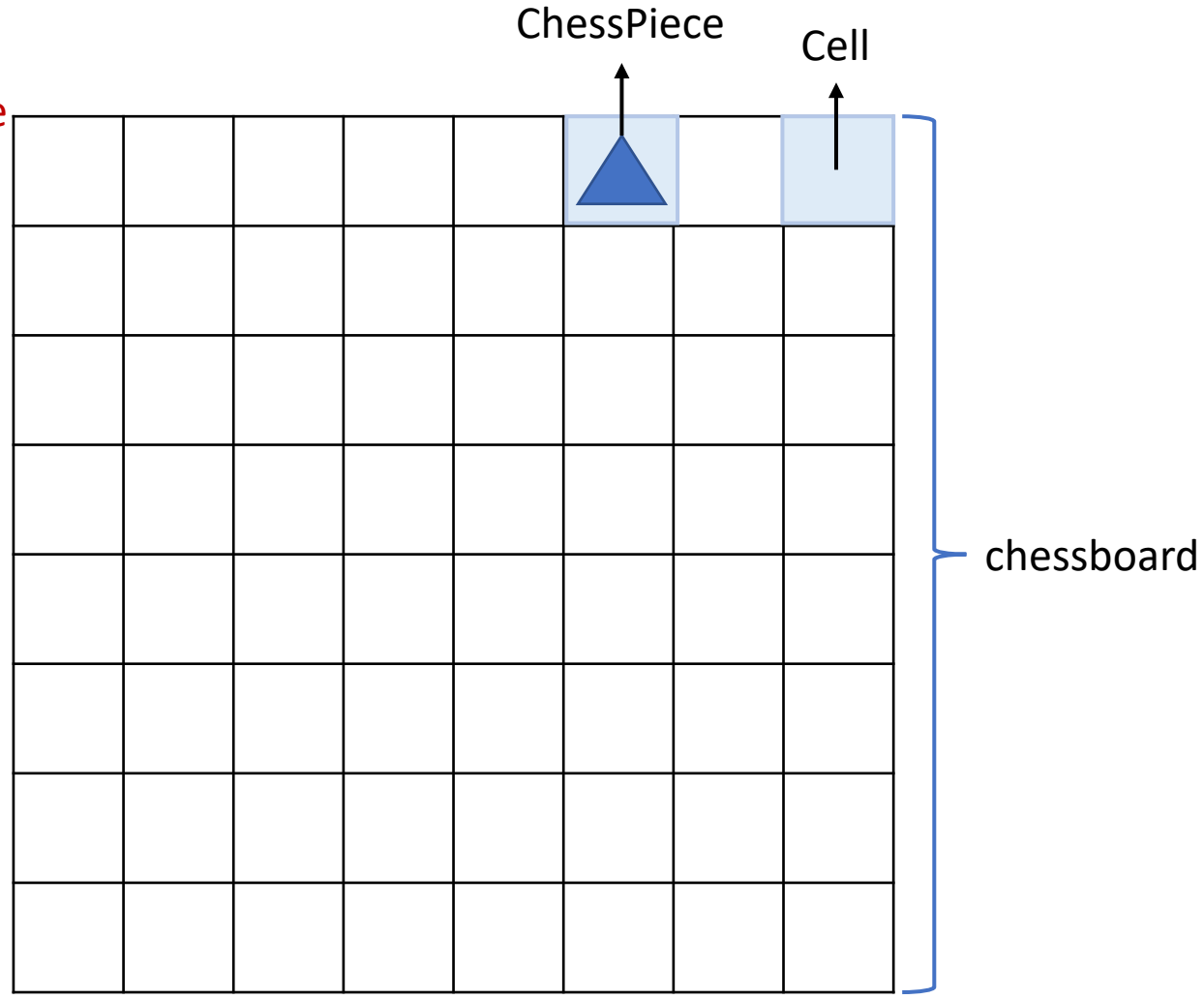
了解Demo之前要掌握的基础

- Swing的窗体、组件的基本关系
- Swing布局中，关于x,y坐标点的绝对布局
- Swing绘制中的paintComponent() 与 repaint();
- Swing 中的addListener监听→ 点击Jcomponent 触发的事件
- Swing中点击button触发事件

- model
  - Cell　表示每一个小方格，包含一个ChessPiece
  - Chessboard　包含8*8的Cell类型数组
  - ChessboardPoint　表示每个小方格的坐标点
  - ChessPiece　表示图形
  - Constant　定义一些全局的参数
  - Util　定义一个对于任何类型的数组随机返回一种一个元素的方法



Model层中，我们决定了棋盘大小、棋盘中每一个小方格到底要存放哪些图形。

我们要确保在设计程序时，chessboard要实时更新（比如交换棋子、消除棋子、生成新棋子等）当前棋盘状态，每一个小方格中存放棋子的内容，

Chessboard.initGrid()

- 这个方法里创建8*8的Cell类型数组
- 这个方法建议不用改动，无论restart程序几回，只实行一次就好。

Chessboard.initPieces

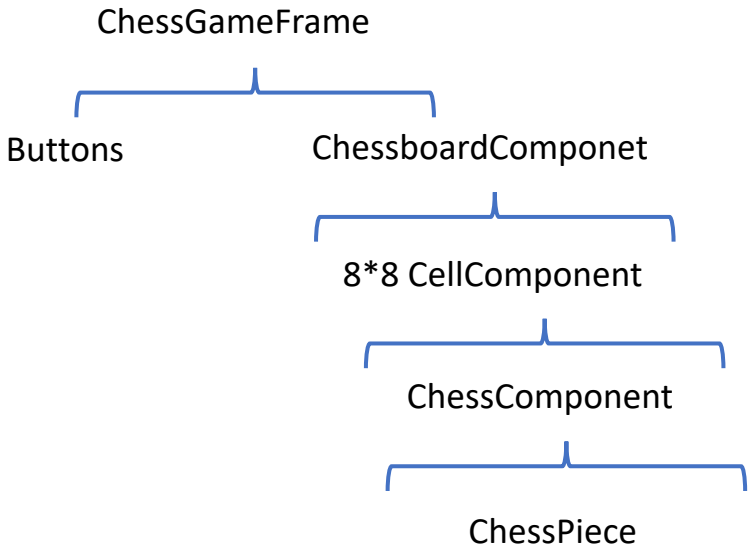
- 这个方法是在8*8 的二维数组的Cell已经创建好的基础上，给每一个Cell中添加一个ChessPiece
- 这个方法在restart功能中可以调用，调用途径：
    1. 点击button调用GameController中的initialize 方法
    2. 在initialize方法中，调用model.initPieces() 方法

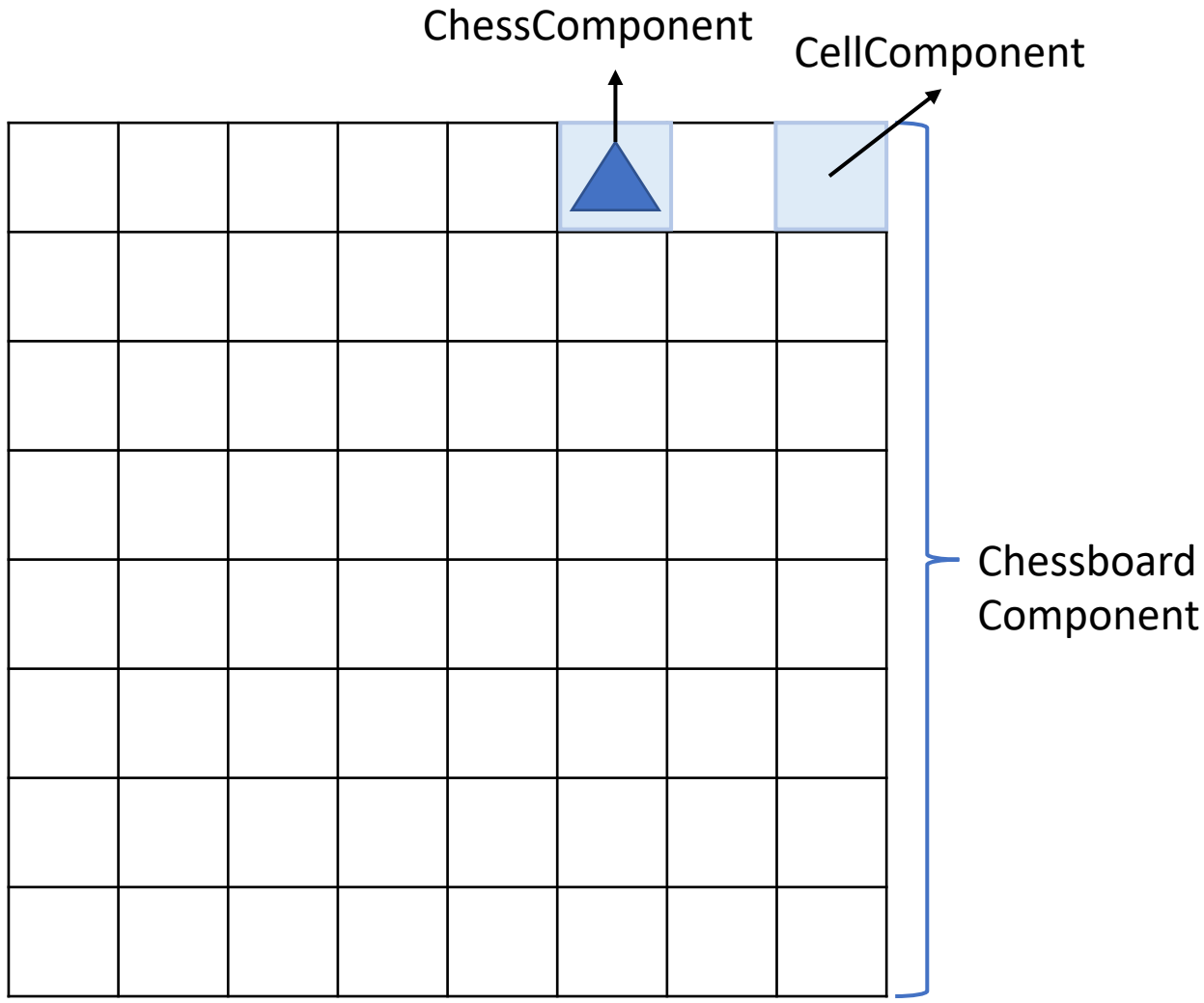
setChessPiece
removeChessPiece

- 在Chessboard的某一个Cell里，放置ChessPiece与移除ChessPiece可以直接用这两个方法。

- view
  - CellComponent 包含paint格子的方法
  - ChessboardComponent 包含一个8*8的CellComponent数组
  - ChessComponent 包含一个Chesspiece与paint棋子的方法
  - ChessGameFrame 表示窗体，里面包含一个ChessboardComponent，与一系列按钮

View层中，我们主要根据model层的chessboard进行绘制操作。

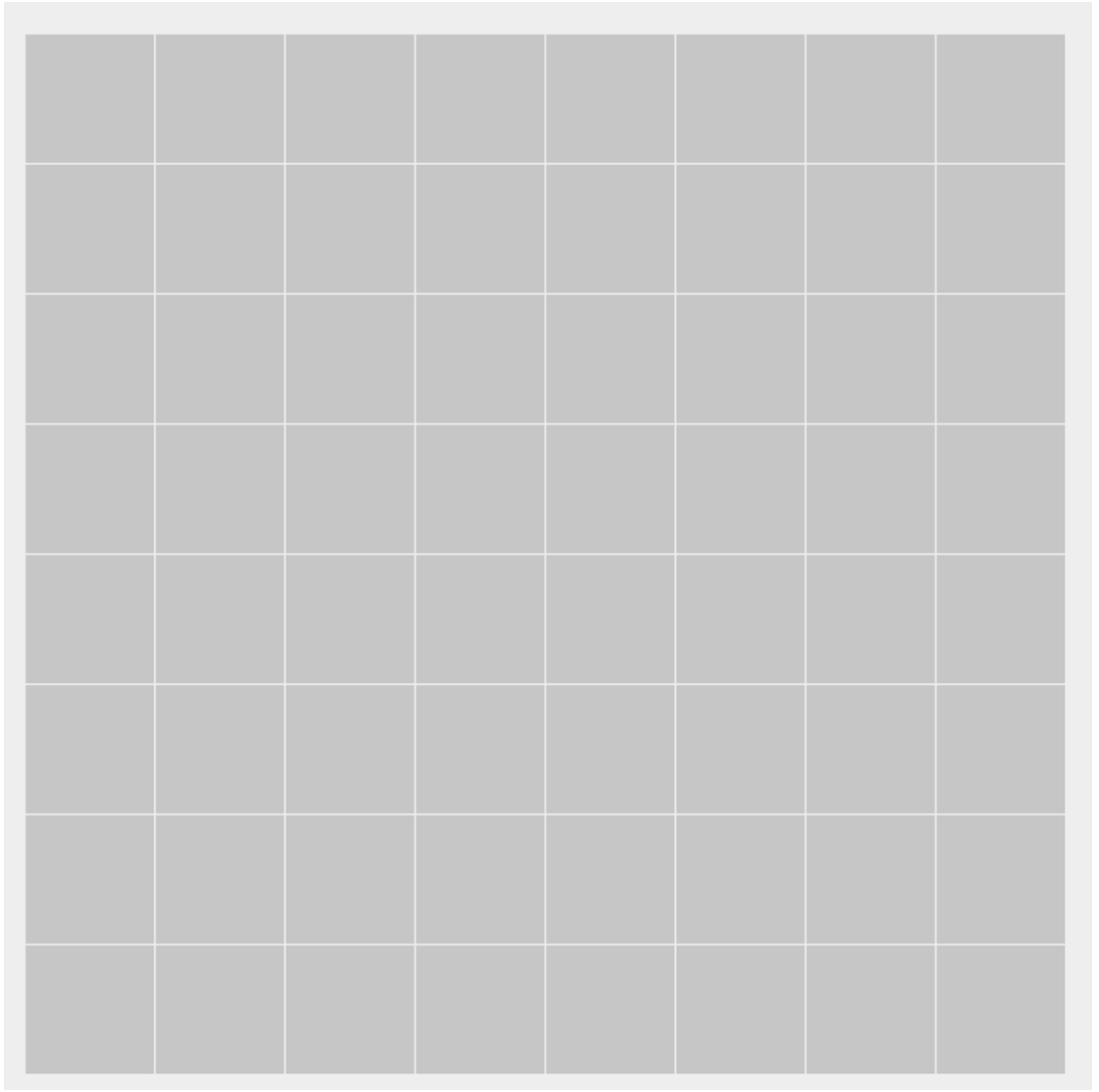




initiateGridComponents

这个方法就是在ChessboardComponent中创建了8*8的CellComponent。 单独执行像右边所示：

这个方法建议不用改动，无论restart程序几回，只实行一次就好。



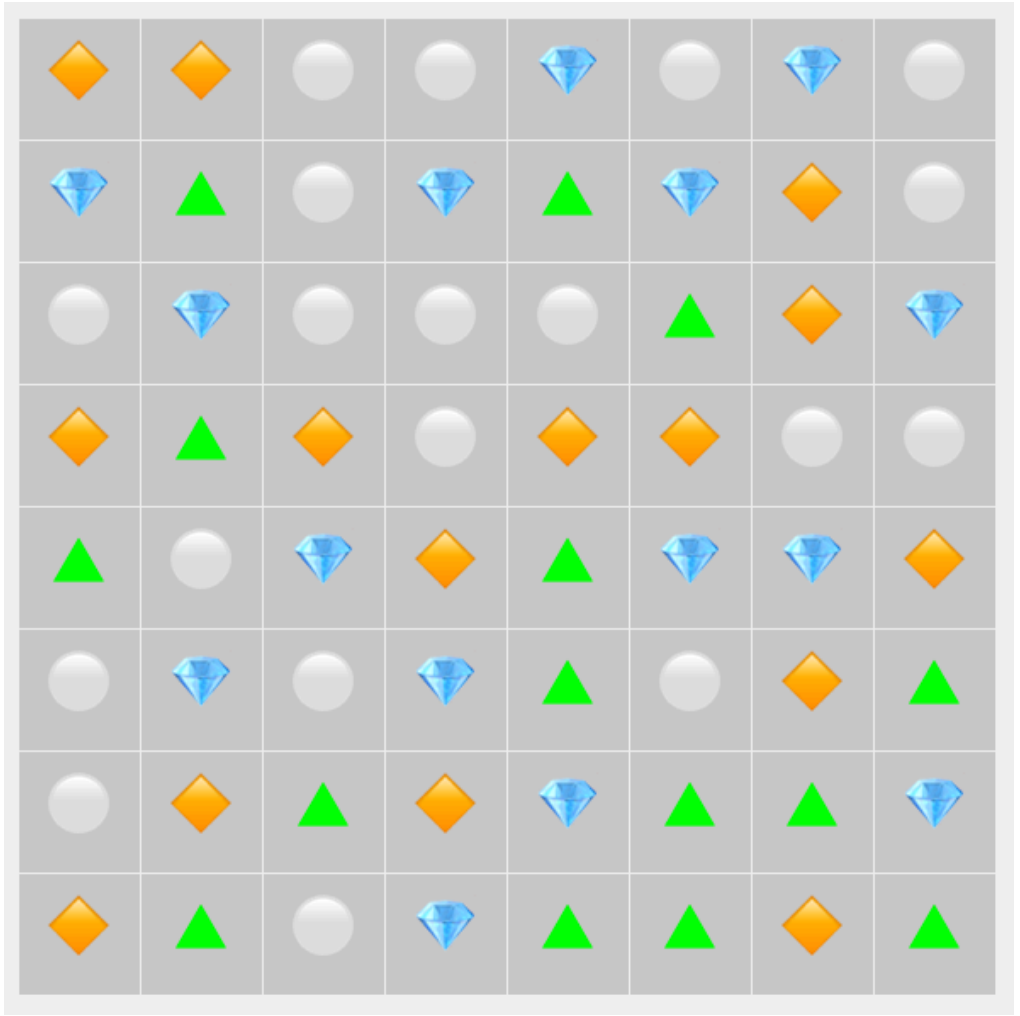
initiateChessComponent

这个方法是在8*8CellComponent已经创建好的基础上，将Chessboard里的内容，绘制在每一个小方格上。

- 这个方法需要传递参数 Chessboard chessboard
- 根据参数chessboard，获取Cell[][] grid = chessboard.getGrid();
- 再看每一个grid里是否有chessPiece：

```
if (grid[i][j].getPiece() != null) {
```

每重新restart一次游戏且清空棋盘之后，都可以根参数据chessboard的内容，生成一个新的棋盘



removeChessComponentAtGrid

这个方法是清空ChessboardComponent组件上的point位置的ChessComponent

对于交换棋子，需要先清空两个位置的ChessComponent组件，再重新add进去

对于restart游戏，需要遍历每一个元素，调用这个方法，来清空ChessboardComponet 中每一个CellComponent中的ChessComponent组件。

所有对画面改变的操作，都要进行repaint(); 重新绘制在能体现在页面上

```
public ChessComponent removeChessComponentAtGrid(ChessboardPoint point) {
    // Note re-validation is required after remove / removeAll.
    ChessComponent chess = (ChessComponent) getGridComponentAt(point).getComponents()[0];
    getGridComponentAt(point).removeAll();
    getGridComponentAt(point).revalidate();
    chess.setSelected(false);
    return chess;
}
```

setChessComponentAtGrid

- 将ChessComponent放在指定point位置的CellComponent上
- 这个方法可以用于交换棋子，移动棋子等

```
public void setChessComponentAtGrid(ChessboardPoint point, ChessComponent chess) {
    getGridComponentAt(point).add(chess);
}
```

GameController

GameController是用于设计事件触发方法的，整个游戏可分两类事件：

- 1. 鼠标点击事件触发的方法
  - onPlayerSwapChess
  - onPlayerNextStep
  - Initialize（可用于restart按钮触发的方法）
- 2. 点击棋子事件触发的方法
  - onPlayerClickCell
  - onPlayerClickChessPiece

```
1 usage
private Chessboard model;
8 usages
private ChessboardComponent view;
```

包含一个Chessboard与一个ChessboardComponent

- 方便两者之间同步绘制
- 方便调用写在两者之间的所有子类与子组件

```
// Record whether there is a selected piece before
16 usages
private ChessboardPoint selectedPoint;
16 usages
private ChessboardPoint selectedPoint2;
```

表示第一次点击的位置

表示第二次点击的位置

ChessGameFrame

- 所有点击addHelloButton要执行的java语句，都写在(e)->{}这个大括号内

```
private void addHelloButton() {
    JButton button = new JButton("Show Hello Here");
    button.addActionListener((e) -> {
        JOptionPane.showMessageDialog(this, "Hello, world!");
    });
    button.setLocation(HEIGTH, HEIGTH / 10 + 120);
    button.setSize(200, 60);
    button.setFont(new Font("Rockwell", Font.BOLD, 20));
    add(button);
}
```