实验报告：五子棋扩展功能

执行 Java 程式

```
java -cp bin App
```

1. 项目概述

本次实验在之前棋类游戏的基础上，为五子棋(Gomoku)模式增加了多项扩展功能：

- 15×15大小棋盘，使用十六进制行号和字母列号
- 棋盘上固定位置的障碍物
- 炸弹道具功能
- 演示(Playback)模式

这些扩展功能丰富了游戏体验，增加了游戏的策略性和可玩性。

2. 源代码文件及主要功能

以下是本项目中的主要源代码文件及其功能：

文件名	主要功能
App.java	程序入口点，启动游戏
GameManager.java	游戏管理器，负责游戏流程控制和状态管理
GameView.java	游戏视图，负责棋盘和游戏信息的显示
InputUtils.java	输入工具类，处理用户输入和命令解析，包括playback模式
GameEngine.java	游戏引擎接口，定义游戏基本行为
Gomoku.java	五子棋游戏实现，包括障碍物和炸弹功能
Reversi.java	黑白棋游戏实现
Peace.java	和平模式游戏实现
GameSetup.java	游戏设置，负责初始化
model/Board.java	棋盘模型，可设置不同大小
model/Piece.java	棋子枚举类，包括新增的障碍物和弹坑
model/Player.java	玩家类，包含炸弹数量属性

3. 关键增加代码及设计思路

3.1 棋盘大小扩展与坐标显示

五子棋棋盘从标准的8×8扩展到15×15，并采用十六进制表示行号(1-F)，字母表示列号(A-O)。

设计思路：通过将棋盘大小参数化，为每种游戏模式分配独立的棋盘大小常量。

```
// Gomoku.java
public class Gomoku implements GameEngine {
    private static final int BOARD_SIZE = 15; // Standard Gomoku board
size
    private Board board;

    public Gomoku(Player player, Player player2, Scanner scanner) {
        this.board = new Board(BOARD_SIZE);
        // ...
    }
    // ...
}
```

```
// Board.java
public class Board {
    private int SIZE = 8; // Default size
    private Piece[][] board;

    public Board(int size) {
        this.SIZE = size;
        this.board = new Piece[SIZE][SIZE];
        initializeBoard();
    }
    // ...
}
```

坐标显示：在GameView类中修改显示逻辑，为五子棋模式使用十六进制行号和字母列号：

```
// GameView.java - 列标题显示
if (engine instanceof Gomoku) {
    System.out.print("  ");
    for (char c = 'A'; c < 'A' + board.getSize(); c++) {
        System.out.print(c + " ");
    }
    System.out.println();
} else {
    System.out.println("  A B C D E F G H");
}

// GameView.java - 行号显示
if (engine instanceof Gomoku) {
    String hexRow = Integer.toHexString(row + 1).toUpperCase();
    left.append(hexRow + " ");
} else {
```

```
        left.append((row + 1) + " ");
    }
```

3.2 障碍物功能实现

在五子棋盘上增加了固定位置的障碍物(#)，玩家无法在障碍物位置落子。

设计思路：扩展Piece枚举，增加障碍物类型，并在五子棋初始化时放置固定位置的障碍物。

```
// Piece.java
public enum Piece {
    EMPTY("·"),
    BLACK("○"),
    WHITE("●"),
    CANPLACE("+"),
    BARRIER("#"),
    CRATER("@");
    // ...
}
```

```
// Gomoku.java - 放置障碍物
public Gomoku(Player player, Player player2, Scanner scanner) {
    // ...
    // Place fixed barriers at specified positions: 3F, 8G, 9F, CK
    placeBarrierAtPosition("3F");
    placeBarrierAtPosition("8G");
    placeBarrierAtPosition("9F");
    placeBarrierAtPosition("CK");
}

private void placeBarrierAtPosition(String position) {
    // ... 解析位置并放置障碍物
}
```

3.3 炸弹道具功能

玩家可以使用炸弹移除对方棋子，并在该位置留下弹坑(@)，无法再在此处落子。为了平衡先手优势，黑方初始有2个炸弹，白方有3个炸弹。

设计思路：

1. 在Player类中增加炸弹计数
2. 在Gomoku类中实现炸弹使用逻辑
3. 修改输入处理以支持炸弹命令格式(@1A)

```
// Player.java
public class Player {
```

```
    // ...
    private int bombCount = 0;

    public int getBombCount() {
        return bombCount;
    }

    public void setBombCount(int count) {
        this.bombCount = count;
    }

    public void decrementBombCount() {
        if (bombCount > 0) {
            bombCount--;
        }
    }
    // ...
}
```

```
// Gomoku.java - 初始化炸弹数量
public Gomoku(Player player, Player player2, Scanner scanner) {
    // ...
    // Initialize bomb counts
    if (player.pieceType == Piece.BLACK) {
        player.setBombCount(BLACK_INITIAL_BOMBS); // 2
        player2.setBombCount(WHITE_INITIAL_BOMBS); // 3
    } else {
        player.setBombCount(WHITE_INITIAL_BOMBS);
        player2.setBombCount(BLACK_INITIAL_BOMBS);
    }
    // ...
}

// Gomoku.java - 使用炸弹
public boolean useBomb(int row, int col) {
    Player currentPlayer = getCurrentPlayer();

    // Check if the player has bombs left
    if (currentPlayer.getBombCount() <= 0) {
        return false;
    }

    // Check if position has opponent's piece
    Piece targetPiece = board.getWhatPiece(row, col);
    Piece currentPieceType = currentPlayer.pieceType;
    Piece opponentPieceType = (currentPieceType == Piece.BLACK) ?
Piece.WHITE : Piece.BLACK;

    if (targetPiece != opponentPieceType) {
        return false;
    }
```

```
        // Remove opponent's piece and place a crater
        board.setPiece(row, col, Piece.CRATER);

        // Decrement bomb count
        currentPlayer.decrementBombCount();

        return true;
    }
```

3.4 演示(Playback)模式

增加了演示模式，可以从文件中读取命令序列并自动执行，每条命令间隔1秒。

设计思路：

1. 实现命令队列系统，存储从文件读取的命令
2. 增加对`playback`命令的支持
3. 在命令执行之间添加延时

```
// InputUtils.java
public class InputUtils {
    // ...
    private static Queue<String> commandQueue = new LinkedList<>();
    private static boolean playbackMode = false;

    public static boolean processPlaybackCommand(String filename) {
        try {
            File file = new File(filename);
            Scanner fileScanner = new Scanner(file);

            // Clear any existing commands in the queue
            commandQueue.clear();

            // Read all commands from the file into the queue
            while (fileScanner.hasNextLine()) {
                String command = fileScanner.nextLine().trim();
                if (!command.isEmpty()) {
                    commandQueue.offer(command);
                }
            }

            fileScanner.close();
            playbackMode = true;
            System.out.println("Playback mode: Loaded " +
commandQueue.size() + " commands from " + filename);
            return true;
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + filename);
            return false;
        }
```

```
    }

    // 读取输入 - 支持playback模式
    public static int[] readValidInput(Scanner scanner, GameEngine engine,
Piece piece) {
        while (true) {
            String input;

            // Check if we have commands in the queue (playback mode)
            if (playbackMode && !commandQueue.isEmpty()) {
                input = commandQueue.poll();
                System.out.println("> " + input); // Echo command

                // Delay for 1 second
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    // Ignore
                }

                // Switch back to keyboard if queue is empty
                if (commandQueue.isEmpty()) {
                    playbackMode = false;
                    System.out.println("Playback finished. Switching back
to keyboard input.");
                }
            } else {
                // Regular keyboard input
                input = scanner.nextLine().trim();
            }

            // ... 解析输入 ...
        }
    }
    // ...
}
```

4. 运行效果

4.1 15×15五子棋棋盘

扩展后的五子棋棋盘现在是15×15大小，使用十六进制行号(1-F)和字母列号(A-O)：

```
  A B C D E F G H I J K L M N O
1 · · · · · · · · · · · · · · ·
2 · · · · · · · · · · · · · · ·
3 · · · · · # · · · · · · · · ·  Current Game: 3
4 · · · · · · · · · · · · · · ·  Player [a] ○      Bombs: 2
5 · · · · · · · · · · · · · · ·  Player [b]        Bombs: 3
6 · · · · · · · · · · · · · · ·  Current Round : 1
7 · · · · · · · · · · · · · · ·
```

```
8 · · · · · · # · · · · · · · ·
9 · · · · · # · · · · · · · · ·
A · · · · · · · · · · · · · · ·
B · · · · · · · · · · · · · · ·
C · · · · · · · · · · # · · · ·
D · · · · · · · · · · · · · · ·
E · · · · · · · · · · · · · · ·
F · · · · · · · · · · · · · · ·
```

4.2 障碍物和炸弹功能

棋盘上的障碍物用#表示，玩家无法在此处落子。炸弹使用后会在对方棋子的位置形成弹坑@，同样无法在此处落子：

```
  A B C D E F G H I J K L M N O
1 ○ · · · · · · · · · · · · · ·
2 @ · · · · · · · · · · · · · ·  Current Game: 3
3 · · · · · # · · · · · · · · ·  Player [a]        Bombs: 1
4 · · · · · · · · · · · · · · ·  Player [b] ●      Bombs: 3
5 · · · · · · · · · · · · · · ·  Current Round : 2
```

4.3 演示(Playback)模式

使用playback命令可以从文件中读取命令序列：

```
Please enter your move (1-F,A-O) or use bomb (@1A) / game number (1-3) /
new game (peace / reversi / gomoku) / quit the game (quit) : playback
test3.cmd
Playback mode: Loaded 30 commands from test3.cmd
> 1A
> 2B
> 3C
...
```