



UNIVERSITY OF
ABERDEEN

阿伯丁大学

自然与计算科学学院 计算科学系

2023 - 2024

编程作业 - 单独评估（无团队合作）

标题JC2002 - Java 编程

这项作业占课程总分的 30%。

学习成果

成功完成本任务后，学生将证明自己能够

- 编写和运行基本的 Java 应用程序
- 判断 Java 应用程序的结构
- 为 Java 应用程序编写正确的功能代码
- 在计算机程序中分析和解决简单问题
- 应用面向对象编程概念在 Java 程序中创建组件

有关剽窃和串通的信息：源代码和您的报告可能会被提交到 *Codio* 中进行抄袭检查。在开始作业前，请参阅 *MyAberdeen* 上的幻灯片，了解更多有关避免抄袭的信息。请***注意，与其他学生一起提交类似作业会被视为串通作业。***如果遇到困难，请联系老师寻求帮助；***不要请其他学生、朋友或任何其他人为完成作业！***此外，请阅读学校提供的以下信息：

<https://www.abdn.ac.uk/sls/online-resources/avoiding-plagiarism/>

引言

本课程作业的目标是向我们展示您在 Java 面向对象编程以及使用 JUnit 和 Maven 方面的技能。在作业的 Codio 框中，您将找到模板代码，您需要对其进行扩展以完成作业。您的目标是让您的代码通过 JUnit 测试，编写这些测试是为了确认应用程序完成了它应该做的事情。您只能在源文件中编写代码。您可以随意阅读测试代码，但除了取消对测试的注释外，**不要编辑测试代码**。原始测试将在评分过程中使用，您对测试的任何编辑或修改都将意味着您的代码可能无法通过测试。

游戏规则很简单。游戏区域是一个 2D 网格，玩家（你）从一个角落开始，三个怪物从其他三个角落开始。每轮游戏中，玩家可以向上、向下、向左或向右移动一步。怪物也会随机移动一步。如果玩家试图移动到怪物占据的位置，则表示玩家正在攻击怪物。如果攻击成功，怪物会损失 50 点健康值。相反，如果怪物试图移动到玩家占据的位置，则怪物会攻击玩家。如果攻击成功，玩家将损失 20 点健康值。

开始时，玩家和怪物各有 100 点健康值。当健康值为零时，角色就会死亡。你在游戏中的目标是在怪物杀死你之前杀死它们。下图 1 展示了这款游戏。

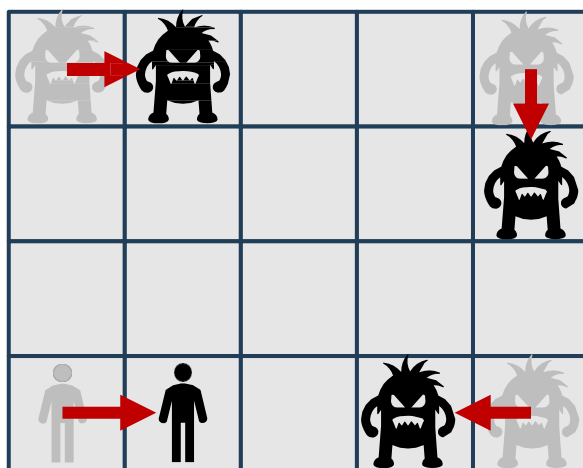


图 1.游戏示意图。

指导和要求

该评估包括编码任务和概述上述步骤的最终书面任务。我们将为您提供一些类的模板代码，以帮助您入门。您的 Java 代码和报告必须符合下面描述的结构，并包含各部分概述的所需内容。每个子任务都有自己的分数分配。您**必须**

提供一份书面报告及相应的代码，其中包含对所开展的工作进行全面批判和反思的所有不同部分。

虽然我们不会在作业中对 Git 的使用进行评分，但我们强烈建议您在 Git 仓库中完成这项工作。这样做的好处是，你可以回滚编辑，或在分支中完成工作，然后在完成任务的过程中将其合并到主分支。


为了测试进度，项目模板包含 JUnit 5 测试。这些测试最初都被注释掉了，这样就不会出现编译错误，因为这些测试期望您在测试之前必须实现某些方法。您可以在实现过程中取消注释测试。

您可以通过 "我的阿伯丁" (MyAberdeen) 中的课程作业链接访问作业模板。这里有两个链接，分别是 AI 组和 CS 组，请根据自己的学习计划选择正确的链接。您可以编写 `mvn clean test` 命令来测试简洁模板，结果应该会有如下输出：

```
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ assignment ---
[INFO]
[INFO] -----
[INFO]  T E S T S
[INFO] -----
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  2.323 s
[INFO] Finished at: 2023-10-20T11:45:19Z
[INFO] -----
codio@derbyecology-summerelectra:~/workspace/assignment$
```

完成子任务后，可以取消对相关测试的注释。例如，完成任务 1.2 后，应像这样修改文件

"CharacterTest.java":



```
/**
 * Task 1.2
 */
/*@Test
public void checkSayName()
{
    class TestImplementation extends GameCharacter {
        public TestImplementation(String name) {
            super(name);
        }
        @Override
        public void hurtCharacter (GameCharacter character) {}
        @Override
        public boolean successfulDefense() {return true;}
    }
    try {
        assertEquals ("milan", new TestImplementation("milan").sayName());
    } catch (Exception | Error e) {
        fail(ANSI_WHITE_BACKGROUND +ANSI_BLACK+"Make sure the constructor of Gan
    }
}*/

/**
 * Task 1.2
 */
@Test
public void checkSayName()
{
    class TestImplementation extends GameCharacter {
        public TestImplementation(String name) {
            super(name);
        }
        @Override
        public void hurtCharacter (GameCharacter character) {}
        @Override
        public boolean successfulDefense() {return true;}
    }
    try {
        assertEquals ("milan", new TestImplementation("milan").sayName());
    } catch (Exception | Error e) {
        fail(ANSI_WHITE_BACKGROUND +ANSI_BLACK+"Make sure the constructor of G
```

如果取消注释后测试失败，说明任务没有正确完成。但要**注意的是，测试通过并不能保证任务成功**，因为测试只涵盖了部分可能出现的错误！

在后期阶段，可以使用 `mvn package` 命令生成一个 JAR 文件，然后使用命令从**目标**文件夹运行该文件：

`java -jar assignment-1.0-SNAPSHOT.jar <您的参数> <您的参数>`

参数将包括游戏区域的高度和宽度，稍后会有说明。您也可以使用 Maven 运行主方法：

`mvn 编译 exec:java -Dexec.mainClass="uoa.assignment.game.RunGame" -Dexec.args="arg1 arg2"`

每项任务的详细说明如下。要完成这些任务，请按照说明进行操作，并确保使用提供的模板。报告中必须清楚地描述运行最终项目的所有必要步骤。有关提交说明，请参阅后面的章节。

任务 1：实现字符的基本方法（10 分）

1.1) 使用 MyAberdeen 中的作业链接，在 Codio 中找到名为 **assignment** 的项目。在 `uoa.assignment.character` 软件包和 `uoa.assignment.game` 软件包下，您会看到一个 Maven 项目的模板代码，其中包含大量 java 文件。

1.2) 修改 `GameCharacter` 类的构造函数，使字段名由代表角色名称的参数值填充，例如 `new GameCharacter ("Player")`。在 `GameCharacter` 类中，完成以字符串形式返回角色名称的公共方法 `sayName()`。

1.3) 修改 `GameCharacter` 类，以便在创建实例时将私有整数变量 `health` 初始化为默认值 100。完成变量 `health` 的 setter 和 getter 方法 `getHealth()` 和 `setHealth()`。

1.4) 在文件 **"Monster.java "中的 "Monster "类中**，实现 **"Monster.java "中的抽象方法**。
`GameCharacter` 类如下：

`hurtCharacter(GameCharacter 角色)` 在角色没有成功防御的情况下（即当角色的 `successfulDefence()` 返回 `false` 时），移除 50 点健康值。

successfulDefence() 如果怪物成功自卫，则返回 *true*，否则返回 *false*。结果是随机决定的，成功和失败的几率各占一半。提示：您可以使用 `java.util.Random` 包中的 `nextInt()` 方法来决定随机结果；详情请查看 `Random` 文档。

1.5) 在文件 **"Player.java "**中的 `Player` 类中，实现从 **"Player.java "**继承的抽象方法。

`GameCharacter` 类如下：

`hurtCharacter(GameCharacter 角色)` 在角色没有成功防御的情况下（即当角色的 `successfulDefence()` 返回 `false` 时），移除 20 点健康值。

`successfulDefence()` 返回玩家成功防守的可能性。结果是随机选择的，成功防御的概率为 30%。

任务 2：绘制游戏区地图（10 分）

2.1) 修改 `Map` 类中的变量 `characters`，使其能容纳四个 `GameCharacter` 类型的对象。修改 `Map` 类的构造函数，使其根据传入构造函数的输入参数（整数 `height` 和 `width`）初始化二维数组布局。参数 `height` 指定主数组的长度，参数 `width` 指定嵌套数组的长度（即结果是一个由数组组成的二维数组）。

2.2) 在 `Map` 类中创建一个初始化数组（`initialiseArray()`）的私有方法，该方法将在每次创建 `Map` 类的新实例时用句号字符"."填充二维数组。在同一个类中，完成公共方法 `printLayout()`，将二维布局数组打印到控制台。

`printLayout()` 方法的输出示例，假设是 3x6 二维数组：

```
.....  
.....  
.....
```

2.3) 每次创建 `Map` 实例时，创建一个玩家类实例和三个怪物类实例。创建玩家和怪物时，使用类 `GameCharacter` 中的变量 `row` 和 `column` 来存储数组索引，描述每个游戏角色在任务 2.1 中实现的二维数组布局所代表的地图上的当前位置。玩家应位于右下角，怪物应分别位于其余三个角上。将角色添加到任务 2.1 中创建的角色数组中（玩家应存储在 0 号位置，怪物应存储在右上角 1 号位置，怪物应存储在左下角 2 号位置，怪物应存储在左上角 3 号位置）。

2.4) 修改代码，将怪物和玩家的位置存储在布局数组中（使用符号 "%" 表示怪物， "*" 表示玩家）。地图初始化后 `printLayout()` 的输出应如下所示（假设是 3x3 的二维数组）：

```
%.%  
...  
%.*
```

任务 3：实现基本游戏功能（15 分）

3.1) 修改 **Game** 类的构造函数，使其为游戏实例化和初始化**地图**对象，并将**布局**打印到控制台。

在 **RunGame** 的方法 **main()** 中创建一个新对象 **Game**。地图的**高度**和**宽度**参数应取自 **Game** 对象的 **main()** 方法的字符串参数（第一个参数是高度，第二个参数是宽度；使用 **java** 或 **mvn** 编译执行命令启动程序时，这些参数将作为参数传递）。

3.2) 扩展 **RunGame** 类中的 **while** 循环，以便在创建**游戏**对象后扫描用户控制台输入，直到满足结束游戏的条件（稍后将解释该条件，现阶段可以使用虚拟条件）。每次迭代开始时，控制台将打印该轮的编号，例如 "第 1 轮"。

3.3) 在 **GameLogic** 类中，完成公共静态方法 **moveCharacter(String input, Map gameMap, GameCharacter character)**。如果**输入**值不是 "up"、"down"、"left" 或 "right"，该方法应在控制台打印 "仅使用关键字 up、down、left、right"。

3.4) 在 **GameLogic** 类中，创建 **moveRight(GameCharacter character, Map gameMap)**、**moveLeft(GameCharacter character, Map gameMap)**、**moveUp(GameCharacter character, Map gameMap)** 和 **moveDown(GameCharacter character, Map gameMap)** 等私有静态方法，这些方法将相应地移动角色，并更新保存在对象实例和**布局**数组中的角色位置。在任务 3.3 中实现的 **moveCharacter** 方法中使用这些方法可根据用户输入移动角色。

在第一回合中对**玩家**角色执行 **moveUp** 方法后打印**布局**的预期输出如下：

```
%.%  
..*  
%..
```

任务 4：实现游戏控制和基本逻辑（15 分）

4.1) 更新任务 3.4 中实施的方法，以便在字符试图向墙壁移动时打印以下内容："您不能向右移动。您将失去一个移动。"、"您不能向左移动。您将失去一个移动位置"，"您不能向上移动。"你不能向下走" "You can't go down. You lose a move. 你输了一步棋"

4.2) 完成 "怪物" 类的公共方法 **decideMove()**，该方法将随机返回以下结果之一字符串 "上"、"下"、"左"、"右"（每个选择的概率相等）。

4.3) 在 **Game** 类中，完成带有布尔返回值的公共方法 **nextRound(String input)**，其中参数 **input** 是用户输入（"up"、"down"、"left"、"right"），如果 **RunGame** 的 **main()** 方法中的循环应继续

，则该方法返回 *false*。使用 **GameLogic** 类中的 **moveCharacter()** 方法完成 **nextRound()** 方法，使玩家首先使用控制台输入移动，然后所有活着的怪物（健康值大于 0 的怪物）根据任务 4.2 中实施的 **decideMove()** 方法的决定自动移动。每个角色执行的移动都应打印在控制台中。

在 **Game** 类中，实现一个公共方法 **getMap()**，返回当前版本的游戏地图。假设游戏逻辑已正确实现，控制台中的示例输出如下：

```
%.%  
...  
%.*第 1
```

轮向上

玩家向上移动 怪物 1 向左移动 怪物 2 向右移动 怪物 3 向下移动

```
.%.  
%.*  
%.%
```

第二轮

4.4) 更新你的代码，使一个角色不能在有另一个角色的情况下移动到某个地方。当一个怪物试图在另一个怪物占据的地方移动时，打印 "怪物已经在那里，所以它不能移动"。如果怪物无法移动，它将停留在原来的位置。

更新后，输出示例如下：

```
%.%  
...  
%.*
```

第一轮

玩家向上移动 怪物1向左移动 怪物 2向上移动 怪物3向下移动
怪物已经在那里了，所以不能移动

```
%%.  
%.*  
...
```

第二轮

任务 5：完成代码以实现剩余的游戏逻辑（15 分）

5.1) 更新 `RunGame` 类中的 `main()` 方法，以便在每轮游戏中，即调用 `nextRound(String input)` 时（见任务 4.3），程序会打印每个角色的健康状况。更新后，输出示例如下：

%. %

```
...
%. *
```

第一轮

玩家向上移动 怪物1向左移动 怪物

2向上移动 怪物3向下移动

怪物已经在那里了，所以不能移动

健康玩家100

健康怪物1: 100

健康怪物2: 100

健康怪物3: 100

```
%%.
```

```
%. *
```

```
...
```

第二轮

5.2) 更新代码，当玩家角色试图移动到有怪物的位置时，将通过调用**玩家**对象的 **hurtCharacter(GameCharacter character)** 方法来尝试攻击。攻击是否成功取决于怪物对象的 **successfulDefense()** 方法的结果。如果防御失败，怪物的健康就会受损。攻击成功后，怪物应损失 50 点健康值。用适当的信息更新控制台。

5.3) 更新代码，当怪物角色试图移动到玩家所在的位置时，将通过调用**怪物**对象的 **hurtCharacter(GameCharacter 角色)**方法进行攻击。攻击是否成功取决于玩家对象的 **successfulDefense()** 方法的结果。如果防御失败，玩家的健康就会受损。攻击成功后，玩家将损失 20 点健康值。用适当的信息更新控制台。

5.4) 更新代码，当怪物受到攻击后生命值为零时，地图上怪物的字符"**%**"会变为 "**x**"，表示怪物死亡。死亡的怪物无法移动，也无法再次受到攻击。如果玩家试图攻击已死亡的怪物，则在控制台中打印 "角色已死亡"。

5.5) 更新**游戏类**中 **nextRound(String input)** 的代码，如果没有活着的怪物，方法返回 *true*。

在这种情况下，会通过打印一条信息通知玩家他们赢了

"你赢了！"。同样，如果玩家的健康值为 0，**gameOver** 变量将被设置为 *true*，并在控制台中打印信息 "YOU HAVE DIED！"。

任务 6：报告（35 分）

您的报告应描述程序的整体设计，以及程序开发过程中面临的挑战。请使用清晰的描述性文字、支持性代码片段和屏幕截图来描述和证明重现和运行您的作品所需的每一个步骤。使用截图时，请确保截图清晰可读。请慎用代码片段和屏幕截图；报告不应该是屏幕截图的集合！

如果您的工作依赖于任何外部资源，请附上参考文献。如果您使用了开放源代码，您必须指出该代码的来源（即使来源是在线教程或博客），并详细说明您在任务中对该代码所做的任何修改。您应在代码和报告中提及这一点。否则，相关（子）任务的分数将为零分。

评分标准

作业将按照以下标准，使用总分 100 分的最高分和每个子任务的最高分进行评分：

- 源代码的清晰度和可读性，包括代码注释。
- 程序的技术正确性（即程序能通过预定义的 JUnit 测试，游戏逻辑能正常运行）。
- 可重复性。其他学生根据您的报告和代码重复你的工作有多容易？
- 报告的质量，包括语言、结构、清晰度、简洁性、参考文献。

提交说明

您应提交 PDF 版本的报告和代码。PDF 文件的名称应为 "JC2002_Assignment_<您的姓>_<您的名>_<您的学号>"。例如，"JC2002_Assignment_Smith_John_4568985.pdf"，其中 4568985 是您的学号。您应使用 Codio 提交代码和报告。请将报告的 PDF 文件保存在 "**pom.xml**" 所在的 **assignment** 目录中。

确保代码在 Codio 中运行是您的责任。如果您在其他平台上完成工作，然后将其转移到 Codio，您需要确保程序也能在 Codio 中运行，而且运行效果符合预期。

如对本次评估的任何方面有任何疑问，请向其中一位教师提出：

Marco Palomino marco.palomino@abdn.ac.uk (人工智能
小组) Jari Korhonen, jari.korhonen@abdn.ac.uk (计算
机科学与技术小组)

附录 A：游戏运行示例

```
$ java -jar assignment-1.0-SNAPSHOT.jar 3, 3
```

```
%.%  
...  
%.*
```

第一轮

玩家向上移动 怪物1向上移动

你不能往上走。你失去了一步棋 怪物 2 向左移动

你不能向左移动。您将失去一步棋 怪物 3 正在向下移动

健康玩家100

健康	怪物1	100
健康	怪物2	100
健康	怪物3	100

```
..%  
%.*  
%..
```

第二轮比

赛

球员升级

失误怪物 1 成功防御了玩家的攻击 怪物 1 向下移动

失误玩家成功防御了怪物 1 的攻击 怪物 2 正在向上移动

怪物已经在那里，所以无法移动 怪物3正在向右移动

健康玩家100

健康怪物1： 100
健康怪物2： 100
健康怪物3： 100

```
..%  
.%*  
%..
```

第三轮上

场

球员升级

失误怪物 1 成功防御了玩家的攻击 怪物 1 正在向上移动
你不能上升。你输了一步棋

怪物 2 正在向上移动 怪物 3 正

在向左移动

怪物已经在那里了，所以不能移动

健康玩家100

健康怪物1： 100

健康怪物2： 100

健康怪物3： 100

..%

%%*

...

第 4 轮

左侧

玩家向左移动

命中玩家成功攻击怪物 3 怪物 1 向左移动

怪物 2 向左移动

你不能向左移动。怪物 3 向左移动，您将失去一步棋

怪物已经在那里了，所以不能移动

健康玩家100

健康怪物1： 100

健康怪物2： 100

健康怪物3： 50

.%.

%%*

...

第 5 轮

左侧

玩家向左移动

没打中怪物 3 成功防御了玩家怪物 1 向左移动的攻击

怪物 2 向下移动 怪物 3 向上移

动

健康玩家100

健康怪物1： 100

健康怪物2： 100

健康怪物3： 50

%%.

..*

%..

第六轮上

场

玩家向上移动 怪物1向左移动

你不能向左移动。您将失去一步棋 怪物 2 正在向右

移动

怪物 3 正在升级

你不能上升。你输了一步棋

健康玩家100

健康怪物1: 100

健康怪物2: 100

健康怪物3: 50

%%*

...

.%.

第 7 轮

左侧

玩家向左移动

没打中怪物 3 成功防御了玩家怪物 1 向右移动的攻击

怪物已经在那里，所以无法移动 怪物 2 正在向下移

动

你不能向下移动。您将失去一步棋 怪物 3 正在向上

移动

你不能上升。你输了一步棋

健康玩家100

健康怪物1: 100

健康怪物2: 100

健康怪物3: 50

%%*

...

.%.

第 8 轮

左侧

玩家向左移动

失误怪物 3 成功防御了玩家怪物 1 的攻击，正在向下移动

怪物 2 向右移动 怪物 3 向下移

动

健康玩家100

健康怪物1: 100

健康怪物2: 100

健康怪物3: 50

..*
%%.
..%

第 9 轮
左侧

玩家向左移动 怪物1向左移动

你不能向左移动。您将失去一步棋 怪物 2 正在向右移动

有一堵墙。怪物 3 向右移动，您将失去一步棋

健康玩家100
健康怪物1： 100
健康怪物2： 100
健康怪物3： 50

. * .
%. %
.. %

还剩第 10

轮

玩家向左移动 怪物 1 向下移动 怪物 2 向左移动 怪物 3 向左移动

健康玩家100
健康怪物1： 100
健康怪物2： 100
健康怪物3： 50

* ..

. %.

%%.

第 11 轮结

束

玩家向下移动 怪物1向左移动

你不能向左移动。您将失去一步棋 怪物 2 正在向右移动

怪物 3 向右移动

健康玩家100
健康怪物1： 100
健康怪物2： 100
健康怪物3： 50

. . .
* . %

%.%

第 12 轮结
束

玩家向下移动

命中玩家成功攻击怪物 1 怪物 1 向下移动

你不能向下移动。您将失去一步棋 怪物 2 正在向上移动

怪物已经在那里，所以无法移动 怪物 3 向左移动

健康玩家100

健康怪物1: 50

健康怪物2: 100

健康怪物3: 50

...

*%. .

%. %

第 13 轮结

束

玩家向下移动

失误怪物 1 成功防御了玩家的攻击 怪物 1 向下移动

你不能倒下。你失去了一步怪兽 2 正向右移动

有一堵墙。怪物 3 向左移动，您失去一步棋

命中怪物3成功攻击玩家

健康 球员80

健康 怪物1 50

健康 怪物2 100

健康 怪物3 50

...

*%. .

%. %

第 14 轮右

玩家向右移动

命中玩家成功攻击怪物 3 怪物 1 正在向上移动

没打中玩家成功防御怪物 1 的攻击 怪物 2 向右移动

有一堵墙。你输了一步棋

健康 球员80

健康 怪物1 50

健康 怪物2 100

健康 怪物3 0

...

*x. .

%.%

第 15 轮结

束

玩家向下移动

命中玩家成功攻击怪物 1 怪物 2 向下移动

你不能倒下。你输了一步棋

健康	球员	80	
健康	怪物1		0
健康	怪物2		100
健康	怪物3		0

...
*x.
x.%

第 16 轮

玩家向上移动 怪物2向上移动

健康	球员	80	
健康	怪物1		0
健康	怪物2		100
健康	怪物3		0

*..
.x%
x..

第 17 轮右

玩家向右移动 怪物2向下移动

健康	球员	80	
健康	怪物1		0
健康	怪物2		100
健康	怪物3		0

.*.
.x.
x.%

第 18 轮右

玩家向右移动 怪物2向右移动

有一堵墙。你输了一步棋

健康玩家80

健康	怪物1	0
健康	怪物2	100
健康	怪物3	0

..*
.x.
x.%

第 19 轮结束

玩家向下移动 怪物2向上移动
HIT!怪物2成功攻击玩家

健康	球员60	
健康	怪物1	0
健康	怪物2	100
健康	怪物3	0

...
.x*
x.%

第 20 轮下降

玩家向下移动
命中玩家成功攻击怪物 2 怪物 2 向左移动

健康	球员60	
健康	怪物1	0
健康	怪物2	50
健康	怪物3	0

...
.x*
x%.

第 21 轮结束

玩家向下移动 怪物2向上移动
怪物已经在那里了，所以不能移动

健康	球员60	
健康	怪物1	0
健康	怪物2	50
健康	怪物3	0

\dots
 $\cdot X \cdot$
 $X \circledast \star$

还剩第 22

轮

玩家向左移动

没打中怪物 2 成功防御了玩家的攻击 怪物 2 向左移动

怪物已经在那里了，所以不能移动

健康	球员	60	
健康	怪物1	0	
健康	怪物2	50	
健康	怪物3	0	

...

.x.

x%*

第 23 轮左

侧

玩家向左移动

失误怪物 2 成功防御了玩家的攻击 怪物 2 向下移动

你不能倒下。你输了一步棋

健康	球员	60	
健康	怪物1	0	
健康	怪物2	50	
健康	怪物3	0	

...

.x.

x%*

第 24 轮左

侧

玩家向左移动

命中玩家成功攻击怪物 2

健康	球员	60	
健康	怪物1	0	
健康	怪物2	0	
健康	怪物3	0	

你赢了

...

.x.

xx*