块盒子的特征进一步分析。方块盒子的行为主要包括空间检测、约束和方块标记管理几个方面：

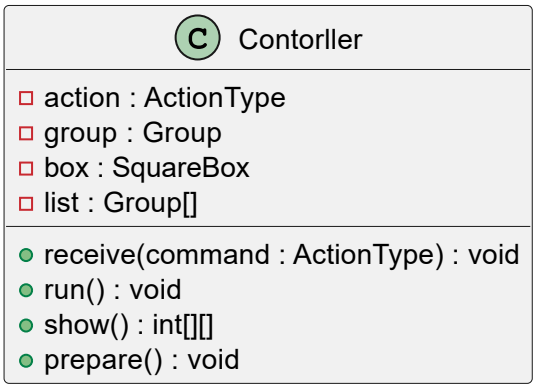
- 检测（check）方块组是否能进入盒子的某个位置；
- 判断当前方块组是否已经搁浅（stranded），搁浅后需要对方块进行标记（mark）或放置（place）；



5 控制类的引入与分析

这里还需要一个对象负责接收用户的操作请求，并组织游戏实体完成一次游戏业务后返回结果。这个对象类似于游戏控制中枢，因此命名为控制器`Controller`，实际上控制器也是一种典型的设计模式，感兴趣的同学可以查阅资料。具体职责如下：

- 接收（receive）图形用户接口输入的操作请求，这里的操作请求类型有多种，可以采用枚举方式设计。
- 根据一定算法或参数准备（prepare）当前方块组，这里选择了简单的随机算法；
- 实现单次运行（run）逻辑；
- 向图形用户接口反馈要显示（show）的信息。

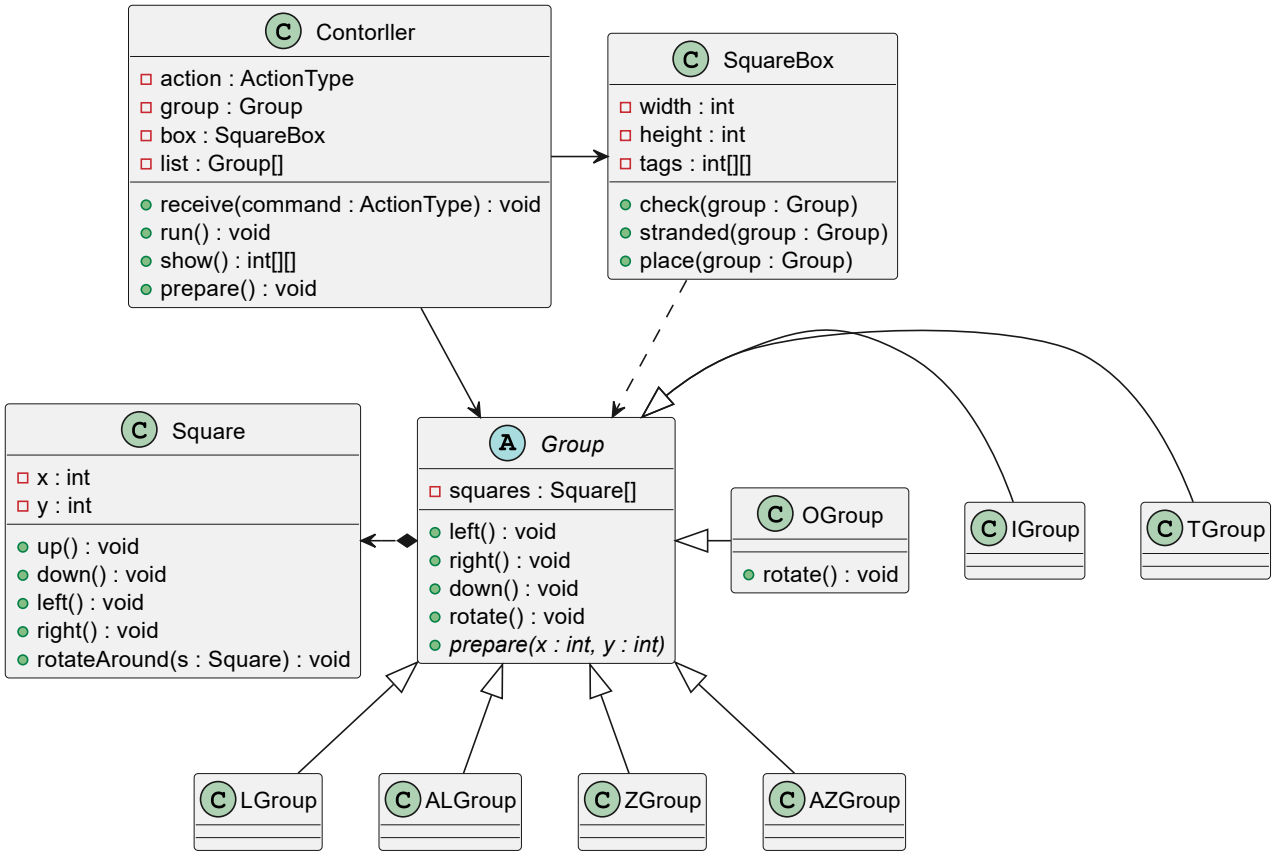


6 界面设计

用户界面包括主界面（MainApp）、盒子视图（SquareBoxView）和游戏循环（GameLoop）三个类。这里不做要求，感兴趣的同学自己学习。

7 系统结构与代码骨架

系统结构如下，可以看出当前结构与GUI无关，GUI使用`Controller`实现游戏逻辑，当更换一个GUI类库或风格不会影响底层的游戏代码。



方块类：

```
public class Square {
  private int x, y;

  public Square() {}
  public Square(int x, int y) {
    this.setX(x);
    this.setY(y);
  }

  // 坐标系：左下方向递增
  public void up(int step) {}
  public void down(int step) {}
  public void left(int step) {}
  public void right(int step) {}

  // 围绕目标方块顺时针转动90度
  public void rotateAround(Square s){}

  // x,y的Getter与Setter函数省略
}
```

方块组合类：

```
public abstract class Group implements Cloneable{
  protected Square[] squares; // 方块组，中心方块存放在第一个位置

  // 创建一个默认组合，没有实际意义，需要子类给出形状
  protected Group() {
    squares = new Square[]{new Square(),new Square(),new Square(),new Square()};
    prepare(squares[0].getX(), squares[0].getY()); // 整理队形
  }

  /*
   *
   * 组合的运动系统，包括左移、右移、下移和转动；运动后向盒子对象请求放置，失败则撤销动作
   *
   * */
  protected void left() {}
  protected void right() {}
  protected void down() {}
  protected void rotate() {}

  /*
   *
   * 组合的边界信息，包括左、右和下三个边缘坐标
   *
   * */
  public int minX() {}
  public int maxX() {}
  public int maxY() {}

  // 以（x,y）为中心按特定形状准备方块组
  public abstract void prepare(int x, int y);

  // 部分Getter与Setter函数省略
}
```

```
public class LGroup extends Group{
  // 以(x,y)为中心整理队形
  public void prepare(int x, int y) {}
}
```

```
public class ALGroup extends Group{
  // 以(x,y)为中心装填方块
  public void prepare(int x, int y) {}
}
```

```
public class ZGroup extends Group{
  // 以(x,y)为中心装填方块
  public void prepare(int x, int y) {}
}
```

```
public class AZGroup extends Group{
  // 以(x,y)为中心装填方块
  public void prepare(int x, int y) {}
}
```

```
public class TGroup extends Group{
  // 以(x,y)为中心装填方块
  public void prepare(int x, int y) {}
}
```

```
public class IGroup extends Group{
  // 以(x,y)为中心装填方块
  public void prepare(int x, int y) {}
}
```

```
public class OGroup extends Group{
  // 以(x,y)为中心装填方块
  public void prepare(int x, int y) {}
}
```

盒子类:

```
public class SquareBox{
  private int width, height;
  private int[][] tags; // 方块落地标记，1表示有方块，0表示没有

  public SquareBox(int width, int height) {
    this.setWidth(width);
    this.setHeight(height);
    tags = new int[height][width];
  }

  // 放置
  public void place(Group group) {}

  // 判断是否搁浅
  public boolean stranded(Group g) {}

  // 判断是否能放置
  public boolean check(Group g) {}

  // 复制一份tags用于显示
  public int[][] copyTags(){}

  // 省略Getter和Setter函数

}
```

控制器类：

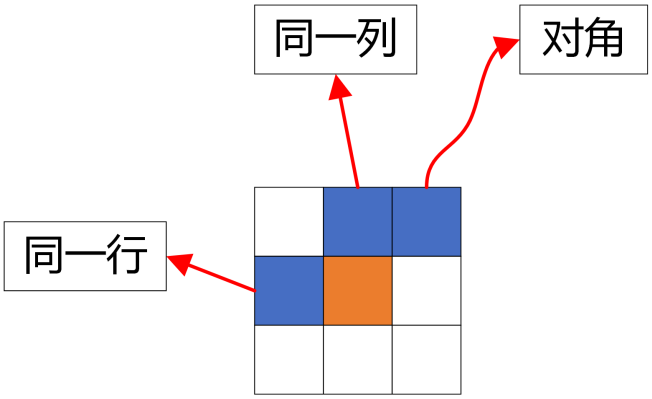
```
public class Controller {
  private ActionType action;
  Group[] list = {new LGroup(), new ALGroup(),
      new ZGroup(), new AZGroup(),
      new TGroup(), new OGroup(), new IGroup()};
  private Group group;
  private SquareBox box;

  public Controller(SquareBox box){
    this.box = box;
    this.action = ActionType.DOWN;
    prepare();
  }

  // 动作类型的枚举
  public static enum ActionType{
    LEFT,RIGHT,DOWN,ROTATE
  }

  // 接受一个动作命令
  public void receive(ActionType command){}

  // 游戏单次运行逻辑
  public void run() {}

  // 随机准备一个方块组
  public void prepare() {}

  // 返回显示信息
  public int[][] show(){}

}
```

8 具体业务逻辑的实现

（1）方块的平移。平移实现较简单，就是X或Y坐标的增减。

```
public void up(int step) {
  setY(getY() - step);
}
public void down(int step) {
  setY(getY() + step);
}
public void left(int step) {
  setX(getX() - step);
}
public void right(int step) {
  setX(getX() + step);
}
```

（2）方块绕中心顺时针旋转。这里主要分三种情况：与中心在同一行、与中心在同一列，与中心在对角上。



```
// 围绕目标方块顺时针转动90度
public void rotateAround(Square s){
  int xDist = this.x - s.getX();
  int yDist = this.y - s.getY();
  if(yDist == 0) { // 和中心在同一行
    setX(s.getX()); // 转动后同一列，x相同
    setY(getY() + xDist); // 在右边，xDist为正，下移，在左边，xDist为负，上移
  }else if(xDist == 0){ // 和中心在同一列
    setY(s.getY()); // 转动后同一列，y相同
    setX(getX() - yDist); // 在下边，yDist为正，左移，在上边，yDist为负，右移
  }else { // 取巧：游戏里只存在九宫格四个角的情况
    if(xDist * yDist > 0) { // 符号相同，顺时针转动是沿X轴方向移动
      setX(getX() - 2 * xDist); // 同为正，则左移，同为负，则右移
    }else { // 符号不同，顺时针转动是沿Y轴方向移动
      setY(getY() + 2 * xDist); // xDist为正，则下移，为负，则上移
    }
  }
}
```

（3）方块组合的移动或转动逻辑都基于方块的行为完成，这里组织多个方块一起完成一个动作。

```
protected void left() {
  for(int i = 0; i < squares.length; i++) {
    squares[i].left(1);
  }
}
protected void right() {
  for(int i = 0; i < squares.length; i++) {
    squares[i].right(1);
  }
}
protected void down() {
  for(int i = 0; i < squares.length; i++) {
    squares[i].down(1);
  }
}
protected void rotate() {
  for(int i = 1; i < squares.length; i++) {
    squares[i].rotateAround(squares[0]);
  }
}
```

（4）方块组的边界计算，用于移动过程中的检测。

```
// 左边界
public int minX() {
  int x = Integer.MAX_VALUE;
  for(Square s : getSquares()) {
    if(s.getX() < x)  x = s.getX();
  }
  return x;
}
// 右边界
public int maxX() {
  int x = 0;
  for(Square s : getSquares()) {
    if(s.getX() > x)  x = s.getX();
  }
  return x;
}
// 下边界
public int maxY() {
  int y = 0;
  for(Square s : getSquares()) {
    if(s.getY() > y)  y = s.getY();
  }
  return y;
}
```

（5）方块盒子检查方块位置是否合法，包括左右边界以及当前位置是否已经被标记为方块。这里没有检测下边界的原因是，下边界涉及到搁浅检测。

```
public boolean check(Group g) {
  if(g.minX() < 0 || g.maxX() >= width) {
    return false;
  }else{
    for(Square s : g.getSquares()) {
      if(tags[s.getY()][s.getX()] == 1) {
        return false;
      }
    }
  }
  return true;
}
```

（6）方块盒子判断方块组是否搁浅。搁浅的情况包括触碰盒子的底部，或则某个方块的正下方已经存在方块标记。

```
public boolean stranded(Group g) {
  boolean stranded = false;
  if(g.maxY() == height - 1) {
    stranded = true;
  }else {
    for(Square s : g.getSquares()) {
      if(tags[s.getY() + 1][s.getX()] == 1) {
        stranded = true;
      }
    }
  }
  return stranded;
}
```

（7）放置盒子包括两个部分，一是将方块组所在的位置标记为方块，二是检测是否有满格层，所有则消除所有满格层。

```
public void place(Group group) {
  // 放下方块，更新盒子的状态
  for(Square s : group.getSquares()) {
    tags[s.getY()][s.getX()] = 1;
  }
  // 消除满格层
  int[][] temp = new int[height][width];
  int r = height - 1;
  for(int y = height - 1; y >= 0; y--) {
    boolean full = true;
    for(int x = 0; x < width; x++) {
      if(tags[y][x] == 0) {
        full = false;
        break;
      }
    }
    if(!full)
      temp[r--] = Arrays.copyOf(tags[y], width);
  }
  tags = temp;
}
```

（8）控制器接收一个命令，即将修改当前动作类型与外部命令保持一致。

```
public void receive(ActionType command){
  action = command;
}
```

（9）在盒子的顶部随机准备一个方块组。

```
public void prepare() {
  group = list[(int) (Math.random() * list.length)];
  group.prepare(3, box.getWidth() / 2);
}
```

（10）根据当前指令类型执行一个动作。方块组动作是否执行成功需要先进行一次检测，具体思路为：先创建一个方块组复制体，复制体完成动作后作为检测样本进行检测，若通过则本体执行动作。执行完动作后将指令恢复为默认动作`DOWN`，并判断当前方块是否搁浅，若搁浅则执行放置操作。

```
public void run() {
  Group temp = group.clone();
  switch(action) {
    case LEFT: {
      temp.left();
      if(box.check(temp)) {
        group.left();
      }
    }break;
    case RIGHT:{
      temp.right();
      if(box.check(temp)) {
        group.right();
      }
    }break;
    case DOWN:{
      temp.down();
      if(box.check(temp)) {
        group.down();
      }
    }break;
    case ROTATE:{
      temp.rotate();
      if(box.check(temp)) {
        group.rotate();
      }
    }
    default:break;
  }
  action = ActionType.DOWN; // 恢复默认动作
  if(box.stranded(group)) {
    box.place(group);
    prepare(); // 准备新方块组
  }
}
```

`Group`类的复制实现如下：

```
public Group clone() {
  Group group = null;
  try {
    group = (Group) super.clone();
    Square[] squares = group.getSquares();
    Square[] temp = new Square[squares.length];
    for(int i = 0; i < squares.length; i++) {
      temp[i] = new Square(squares[i].getX(), squares[i].getY());
    }
    group.squares = temp;
  } catch (CloneNotSupportedException e) {
    e.printStackTrace();
  }
  return group;
}
```

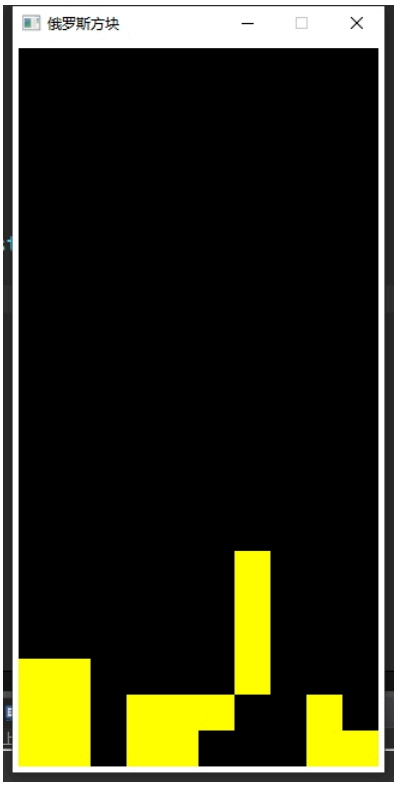
（11）反馈当前业务状态，供GUI显示用。信息包含方块组和盒子中的方块标记两部分。

```
public int[][] show(){
  int[][] resp = box.copyTags();
  Square[] a = group.getSquares();
  for(Square s : a) {
    resp[s.getY()][s.getX()] = 1;
  }
  return resp;
}
```

（12）GUI源代码直接给出。

9 运行与调试

调试、运行程序



10 思考与训练

- 仔细读代码，理解和掌握设计思想和技巧；
- `Contorller`类的作用是什么，添加该类能带来什么效果？
- 程序中体现了几处多态性，有什么作用？
- 根据自己的理解，尝试完善程序，例如不同颜色、游戏结束机制等。

3.2 环境配置

由于3.1实验采用JavaFX开发了一个简单的游戏界面，这里对JavaFX的使用环境进行简单说明。

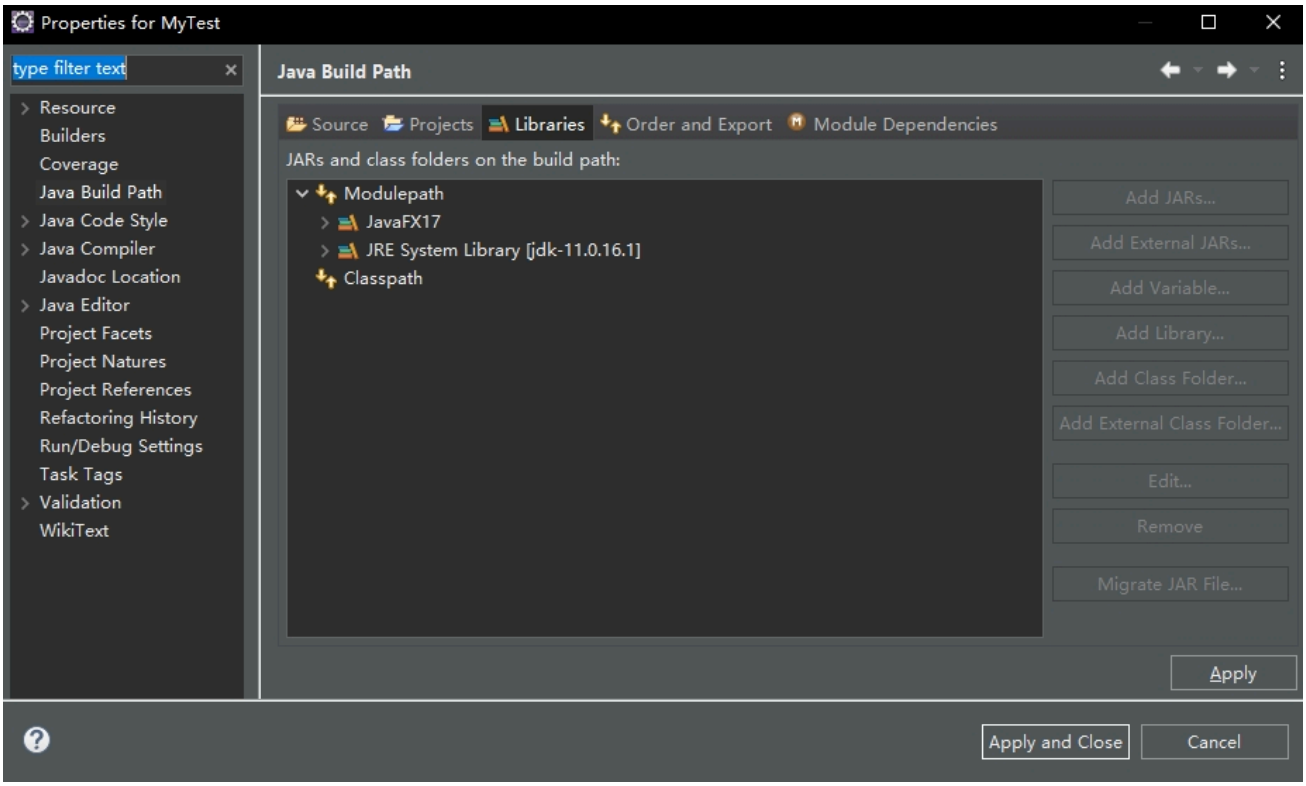
1 Java SE8

Java SE8中绑定了JavaFX库，不用独立引入和配置。

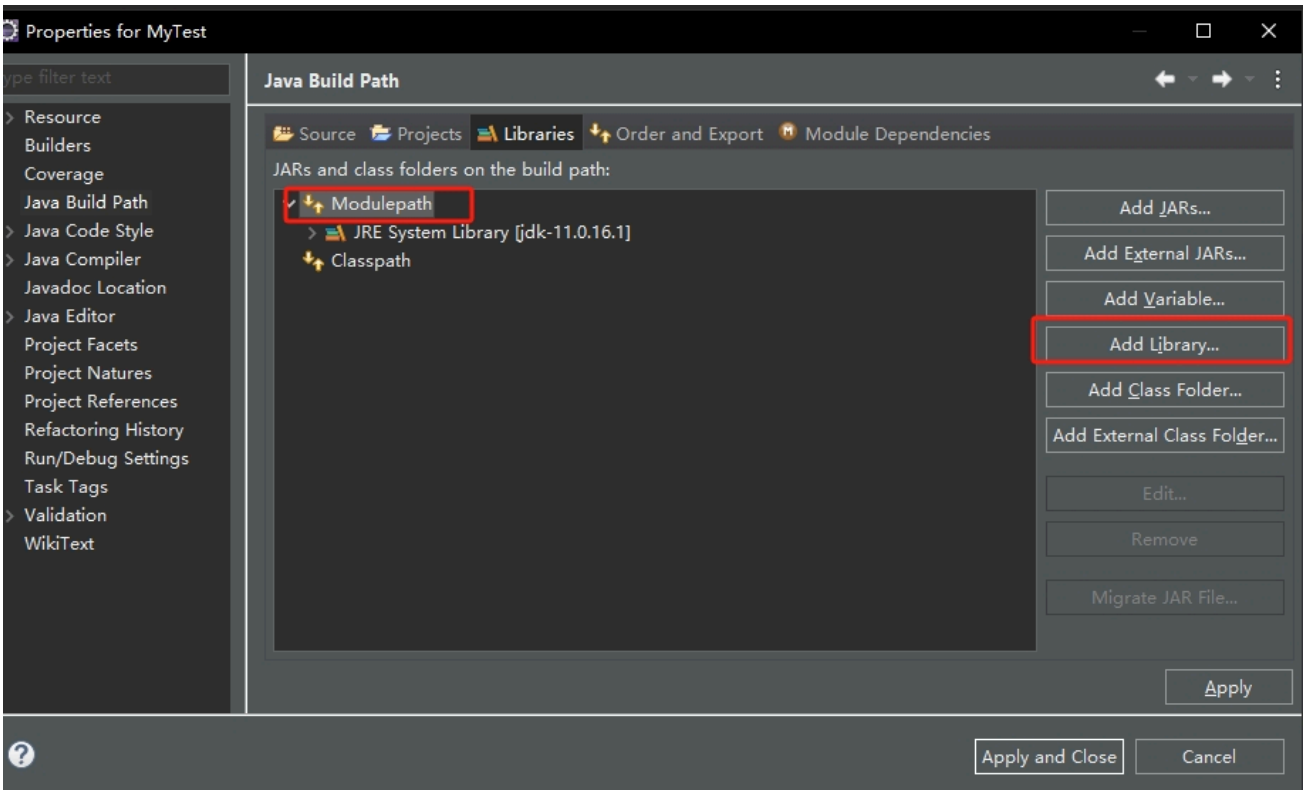
2 Java SE 11及以上版本

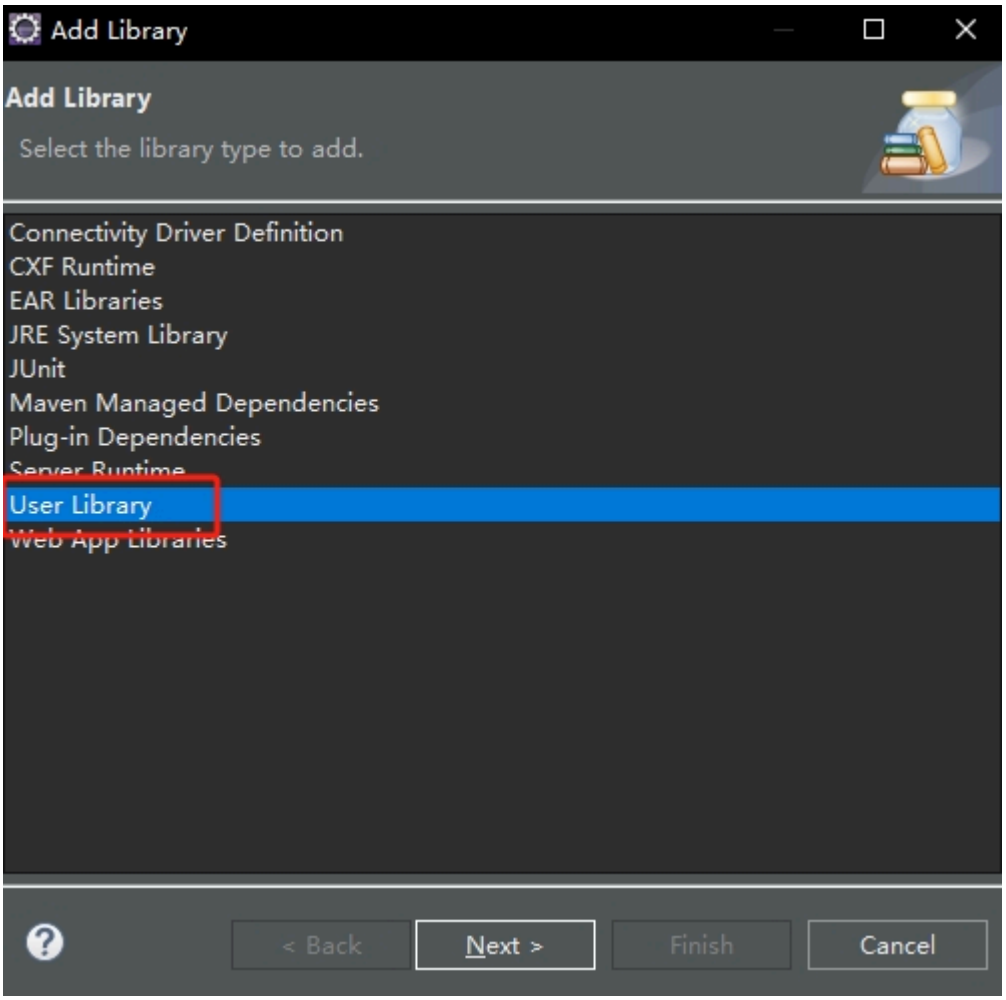
需要下载对应的JavaFX包，引入到当前的项目。下载地址：https://gluonhq.com/products/javafx/

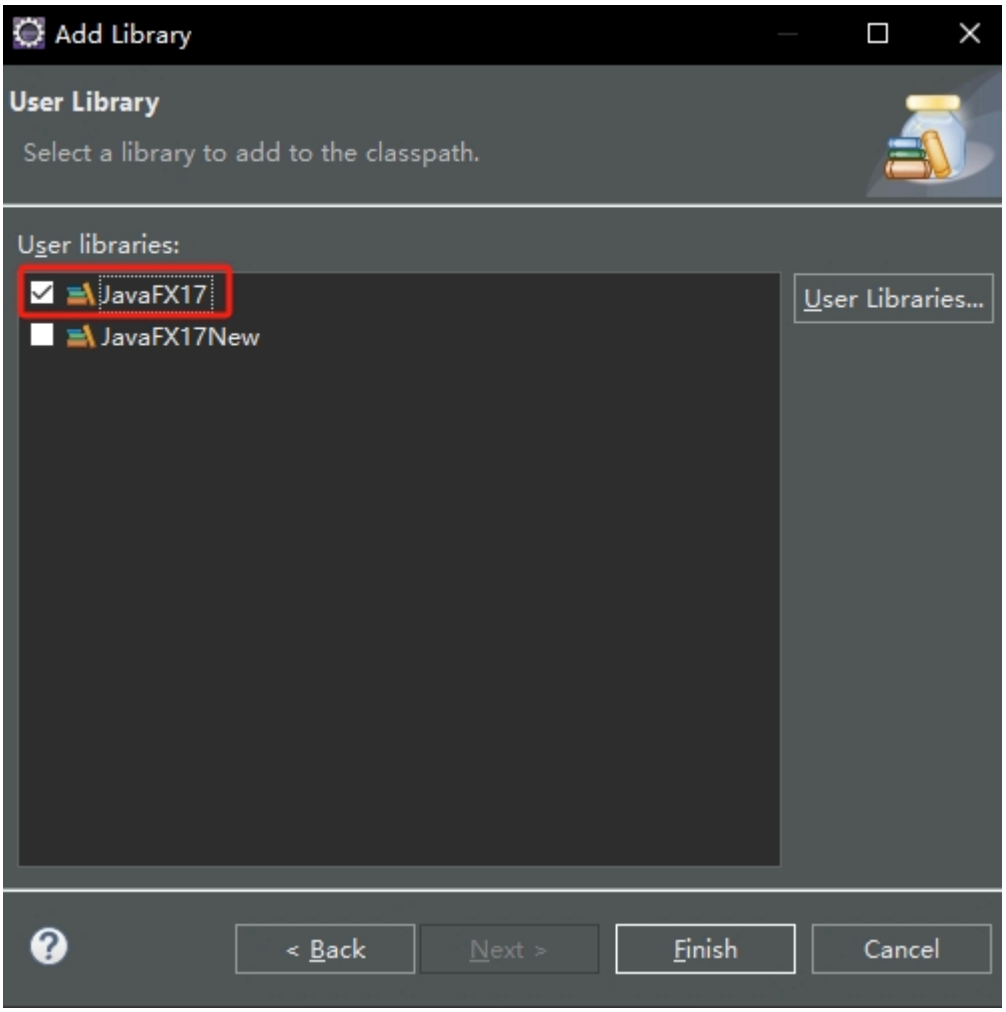
引入步骤：
1、右键项目，通过`构建路径配置`菜单进入构建配置界面

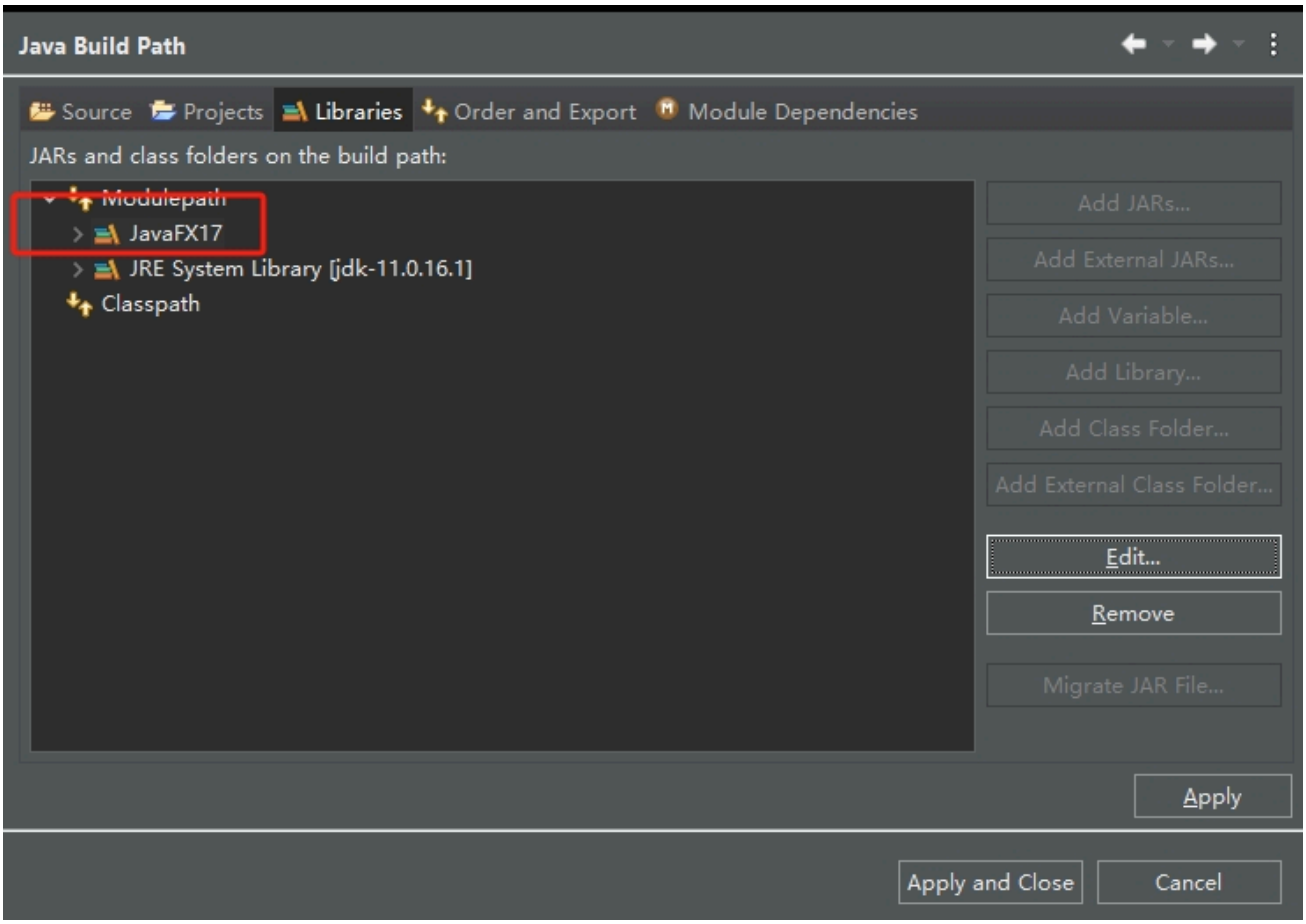


2、引入JavaFX用户库

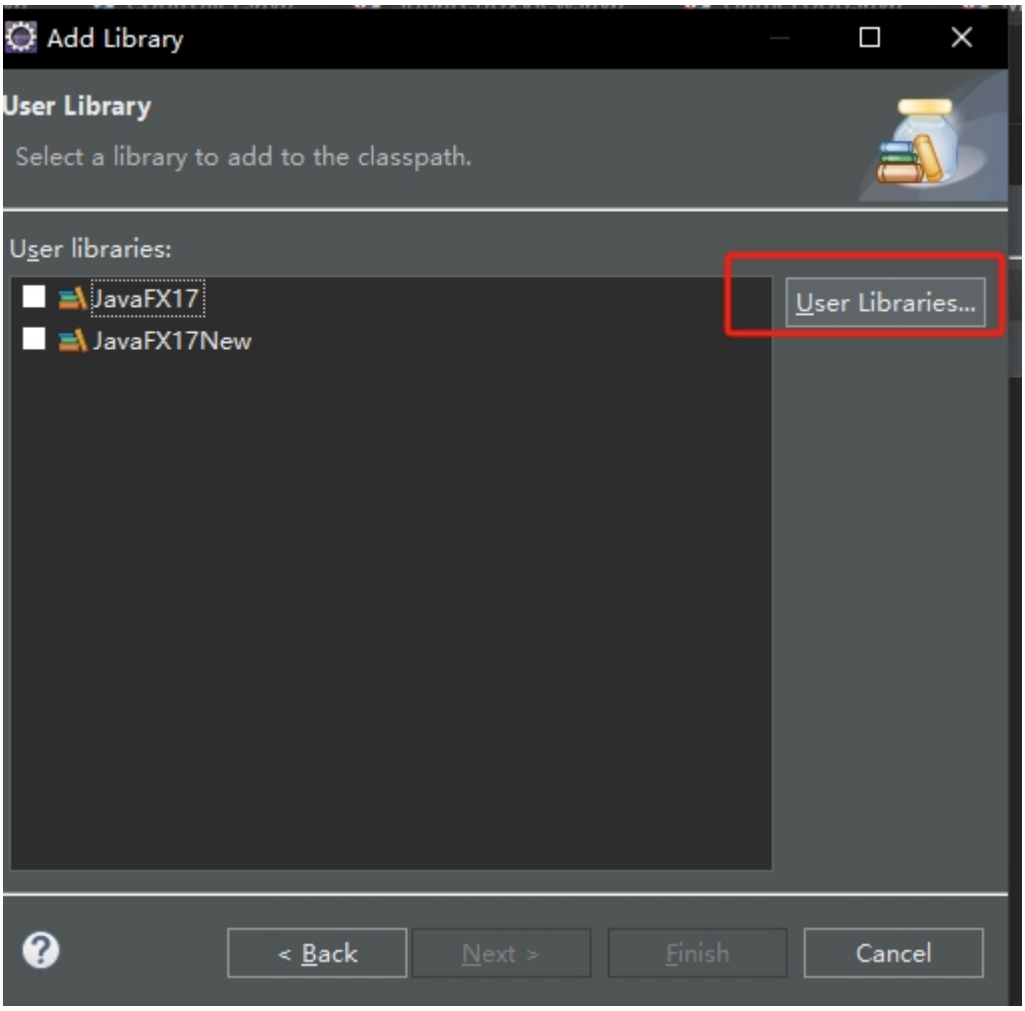


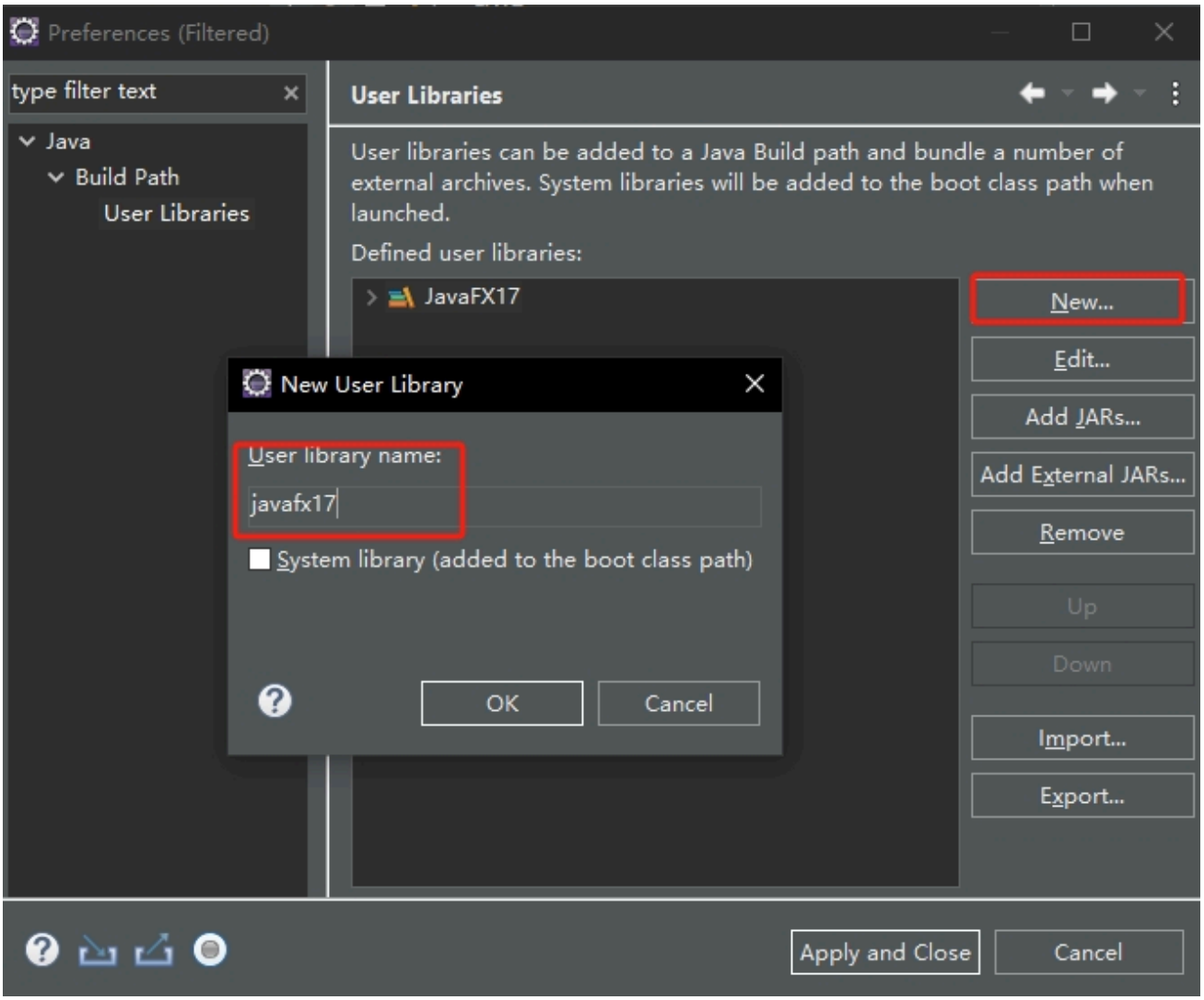

Add Library
Add Library
Select the library type to add.
Connectivity Driver Definition
CXF Runtime
EAR Libraries
JRE System Library
JUnit
Maven Managed Dependencies
Plug-in Dependencies
Server Runtime
User Library
Web App Libraries
< Back
Next >
Finish
Cancel

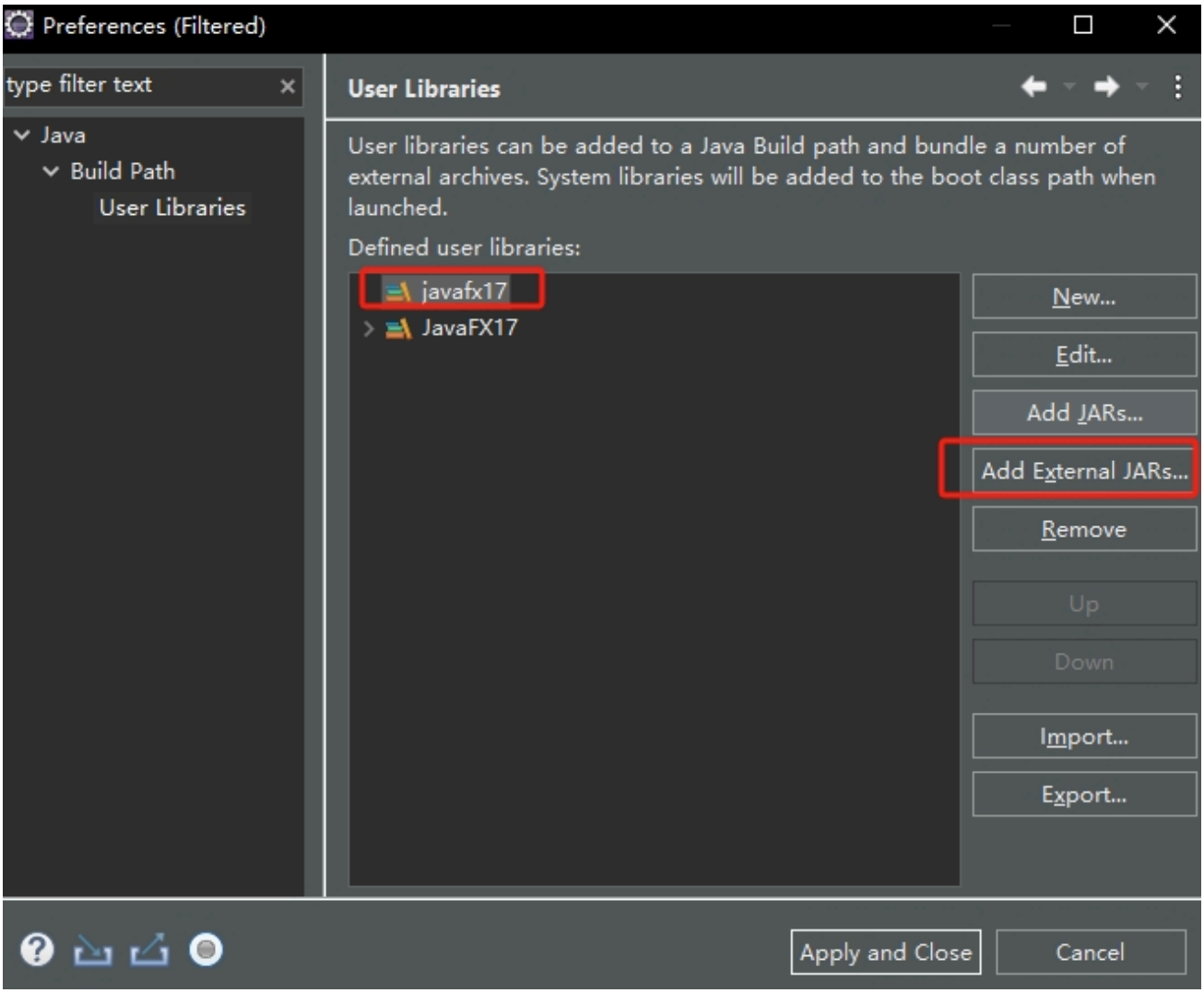

Add Library
User Library
Select a library to add to the classpath.
User libraries:
JavaFX17
JavaFX17New
User Libraries...
< Back
Next >
Finish
Cancel

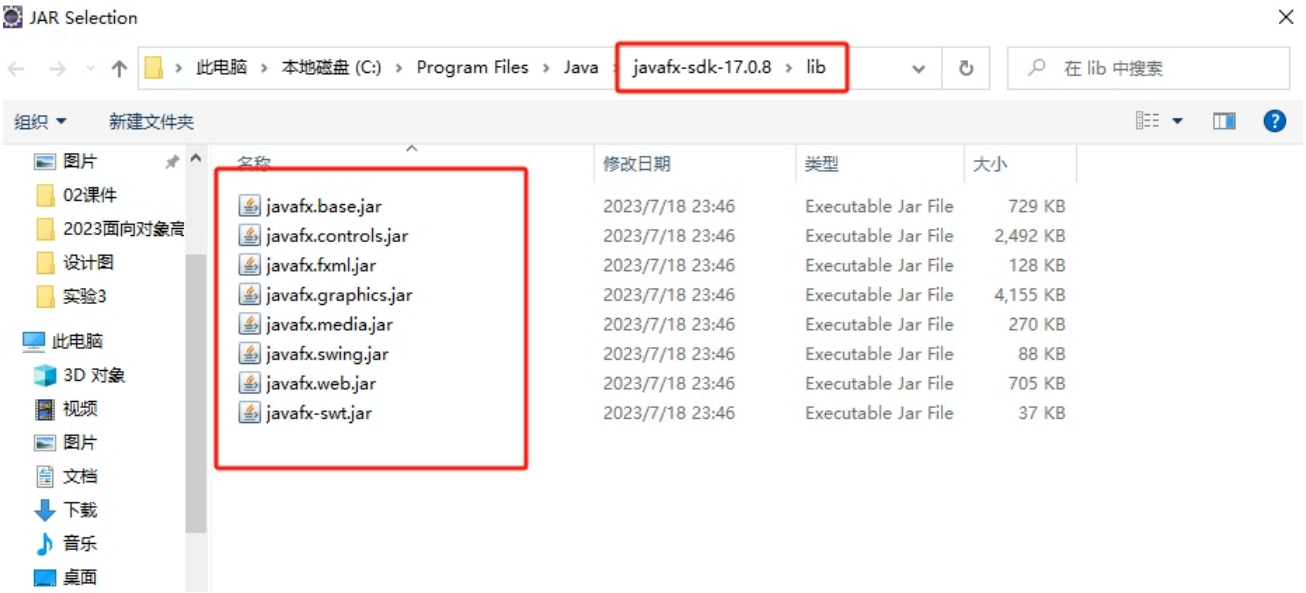


3、若不存在，则添加一个用户库


Add Library
User Library
Select a library to add to the classpath.
User libraries:
JavaFX17
JavaFX17New
User Libraries...
< Back
Next >
Finish
Cancel


Preferences (Filtered)
type filter text
Java
Build Path
User Libraries
User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.
Defined user libraries:
JavaFX17
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...
New User Library
User library name:
javafx17
System library (added to the boot class path)
OK
Cancel
Apply and Close
Cancel


Preferences (Filtered)
type filter text
Java
Build Path
User Libraries
User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.
Defined user libraries:
javafx17
JavaFX17
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...
Apply and Close
Cancel


JAR Selection
此电脑 › 本地磁盘 (C:) › Program Files › Java › javafx-sdk-17.0.8 › lib
在 lib 中搜索
组织
新建文件夹
图片
02课件
2023面向对象高
设计图
实验3
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
名称
修改日期
类型
大小
javafx.base.jar 2023/7/18 23:46 Executable Jar File 729 KB
javafx.controls.jar 2023/7/18 23:46 Executable Jar File 2,492 KB
javafx.fxml.jar 2023/7/18 23:46 Executable Jar File 128 KB
javafx.graphics.jar 2023/7/18 23:46 Executable Jar File 4,155 KB
javafx.media.jar 2023/7/18 23:46 Executable Jar File 270 KB
javafx.swing.jar 2023/7/18 23:46 Executable Jar File 88 KB
javafx.web.jar 2023/7/18 23:46 Executable Jar File 705 KB
javafx-swt.jar 2023/7/18 23:46 Executable Jar File 37 KB

模块代码：

```
module MyTest {
        requires javafx.graphics;
        requires javafx.controls;
        exports exp3.tetris.gui;
        exports exp3.tetris.biz;
        opens exp3.tetris.gui to javafx.graphics;
}
```

注意：模块名和自己的项目保持一致

4 实验要求

4.1 实验评价

1、必须完成实验3.1的所有步骤，实验中的程序能正常运行。
2、遇到问题时能及时与指导老师沟通并解决问题。

4.2 实验报告

本次实验以验证为主，不需要提交实验报告。

5 实验教学录屏

仅供预习和复习参考，实验内容和要求以正式课堂为准。
https://www.bilibili.com/video/BV1TG411m7cv/