南开大学

JAVA 语言与应用

控制台版五子棋作业实验报告

姓 名：冯朝芃

学 号：2012039

年 级： 2020 级

学 院： 计算机学院

专 业 ：计算机科学与技术

授课教师：刘嘉欣

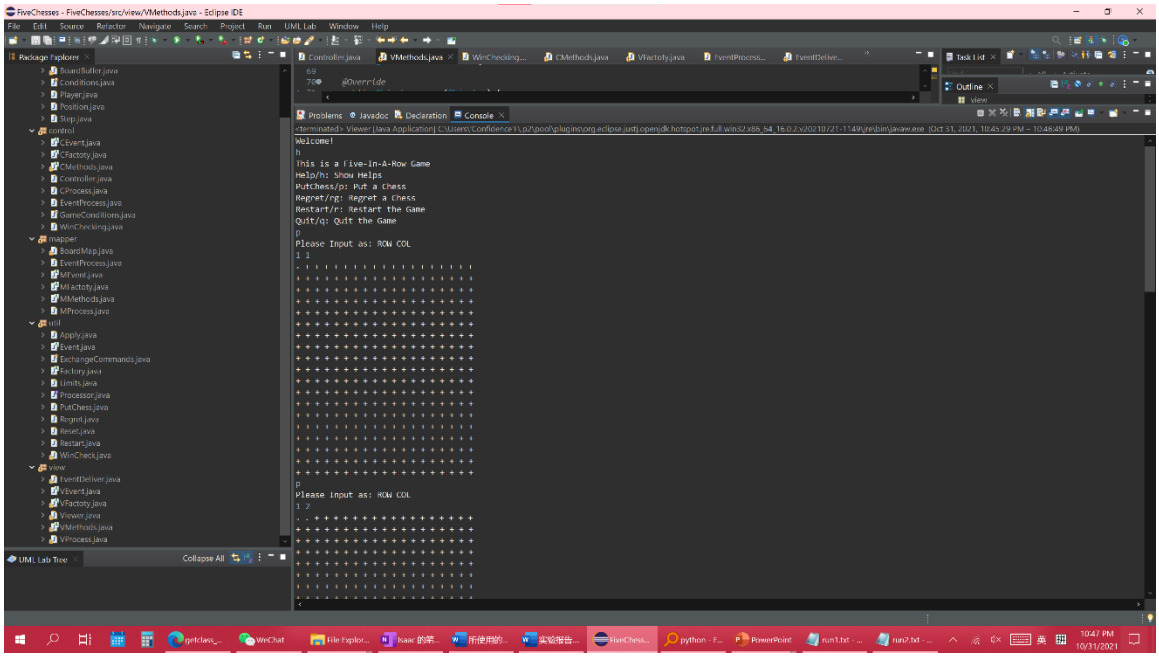
完成日期：2021 年 10 月 31 日

一、概述：

本作业为控制台版 Java 语言五子棋。本作业实现的功能有：人人对战（PutChess）、判断是否胜利、重置棋局、重置获胜条件（n 子棋）、悔棋、重新开始、帮助、退出等功能。本代码使用了 MVC 架构进行开发，并设置了抽象事件（Event）、抽象事件工厂（Factory）、抽象处理（Apply）应用、处理接口（Process）等。本程序充分利用 Java 的面向对象编程思想，一定程度上具有可扩展性高，代码逻辑框架清晰。代码复用性强、事件处理流程明确等特点。

二、运行展示：

运行效果截图:



运行结果展示:

1、悔棋:

Welcome!

p

Please Input as: ROW COL

1 1

。+++++++++++++++++++

（省略部分棋盘）

p

Please Input as: ROW COL

1 2

。.+++++++++++++++++

（省略部分棋盘）

1 3

No this Command! Check Your Input!

rg　　//悔棋命令

Regret One Step:

。+++++++++++++++++

（省略部分棋盘）

rg

Regret One Step:

+++++++++++++++++++

（省略部分棋盘）

rg

Regret Fail!　　//过度悔棋，报错

2、n 子棋:

Welcome!　　+++++++++++++++++++

r　　+++++++++++++++++++

Restarting　　+++++++++++++++++++

Welcome!　　+++++++++++++++++++

rs　　+++++++++++++++++++

Please input a number:　　+++++++++++++++++++

2　　//重新设置为 2 子棋　　+++++++++++++++++++

p　　+++++++++++++++++++

Please Input as: ROW COL　　+++++++++++++++++++

1 1　　+++++++++++++++++++

。+++++++++++++++++　　+++++++++++++++++++

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

1 2

。. + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

2 1

。. + + + + + + + + + + + + + + + + +

。+ + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

White Wins!　//白棋连成两子 胜利

Do you Want Start Again!?[y/n]

y　　//重新启动，再来一盘

Welcome!

3、五子连珠判断胜利完整展示

Welcome!

p

Please Input as: ROW COL

1 1

。++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

p

Please Input as: ROW COL

1 2

。.++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

p

Please Input as: ROW COL

2 1

。.++++++++++++++++++

。++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

1 3

。 . . + + + + + + + + + + + + + + + +

。 + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

2 2

。 . . + + + + + + + + + + + + + + + +

。 。 + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

1 3　　//不能在已经被占用的棋盘位置下棋，设置抛出异常

java.lang.IllegalArgumentException: Input Position is Occupied!

p

Please Input as: ROW COL

1 4

。...+++++++++++++++++

。。+++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

p

Please Input as: ROW COL

3 3

。...+++++++++++++++++

。。+++++++++++++++++++

++。+++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

p

Please Input as: ROW COL

1 5

。....++++++++++++++

。。+++++++++++++++++

++。+++++++++++++++

++++++++++++++++++++

++++++++++++++++++++

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

4 4

。 . . . . + + + + + + + + + + + + + + +

。 。 + + + + + + + + + + + + + + + + +

+ + 。 + + + + + + + + + + + + + + + +

+ + + 。 + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

p

Please Input as: ROW COL

1 6

。 . . . . . + + + + + + + + + + + + + +

。 。 + + + + + + + + + + + + + + + + +

+ + 。 + + + + + + + + + + + + + + + +

+ + + 。 + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

Black Wins!　//判断胜利正确

Do you Want Start Again!?[y/n]

4、重启和退出

Welcome!

p

Please Input as: ROW COL

1 1

。+ + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

r　　//重启命令

Restarting　　//重启游戏

Welcome!

p

Please Input as: ROW COL

10 10

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + 。+ + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

+ + + + + + + + + + + + + + + + + + +

q　　//退出命令

Quiting

三、代码选讲（完整代码见附录）：

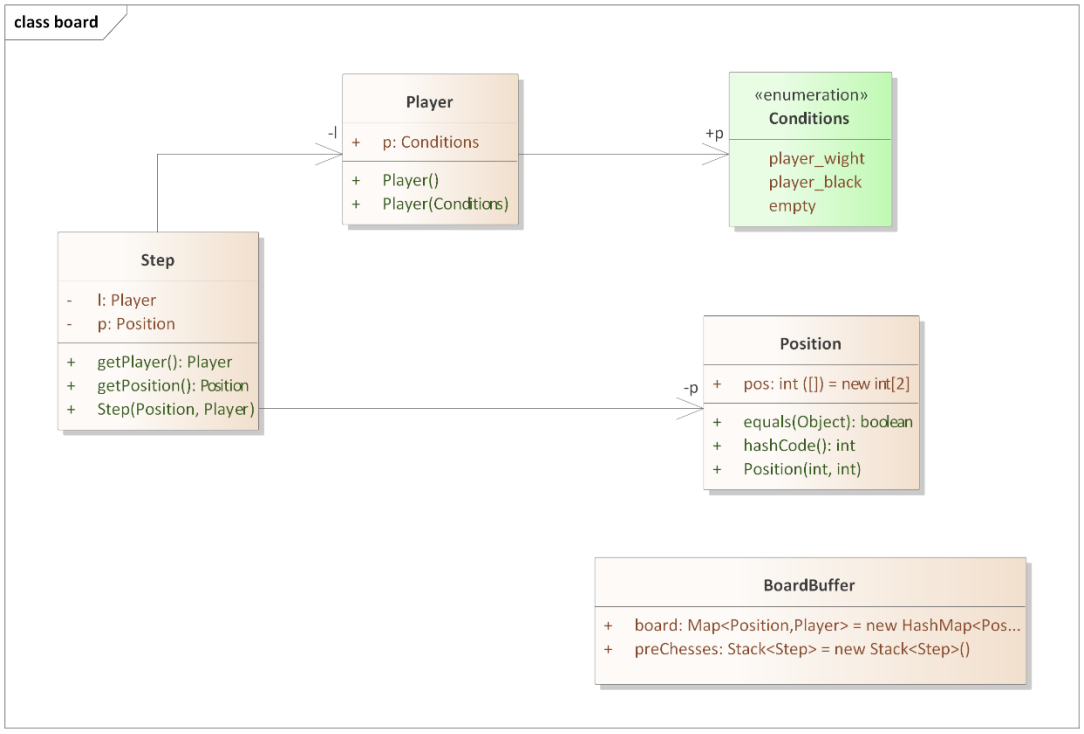
1.view. EventDeliver.java

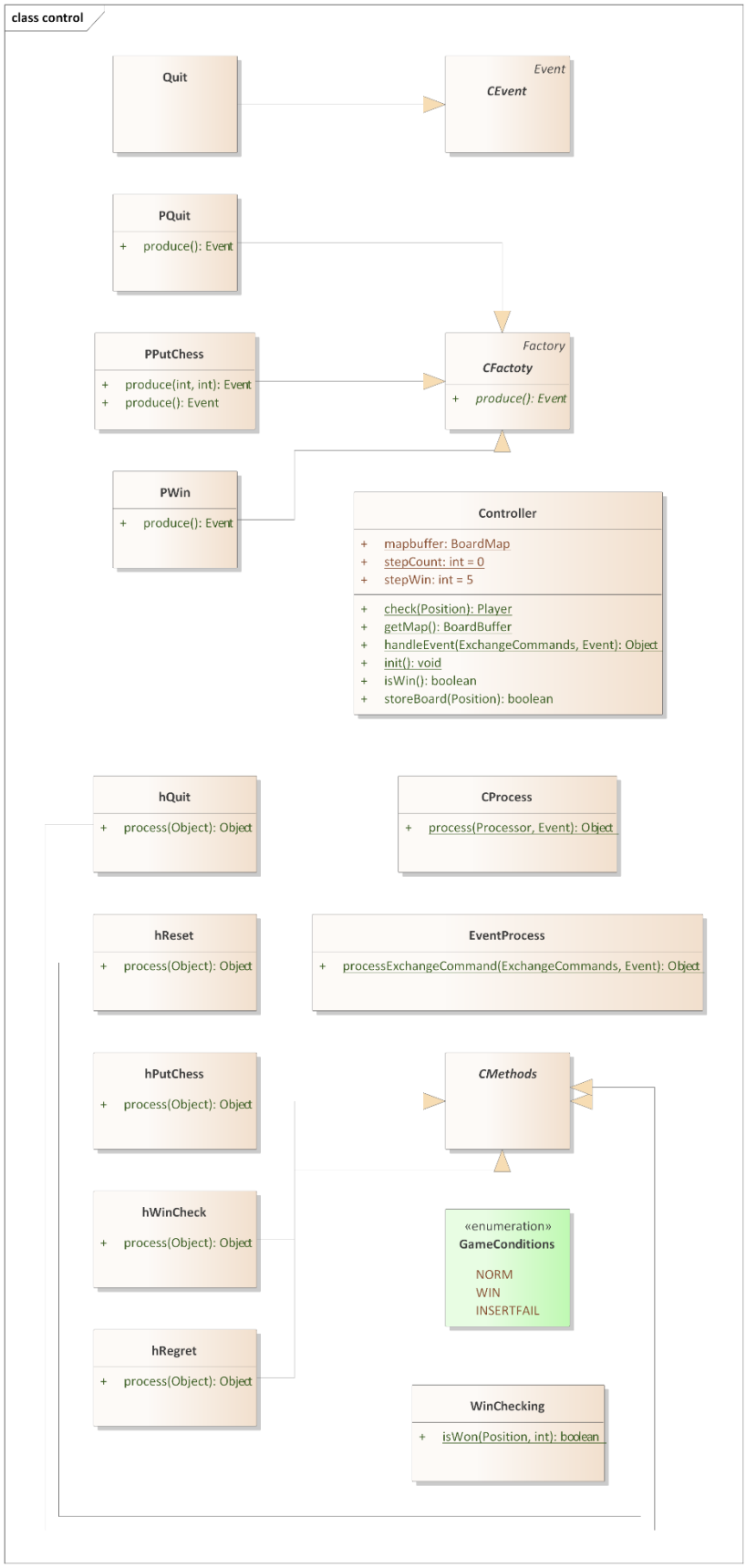
```
package view;


import control.CProcess;
import control.Controller;
import util.*;


public class EventDeliver {
	public static Object processCommand(String in) {
		switch (in) {//处理用户输入的字符串
		case "h":
		case "Help"://适配器模式+抽象工厂模式：VProcess 适配器继承自 util.Process 接口，将真正的处理程序 hHelp 和事件 Help 联系起来。事件 Help 由抽象工厂（由 util.Factory 派生）派生的 PHelp.produce（）创造。
			return VProcess.process(new hHelp(), new PHelp().produce());
		case "q":
		case "Quit"://以下事件处理方式与上述类似
			return VProcess.process(new hQuit(), new PQuit().produce());
		case "p":
		case "PutChess":
			try {
				return VProcess.process(new hPutChess(), new PPutChess().produce());
			} catch (IllegalArgumentException e2) {
				return VProcess.process(new hWrongNum(), e2);
			}
		case "rg":
		case "Regret":
```

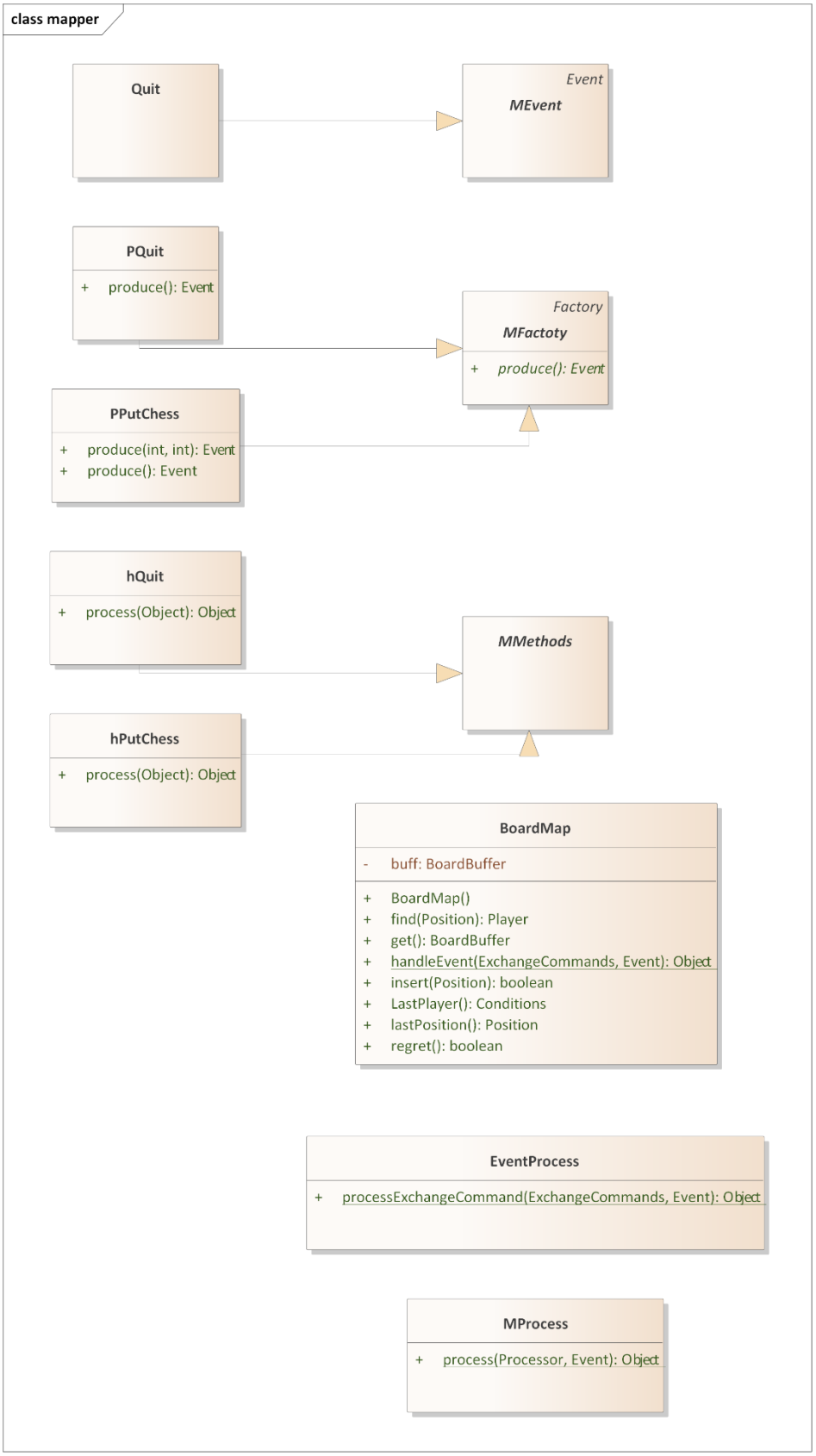
```
                        //return
processExchangeCommand(ExchangeCommands.REGRET, new Regret());
                        return VProcess.process(new hRegret(), new Regret());
                case "r":
                case "Restart":
                        return VProcess.process(new hRestart(), new Restart());
                case "rs":
                case "Reset":
                        //return processExchangeCommand(ExchangeCommands.RESET,
new Reset());
                        return VProcess.process(new hReset(), new Reset());
                default:
                        return VProcess.process(new hWrongCommand(), new
PWrongCommand().produce());
                }
        }
        public static Object processExchangeCommand(ExchangeCommands in,Event e) {
                switch (in) {//这个函数是 Viewer 与 Mapper、Controllor 沟通的桥梁，
用于处理 ExchangeCommands（全局事件）
                case QUIT:
                        //return MProcess.process(new hQuit(), new PQuit().produce());
                        return null;
                case SHOW:
                        Viewer.outputBoard();
                        return null;
                case AfterWin:
                        Viewer.afterWin(e);
                        return null;
                case REGRET:
                case RESET:
```

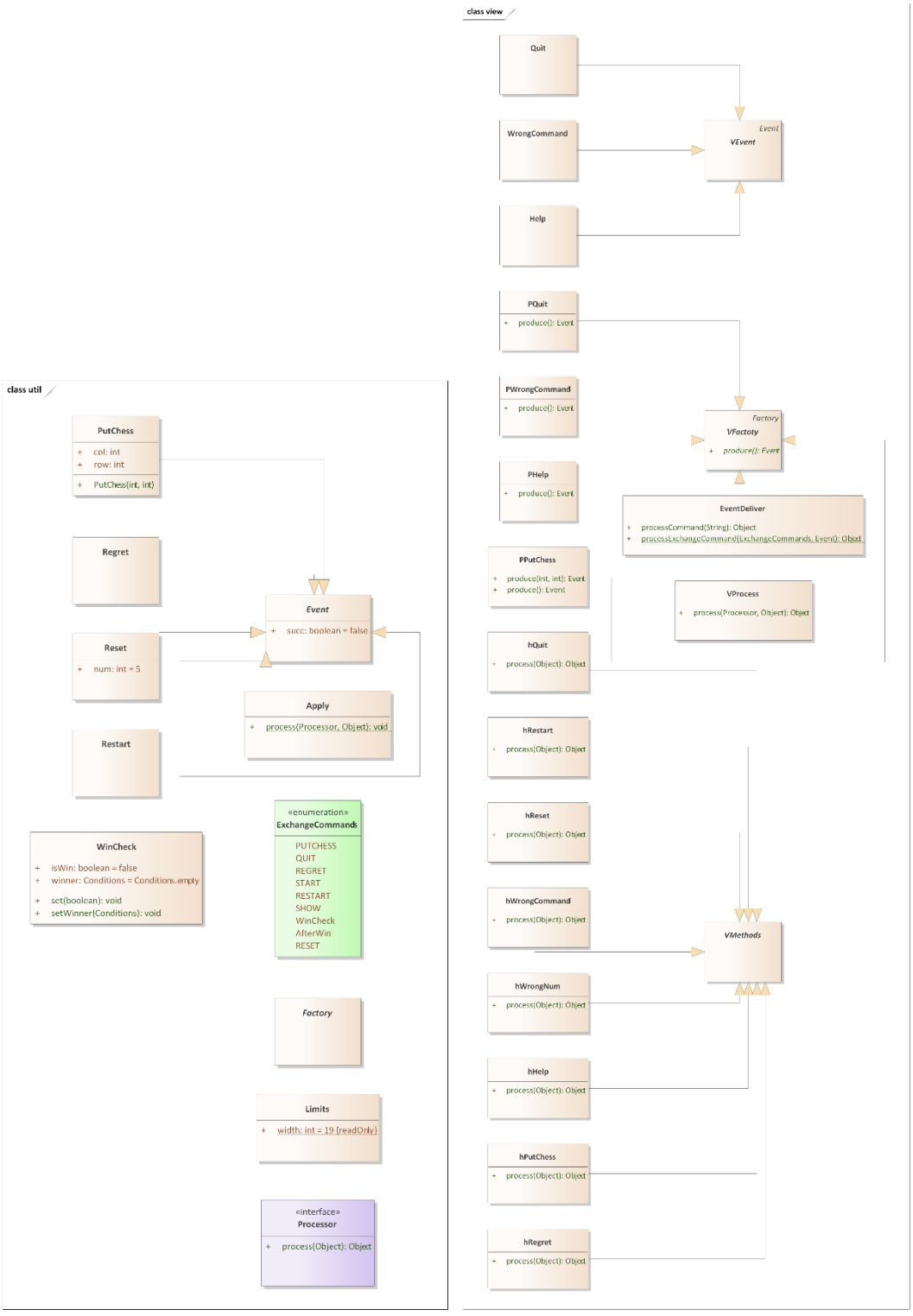
```
                return Controller.handleEvent(in, e);

            default:
                return null;
            }
        }
}
```

四、类图展示：




class control
Quit
Event
CEvent
PQuit
+ produce(): Event
PPutChess
+ produce(int, int): Event
+ produce(): Event
Factory
CFactoty
+ produce(): Event
PWin
+ produce(): Event
Controller
+ mapbuffer: BoardMap
+ stepCount: int = 0
+ stepWin: int = 5
+ check(Position): Player
+ getMap(): BoardBuffer
+ handleEvent(ExchangeCommands, Event): Object
+ init(): void
+ isWin(): boolean
+ storeBoard(Position): boolean
hQuit
+ process(Object): Object
CProcess
+ process(Processor, Event): Object
hReset
+ process(Object): Object
EventProcess
+ processExchangeCommand(ExchangeCommands, Event): Object
hPutChess
+ process(Object): Object
CMethods
hWinCheck
+ process(Object): Object
«enumeration»
GameConditions
NORM
WIN
INSERTFAIL
hRegret
+ process(Object): Object
WinChecking
+ isWon(Position, int): boolean


class mapper
Quit
Event
MEvent
PQuit
+ produce(): Event
Factory
MFactoty
+ produce(): Event
PPutChess
+ produce(int, int): Event
+ produce(): Event
hQuit
+ process(Object): Object
MMethods
hPutChess
+ process(Object): Object
BoardMap
- buff: BoardBuffer
+ BoardMap()
+ find(Position): Player
+ get(): BoardBuffer
+ handleEvent(ExchangeCommands, Event): Object
+ insert(Position): boolean
+ LastPlayer(): Conditions
+ lastPosition(): Position
+ regret(): boolean
EventProcess
+ processExchangeCommand(ExchangeCommands, Event): Object
MProcess
+ process(Processor, Event): Object


class util
PutChess
+ col: int
+ row: int
+ PutChess(int, int)
Regret
Reset
+ num: int = 5
Restart
Event
+ succ: boolean = false
Apply
+ process(Processor, Object): void
WinCheck
+ isWin: boolean = false
+ winner: Conditions = Conditions.empty
+ set(boolean): void
+ setWinner(Conditions): void
«enumeration»
ExchangeCommands
PUTCHESS
QUIT
REGRET
START
RESTART
SHOW
WinCheck
AfterWin
RESET
Factory
Limits
+ width: int = 19 {readOnly}
«interface»
Processor
+ process(Object): Object
class view
Quit
WrongCommand
Help
Event
VEvent
PQuit
+ produce(): Event
PWrongCommand
+ produce(): Event
PHelp
+ produce(): Event
Factory
VFactoty
+ produce(): Event
EventDeliver
+ processCommand(String): Object
+ processExchangeCommand(ExchangeCommands, Event): Object
PPutChess
+ produce(int, int): Event
+ produce(): Event
VProcess
+ process(Processor, Object): Object
hQuit
+ process(Object): Object
hRestart
+ process(Object): Object
hReset
+ process(Object): Object
hWrongCommand
+ process(Object): Object
VMethods
hWrongNum
+ process(Object): Object
hHelp
+ process(Object): Object
hPutChess
+ process(Object): Object
hRegret
+ process(Object): Object

五、附录：完整代码

＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝

『BoardBuffer.java』

＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝

```
package board;

import java.util.*;
import mapper.*;

public class BoardBuffer {

	public Map<Position,Player> board=new HashMap<Position,Player>();

	public Stack<Step> preChesses=new Stack<Step>();



}
```

＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝

『Conditions.java』

＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝

```
package board;

public enum Conditions {
	player_wight,player_black,empty;
}
```

＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝

『Player.java』

＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝ ＝
＝ ＝ ＝ ＝ ＝ ＝ ＝

```
package board;

public class Player{

	public Conditions p;

	public Player() {
		p=Conditions.empty;
	}

	public Player(Conditions c) {
		p=c;
	}
```

```
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Position.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package board;

public class Position {
	public Position(int i, int j) {
		pos[0]=i;
		pos[1]=j;
	}

	public int[] pos=new int[2];

	@Override
	public int hashCode() {
		return new Integer(pos[0]).hashCode()+new Integer(pos[1]).hashCode();
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
	    return true;
	  if (obj == null)
	    return false;
	  if (getClass() != obj.getClass())
	    return false;

	  Position otherPosition=(Position) obj;
		return pos[0]==otherPosition.pos[0]&&pos[1]==otherPosition.pos[1];
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Step.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package board;

public class Step {

	private Position p;
	private Player l;

	public Step(Position pos,Player pl) {
```

```java
		p=pos;
		l=pl;
	}

	public Player getPlayer() {
		return l;
	}

	public Position getPosition() {
		// TODO Auto-generated method stub
		return p;
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『CEvent.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```java
package control;

import util.Event;

public abstract class CEvent extends Event{

}

class Quit extends CEvent{

}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『CFactoty.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```java
package control;

import util.*;

public abstract class CFactoty extends Factory{
	public abstract Event produce();
	//public abstract Event produce(int a,int b);
}

class PQuit extends CFactoty{

	@Override
	public Event produce() {
		return new Quit();
	}
}
```

```
class PPutChess extends CFactoty{

	public Event produce(int r,int c) {
		return new PutChess(r,c);
	}

	@Override
	public Event produce() {
		// TODO Auto-generated method stub
		return null;
	}
}

class PWin extends CFactoty{

	@Override
	public Event produce() {
		return new WinCheck();
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『CMethods.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package control;

import java.nio.Buffer;

import board.Position;
import control.Controller;
import mapper.BoardMap;
import util.*;

public abstract class CMethods implements Processor{

}

class hQuit extends CMethods{

	@Override
	public Object process(Object e) {
	System.out.println("Quiting");
		return e;
	}
}

class hReset extends CMethods{

	@Override
	public Object process(Object e) {
```

```
                Reset
tmpReset=(Reset)e;
                if(!tmpReset.succ) return
e;


        Controller.stepWin=tmpReset.nu
m;
                return e;
        }
}

class hPutChess extends CMethods{


        @Override
        public Object process(Object e) {
                PutChess
tmPutChess=(PutChess)e;



        if(tmPutChess.row<=0||tmPutC
hess.row>Limits.width||tmPutChess.col
<=0||tmPutChess.col>Limits.width) {
                        throw new
IllegalArgumentException("Input
Position out of Bound!");
                }


                Position position=new
Position(tmPutChess.row,
tmPutChess.col);

        if(Controller.mapbuffer.insert(po
sition)) {


        Controller.stepCount++;
                        WinCheck
isWinCheck=(WinCheck)EventProcess.pr
ocessExchangeCommand(ExchangeCom
mands.WinCheck, new WinCheck());


        isWinCheck.succ=true;
                        return
isWinCheck;
                }else {


        //tmPutChess.succ=false;
                        //return
tmPutChess;
                        throw new
IllegalArgumentException("Input
Position is Occupied!");
                }



        //BoardMap.handleEvent(Excha
ngeCommands.PUTCHESS,(Event)e);


        }
}

class hWinCheck extends CMethods{
```

```
	@Override
	public Object process(Object e) {

		WinCheck winCheck=(WinCheck)e;

	winCheck.set(WinChecking.isWon(Controller.mapbuffer.lastPosition(),Controller.stepWin));

	winCheck.setWinner(Controller.mapbuffer.LastPlayer());

		return winCheck;
	}
}

class hRegret extends CMethods{

	@Override
	public Object process(Object e) {

		Event tmpEvent=(Event)e;
		if (Controller.stepCount==0) {

	tmpEvent.succ=false;
			return tmpEvent;
		}
	if(Controller.mapbuffer.regret())
{

	Controller.stepCount--;

	tmpEvent.succ=true;
			return tmpEvent;
		}

		return null;
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Controller.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package control;


import board.*;
import mapper.*;
import util.*;

public class Controller {
```

```
	public static BoardMap
mapbuffer;
	public static int stepCount=0;
	public static int stepWin=5;

	public static Object
handleEvent(ExchangeCommands
x,Event e) {
		try {
			return
EventProcess.processExchangeComman
d(x,e);
		}catch(Exception e1) {
			throw e1;
		}
	}

	public boolean
storeBoard(Position p) {

		return
mapbuffer.insert(p);
	}

	public boolean isWin() {

		return true;
	}

	public static void init() {
		mapbuffer=new
BoardMap();
		stepCount=0;
		stepWin=5;
	}

	public static BoardBuffer
getMap() {

		return mapbuffer.get();
	}

	public static Player
check(Position tmpPos) {
		return
mapbuffer.find(tmpPos);
	}


}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『CProcess.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package control;

import util.Event;
```

```
import util.Processor;


public class CProcess{
	public static Object process(Processor p,Event e) {
		return p.process(e);
	}
}
```

= =

『EventProcess.java』

= =

```
package control;


import util.Event;
import util.ExchangeCommands;


public class EventProcess {
	public static Object processExchangeCommand(ExchangeCommands in,Event e) {
		switch (in) {
		case QUIT:
			//return MProcess.process(new hQuit(), new PQuit().produce());
			return null;
		case WinCheck:
			return CProcess.process(new hWinCheck(), e);
		case PUTCHESS:
			try {
				return CProcess.process(new hPutChess(), e);


			} catch (Exception e2) {
				throw e2;
			}
		case REGRET:
			return CProcess.process(new hRegret(), e);
		case RESET:
			return CProcess.process(new hReset(), e);
		default:
			return null;
		}
	}
}
```

= =

『GameConditions.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package control;

public enum GameConditions {
	NORM,WIN,INSERTFAIL;
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『WinChecking.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package control;

import board.*;
import util.*;

public class WinChecking {
	public static boolean isWon(Position last,int stepWin) {
		int count = 1;     //本身一点为 1
		int posX = 0;
		int posY = 0;

		int WinSteps=stepWin;
		int x=last.pos[0];
		int y=last.pos[1];
		BoardBuffer boardBuffer=Controller.mapbuffer.get();
		Conditions color=boardBuffer.board.get(last).p;
		/**判断水平方向上的胜负
		/* 将水平方向以传入的点 x 上的 y 轴作为分隔线分为两部分
		 * 先向左边遍历，判断到的相同的连续的点 count++
		 */
		for(posX = x - 1; posX > 0 ; posX--) {
			Position tmPosition=new Position(posX, y);
			if (boardBuffer.board.get(tmPosition)!=null
					&& boardBuffer.board.get(tmPosition).p == color) {
				count++;
				if (count >= WinSteps) {
					return true;
				}
			}else {
				break;
			}
		}   //向右边遍历
```

```
    for(posX = x + 1; posX <= Limits.width; posX++) {
        Position tmPosition=new Position(posX, y);
        if (boardBuffer.board.get(tmPosition)!=null
         &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
               return true;
            }
        }else {
            break;
        }
    }
    count=1;
    /**判断垂直方向上的胜负
    /* 将垂直方向以传入的点 y 上的 x 轴作为分隔线分为两部分
     * 先向上遍历，判断到的相同的连续的点  count++
     */
    for(posY = y - 1; posY > 0; posY--) {
        Position tmPosition=new Position(x, posY);
        if (boardBuffer.board.get(tmPosition)!=null
         &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
               return true;
            }
        }else {
            break;
        }
    }//向下遍历
    for(posY = y + 1; posY <= Limits.width; posY++) {
        Position tmPosition=new Position(x, posY);
        if (boardBuffer.board.get(tmPosition)!=null
         &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
               return true;
            }
        }else {
            break;
        }
    }
    count=1;
    /**判断左上右下方向上的胜负
```

```
        * 以坐标点为分割线，将棋盘分为左右两个等腰三角形
        * 先判断左边的
        */
      for(posX = x - 1, posY = y - 1; posX > 0 && posY > 0; posX--, posY--) {
          Position tmPosition=new Position(posX, posY);
         if (boardBuffer.board.get(tmPosition)!=null
          &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
               count = 1;
               return true;
            }
         }else {
            break;
         }
      }//判断右边的
      for(posX = x + 1, posY = y + 1; posX <= Limits.width && posY <= Limits.width; posX++, posY++) {
          Position tmPosition=new Position(posX, posY);
         if (boardBuffer.board.get(tmPosition)!=null
          &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
               count = 1;
               return true;
            }
         }else {
            break;
         }
      }
      count=1;
      /**判断右下左下方向上的胜负
       * 以坐标点为分割线，将棋盘分为左右两个等腰三角形
       * 先判断左边的
       */
      for(posX = x + 1, posY = y - 1; posX <= Limits.width && posY > 0; posX++, posY--) {
          Position tmPosition=new Position(posX, posY);
         if (boardBuffer.board.get(tmPosition)!=null
          &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
               return true;
```

```
            }
        }else {
            break;
        }
    }//判断右边的
    for(posX = x - 1, posY = y + 1; posX > 0 && posY <= Limits.width; posX--, posY++) {
         Position tmPosition=new Position(posX, posY);
        if (boardBuffer.board.get(tmPosition)!=null

         &&boardBuffer.board.get(tmPosition).p == color) {
            count++;
            if (count >= WinSteps) {
                return true;
            }
        }else {
            break;
        }
    }
    return false;
  }
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『BoardMap.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package mapper;

import java.nio.Buffer;

import board.*;
import util.Event;
import util.ExchangeCommands;


public class BoardMap {
        private BoardBuffer buff;
        //public BoardMap mapbuffer=new BoardMap();
        //private Event lastPosEvent;

        public BoardMap() {
                buff=new BoardBuffer();

        buff.preChesses.push(new Step(null,new Player(Conditions.player_black)));//white first
        }

        public static Object handleEvent(ExchangeCommands x,Event e) {
```

```
                return EventProcess.processExchangeCommand(x,e);
        }

        public boolean insert(Position p) {

                if(buff.board.get(p)!=null) {
                        return false;
                }
                Conditions tmPlayer=this.buff.preChesses.peek().getPlayer().p==Conditions.player_black?Conditions.player_wight:Conditions.player_black;
                Player tmPlayer2=new Player(tmPlayer);

        buff.board.put(p,tmPlayer2);

        buff.preChesses.push(new Step(p,tmPlayer2));
                return true;
        }


        public BoardBuffer get() {

                return buff;
        }

        public Player find(Position tmpPos) {

                return this.buff.board.get(tmpPos);
        };

        public Position lastPosition() {
                return buff.preChesses.peek().getPosition();
        }
        public Conditions LastPlayer() {
                return buff.preChesses.peek().getPlayer().p;
        }

        public boolean regret() {

        if(buff.board.remove(buff.preChesses.peek().getPosition())==null||
        buff.preChesses.pop()==null)
                        return false;

                return true;
        }
```

}

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『EventProcess.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package mapper;


import util.Event;
import util.ExchangeCommands;


public class EventProcess {
	public static Object processExchangeCommand(ExchangeCommands in,Event e) {
		switch (in) {
		case QUIT:
			//return MProcess.process(new hQuit(), new PQuit().produce());
			return null;
		case PUTCHESS:
			return MProcess.process(new hPutChess(), e);
		default:
			return null;
		}
	}
```

}

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『MEvent.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package mapper;


import util.Event;


public abstract class MEvent extends Event{


}


class Quit extends MEvent{


}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『MFactoty.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package mapper;
```

```
import util.*;

public abstract class MFactoty extends Factory{
	public abstract Event produce();
	//public abstract Event produce(int a,int b);
}

class PQuit extends MFactoty{

	@Override
	public Event produce() {
		return new Quit();
	}
}

class PPutChess extends MFactoty{

	public Event produce(int r,int c) {
		return new PutChess(r,c);
	}

	@Override
	public Event produce() {
		// TODO Auto-generated method stub
		return null;
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『MMethods.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package mapper;

import board.Position;
import control.Controller;
import util.*;

public abstract class MMethods implements Processor{

}

class hQuit extends MMethods{

	@Override
	public Object process(Object e) {
		System.out.println("Quiting");
		return e;
	}
}
```

```
class hPutChess extends MMethods{

	@Override
	public Object process(Object e) {
		PutChess nEvent=(PutChess)e;
		Position position=new Position(nEvent.row,nEvent.col);

	Controller.mapbuffer.insert(position);
		return e;
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『MProcess.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package mapper;

import util.Event;
import util.Processor;

public class MProcess{
	public static Object process(Processor p,Event e) {
		return p.process(e);
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Apply.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public class Apply {
	public static void process(Processor p,Object i) {
		p.process(i);
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Event.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;
```

```
public abstract class Event {
	public boolean succ=false;
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『ExchangeCommands.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public enum ExchangeCommands {
	PUTCHESS,QUIT,REGRET,START,
RESTART,SHOW,WinCheck,
AfterWin,RESET;
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Factory.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public abstract class Factory {
	//public abstract Event
produce();
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Limits.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public class Limits {
	public static final int width=19;
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Processor.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public interface Processor {
	//String what();
	Object process(Object input);
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『PutChess.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public class PutChess extends Event{
	public int row;
	public int col;

	public PutChess(int r,int c){
		row=r;
		col=c;
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Regret.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public class Regret extends Event{
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Reset.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public class Reset extends Event{
	public int num=5;
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Restart.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

public class Restart extends Event{

}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『WinCheck.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package util;

import javax.sound.midi.VoiceStatus;

import board.Conditions;

public class WinCheck extends Event{
			public boolean isWin=false;
			public Conditions winner=Conditions.empty;

			public void set(boolean b) {
				isWin=b;
			}
			public void setWinner(Conditions winner) {
				this.winner = winner;
			}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『EventDeliver.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package view;

import control.CProcess;
import control.Controller;
import util.*;

public class EventDeliver {
	public static Object processCommand(String in) {
		switch (in) {
		case "h":
		case "Help":
			return VProcess.process(new hHelp(), new PHelp().produce());
		case "q":
		case "Quit":
			return VProcess.process(new hQuit(), new PQuit().produce());
		case "p":
		case "PutChess":
			try {
				return VProcess.process(new hPutChess(), new PPutChess().produce());
			} catch (IllegalArgumentException e2) {
```

```
                    return
VProcess.process(new hWrongNum(),
e2);
                }
            case "rg":
            case "Regret":
                //return
processExchangeCommand(ExchangeCo
mmands.REGRET, new Regret());
                return
VProcess.process(new hRegret(), new
Regret());
            case "r":
            case "Restart":
                return
VProcess.process(new hRestart(), new
Restart());
            case "rs":
            case "Reset":
                //return
processExchangeCommand(ExchangeCo
mmands.RESET, new Reset());
                return
VProcess.process(new hReset(), new
Reset());
            default:
                return
VProcess.process(new
hWrongCommand(), new
PWrongCommand().produce());
            }
        }
    public static Object
processExchangeCommand(ExchangeCo
mmands in,Event e) {
        switch (in) {
        case QUIT:
            //return
MProcess.process(new hQuit(), new
PQuit().produce());
            return null;
        case SHOW:

    Viewer.outputBoard();
            return null;
        case AfterWin:

    Viewer.afterWin(e);
            return null;
        case REGRET:
        case RESET:


            return
Controller.handleEvent(in, e);


        default:
            return null;
        }
    }
}
```

＝＝

『VEvent.java』

＝＝

```
package view;

import util.Event;

public abstract class VEvent extends Event{

}

class Quit extends VEvent{

}

class WrongCommand extends VEvent{}

class Help extends VEvent{}
```

＝＝

『VFactoty.java』

＝＝

```
package view;

import java.util.Scanner;

import util.*;

public abstract class VFactoty extends Factory{
	public abstract Event produce();
	//public abstract Event produce(int a,int b);
}

class PQuit extends VFactoty{

	@Override
	public Event produce() {
		return new Quit();
	}
}

class PWrongCommand extends VFactoty{

	@Override
	public Event produce() {
```

```
		return new
WrongCommand();
	}
}

class PHelp extends VFactoty{
	@Override
	public Event produce() {
		return new Help();
	}
}

class PPutChess extends VFactoty{

	public Event produce(int r,int c) {
		return new PutChess(r,c);
	}

	@Override
	public Event produce() {
		int r,c;

	System.out.println("Please Input
as: ROW COL");
		Scanner inScanner=new
Scanner(System.in);
		r=inScanner.nextInt();
		c=inScanner.nextInt();
		return produce(r,c);
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『Viewer.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package view;

import java.util.Iterator;
import java.util.Scanner;

import board.*;
import control.*;
import mapper.*;
import util.*;

public class Viewer {

	public static boolean
isFreshStart=true;

	public static void main(String[]
args) {
		startup();
```

```
                while (true) {
                        Scanner inputScanner=new Scanner(System.in);
                        String iString =inputScanner.next();

                        Object rst=EventDeliver.processCommand(iString);

        if(rst.getClass()==WinCheck.class) {
                                WinCheck winner=(WinCheck)rst;

        if(winner.isWin) {

        EventDeliver.processExchangeCommand(ExchangeCommands.AfterWin, (Event)rst);
                                }
                        }
                }
        }

        public static void startup() {
                Controller.init();

        System.out.println("Welcome!");

        }
```

```
        public static void outputBoard() {
                //BoardBuffer tmp=Controller.getMap();
                for (int i = 1; i <= Limits.width; i++) {
                        for (int j = 1; j <= Limits.width; j++) {
                                Player p = Controller.check(new Position(i, j));

                                if (p != null) {
                                        if (p.p == Conditions.player_wight) {

        System.out.print("。");
                                        } else if (p.p == Conditions.player_black) {

        System.out.print(". ");

        }else {

        System.out.print("+ ");
                                        }
                                }else {

        System.out.print("+ ");
                                }
                        }

        System.out.println();
                }
```

```
	}

	public static void afterWin(Event e) {

		WinCheck winner=(WinCheck)e;

		switch (winner.winner) {
		case player_black:
	System.out.println("Black Wins!");
			break;
		case player_wight:
	System.out.println("White Wins!");
			break;
		default:
			return;
		}

		isFreshStart=false;

		System.out.println("Do you Want Start Again!?[y/n]");

		Scanner inputScanner=new Scanner(System.in);
		String iString =inputScanner.next();
		switch (iString) {
		case "y":
			startup();
			break;
		case "n":
		default:
	EventDeliver.processCommand("Quit");
			break;
		}
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『VMethods.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```
package view;

import java.util.Scanner;

import board.Conditions;
import control.Controller;
import util.*;
```

```
public abstract class VMethods
implements Processor{

}

class hQuit extends VMethods{

	@Override
	public Object process(Object e) {

	System.out.println("Quiting");
		System.exit(0);
		return e;
	}
}

class hRestart extends VMethods{

	@Override
	public Object process(Object e) {

	System.out.println("Restarting");
		Viewer.isFreshStart=true;
		Viewer.startup();
		return e;
	}
}

class hReset extends VMethods{

	@Override
	public Object process(Object e) {

		Reset tmReset=(Reset)e;

	System.out.println("Please input
a number:");

		int n;
		Scanner
inputScanner=new Scanner(System.in);
		n=inputScanner.nextInt();

		if(n<=0||n>10) {

	System.out.println("Number
Declined");

	tmReset.succ=false;
		}else {
			tmReset.num=n;

	tmReset.succ=true;
		}

		return
EventDeliver.processExchangeComman
d(ExchangeCommands.RESET, tmReset);
	}
}
```

```
class hWrongCommand extends
VMethods{

        @Override
        public Object process(Object e) {
                System.out.println("No
this Command! Check Your Input!");
                return e;
        }
}

class hWrongNum extends VMethods{

        @Override
        public Object process(Object e) {

        System.out.println((Exception)e)
;
                return e;
        }
}

class hHelp extends VMethods{

        @Override
        public Object process(Object e) {
                System.out.println("This
is a Five-In-A-Row Game");

        System.out.println("Help/h:
Show Helps");

        System.out.println("PutChess/p:
Put a Chess");

        System.out.println("Regret/rg:
Regret a Chess");

        System.out.println("Restart/r:
Restart the Game");

        System.out.println("Quit/q: Quit
the Game");
                return e;
        }
}

class hPutChess extends VMethods{

        @Override
        public Object process(Object e) {

                Object isCheck;

        isCheck=(WinCheck)Controller.h
andleEvent(ExchangeCommands.PUTCH
ESS,(Event)e);

        EventDeliver.processExchangeCo
mmand(ExchangeCommands.SHOW,
null);

                return isCheck;
```

```java
	}
}

class hRegret extends VMethods{

	@Override
	public Object process(Object e) {

		Event tmpEvent=(Event)EventDeliver.processExchangeCommand(ExchangeCommands.REGRET, (Event)e);

		if(tmpEvent.succc) {
	System.out.println("Regret One Step:");
	EventDeliver.processExchangeCommand(ExchangeCommands.SHOW, null);
		}else {
	System.out.println("Regret Fail!");
		}

		return tmpEvent;
	}
}
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

『VProcess.java』

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

```java
package view;

import util.Event;
import util.Processor;

public class VProcess{
	public static Object process(Processor p,Object e) {
		return p.process(e);
	}
}
```