**《软件测试》**

**实验报告七 ——综合测试2**

**姓 名： 黄勇斌 学 号：2019112321**

**院 系： 计算机与信息学院 专 业： 计算机科学与技术**

**实 验 室： 实验日期：2022/5/15**

**总评成绩： 审阅教师：**

目录

[**《软件测试》** 1](#_Toc101792344)

[**实验报告六 ——综合测试1** 1](#_Toc101792345)

[一、实验目的： 3](#_Toc101792346)

[二、实验环境： 3](#_Toc101792347)

[三、实验要求： 3](#_Toc101792348)

[四、实验步骤与内容 3](#_Toc101792349)

[五、结论分析与体会 6](#_Toc101792350)

[六、仓库地址 6](#_Toc101792351)

### 一、实验目的：

1. 学习在一定规模实际项目中综合测试方法
2. 学习研读开源代码的技术
3. 熟悉项目构建工具gradle
4. 熟悉静态代码分析工具

### 二、实验环境：

Eclipse2020,JUnit

### 三、实验要求：

题目1

Jpacman有一些重要的行为没有经过测试，单元测试测试：Ghost(幽灵)的移动行为。

查看nl.tudelft.jpacman.npc.ghost包下的Clyde类，认真阅读该类的Javadoc描述，理解这类Ghost的行为方式。

查看Clyde类的 Optional nextAiMove()”方法，考虑测试用例以确保Clyde按预期行为工作。

实例化Clyde对象

在test模块java/nl/tudelft/jpacman包中的GhostMapParser类，它是提供快速生成地图的辅助工具类。比如，你提供下面这样三行字符串，将自动生成一个3X12大小，且含有一个玩家和一个Clyde幽灵的地图。玩家和Clyde之间间隔8个Square。 {“############”, “ #P C#”, “############”}, GhostMapParser的parseMap()方法接收上述字符串数组，并返回一个Level对象。而这个Level对象管理了Board、Player、Ghost等几乎游戏涉及到的所有模型对象。

比如，如果你的测试涉及到Player，需要利用Player的PlayerFactory生成Player对象。然后再调用Level的RegisterPlayer来注册这个新创建的Player对象，并调用Player的setDirection方法来设置一个初始的行走方向。 除了创建Player，通常还需要Ghost。四种Ghost对象在地图解析时，就已经创建好，可以通过调用Navigation的findUnitInBoard()方法来获取.

实现这个ClydeTest测试类时，需要构建游戏运行的所有所需对象。这些对象包括PacManSprites（用于角色显示）、PlayerFactory（用于构造Player）、GhostFactory（提供给LevelFactory），BoardFactory（游戏场景）、GhostMapParser（地图解析）。

思考以上对象构造过程，建议利用JUnit5的@BeforeEach或@BeforeAll来为不同的测试方法创建统一的所需对象集。 在ClydeTest类中至少测试4个情况下的行为； 为你的每个函数和类增加javadoc注解。

为了测试生成Clyde魔鬼，可以增加一个地图解析器，Clyde魔鬼地图解析器

题目2

创建InkyTest测试类

查看Inky幽灵类，可与Clyde幽灵类作对比。特别注意一下它的nextAiMove()方法有什么不同。

注意GhostMapParser的addSquare函数对Inky幽灵的处理，请补充完整。

实现对Inky测试至少5种情况下的行为。

每个函数和类增加javadoc注解

### 四、实验步骤与内容

1. **在实验六的代码基础上， 完成下面题目1，提交到自己的代码仓库**
2. **在完成题目1的基础上完成题目2，题目2选做（学有余力者做）**
3. **编写测试报告，提交到雨课堂“软件测试实验七”**

**测试用例：**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试样例标号 | 输入地图 | 预测输出 | 实际输出 |
| 1 |  | Direction.EAST | Direction.EAST |
| 2 |  | Direction.EAST | Direction.EAST |
| 3 |  | Direction.WEST | Direction.WEST |
| 4 |  | !direction.isPresent() | !direction.isPresent() |
| 5 |  | Direction.SOUTH | Direction.SOUTH |
| 6 |  | Direction.NORTH | Direction.NORTH |

1. **代码实现：**

|  |
| --- |
| **测试代码：**  **Clyde类：**  package nl.tudelft.jpacman.npc.ghost;  import java.util.EnumMap;  import java.util.List;  import java.util.Map;  import java.util.Optional;  import nl.tudelft.jpacman.board.Direction;  import nl.tudelft.jpacman.board.Square;  import nl.tudelft.jpacman.board.Unit;  import nl.tudelft.jpacman.level.Player;  import nl.tudelft.jpacman.npc.Ghost;  import nl.tudelft.jpacman.sprite.Sprite;  /\*\*  \* <p>  \* An implementation of the classic Pac-Man ghost Clyde.  \* </p>  \* <p>  \* Pokey needs a new nickname because out of all the ghosts,  \* Clyde is the least likely to "C'lyde" with Pac-Man. Clyde is always the last  \* ghost out of the regenerator, and the loner of the gang, usually off doing  \* his own thing when not patrolling the bottom-left corner of the maze. His  \* behavior is very random, so while he's not likely to be following you in hot  \* pursuit with the other ghosts, he is a little less predictable, and still a  \* danger.  \* </p>  \* <p>  \* <b>AI:</b> Clyde has two basic AIs, one for when he's far from Pac-Man, and  \* one for when he is near to Pac-Man.  \* When Clyde is far away from Pac-Man (beyond eight grid spaces),  \* Clyde behaves very much like Blinky, trying to move to Pac-Man's exact  \* location. However, when Clyde gets within eight grid spaces of Pac-Man, he  \* automatically changes his behavior and runs away.  \* </p>  \* <p>  \* Source: http://strategywiki.org/wiki/Pac-Man/Getting\_Started  \* </p>  \*  \* @author Jeroen Roosen  \*/  public class Clyde extends Ghost {  /\*\*  \* The amount of cells Clyde wants to stay away from Pac Man.  \*/  private static final int SHYNESS = 8;  /\*\*  \* The variation in intervals, this makes the ghosts look more dynamic and  \* less predictable.  \*/  private static final int INTERVAL\_VARIATION = 50;  /\*\*  \* The base movement interval.  \*/  private static final int MOVE\_INTERVAL = 250;  /\*\*  \* A map of opposite directions.  \*/  private static final Map<Direction, Direction> OPPOSITES = new EnumMap<>(Direction.class);  static {  OPPOSITES.put(Direction.NORTH, Direction.SOUTH);  OPPOSITES.put(Direction.SOUTH, Direction.NORTH);  OPPOSITES.put(Direction.WEST, Direction.EAST);  OPPOSITES.put(Direction.EAST, Direction.WEST);  }  /\*\*  \* Creates a new "Clyde", a.k.a. "Pokey".  \*  \* @param spriteMap The sprites for this ghost.  \*/  public Clyde(Map<Direction, Sprite> spriteMap) {  super(spriteMap, MOVE\_INTERVAL, INTERVAL\_VARIATION);  }  /\*\*  \* {@inheritDoc}  \*  \* <p>  \* Clyde has two basic AIs, one for when he's far from Pac-Man, and one for  \* when he is near to Pac-Man.  \* When Clyde is far away from Pac-Man (beyond eight grid spaces),  \* Clyde behaves very much like Blinky, trying to move to Pac-Man's exact  \* location. However, when Clyde gets within eight grid spaces of Pac-Man,  \* he automatically changes his behavior and runs away  \* </p>  \*/  @Override  public Optional<Direction> nextAiMove() {  assert hasSquare();  Unit nearest = Navigation.findNearest(Player.class, getSquare());  if (nearest == null) {  return Optional.empty();  }  assert nearest.hasSquare();  Square target = nearest.getSquare();  List<Direction> path = Navigation.shortestPath(getSquare(), target, this);    if (path != null && !path.isEmpty()) {  Direction direction = path.get(0);  if (path.size() <= SHYNESS) {  return Optional.ofNullable(OPPOSITES.get(direction));  }  return Optional.of(direction);  }  return Optional.empty();  }  }**ClydeTest类：**  package nl.tudelft.jpacman.clyde;  import nl.tudelft.jpacman.GhostMapParser;  import nl.tudelft.jpacman.board.BoardFactory;  import nl.tudelft.jpacman.board.Direction;  import nl.tudelft.jpacman.level.Level;  import nl.tudelft.jpacman.level.LevelFactory;  import nl.tudelft.jpacman.level.Player;  import nl.tudelft.jpacman.level.PlayerFactory;  import nl.tudelft.jpacman.npc.ghost.Clyde;  import nl.tudelft.jpacman.npc.ghost.GhostFactory;  import nl.tudelft.jpacman.npc.ghost.Navigation;  import nl.tudelft.jpacman.points.PointCalculator;  import nl.tudelft.jpacman.sprite.PacManSprites;  import org.junit.jupiter.api.BeforeAll;  import org.junit.jupiter.api.BeforeEach;  import org.junit.jupiter.api.Test;  import org.junit.jupiter.api.TestInstance;  import org.junit.jupiter.params.ParameterizedTest;  import java.io.IOException;  import java.util.Arrays;  import java.util.Optional;  import static org.mockito.ArgumentMatchers.isA;  import static org.mockito.Mockito.mock;  @TestInstance(TestInstance.Lifecycle.PER\_CLASS)  public class ClydeTest {  private Level level;  private GhostMapParser ghostMapParser;  private PlayerFactory playerFactory;  private Player player;  /\*\*  \* 初始化一些测试所需要的对象  \* @throws IOException 读取文件错误时抛出的异常  \*/  @BeforeEach  public void setUp (){  PacManSprites sprites = new PacManSprites();  LevelFactory levelFactory = new LevelFactory(  sprites,  new GhostFactory(sprites),  mock(PointCalculator.class));  ghostMapParser = new GhostMapParser(levelFactory, new BoardFactory(sprites), new GhostFactory(sprites));  //获取游戏管理者对象  //根据文件生成地图  String[] map = {  "############",  "#P C#",  "############"  };  level = ghostMapParser.parseMap(Arrays.asList(map));  playerFactory = new PlayerFactory(sprites);  player = playerFactory.createPacMan();  //游戏角色初始化朝北面  player.setDirection(Direction.NORTH);  level.registerPlayer(player);  }  private void reParser(String[] map) {  level = ghostMapParser.parseMap(Arrays.asList(map));  //游戏角色初始化朝北面  player.setDirection(Direction.NORTH);  level.registerPlayer(player);  }  /\*\*  \* 拿到实例化后的 Clyde 对象  \*/  @Test  public void test() {  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  System.out.println(clyde);  }  /\*\*  \* 间隔等于8  \*/  @Test  public void testEQ8() {  String[] map = {  "###########",  "#P C#",  "###########"  };  reParser(map);  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  Optional<Direction> direction = clyde.nextAiMove();  //预期行为时向东远离  assert direction.isPresent() && direction.get().equals(Direction.EAST);  }  /\*\*  \* 间隔小于8的情况  \*/  @Test  public void testLT8() {  String[] map = {  "##########",  "#P C#",  "##########"  };  reParser(map);  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  Optional<Direction> direction = clyde.nextAiMove();  //预期行为时向东远离  assert direction.isPresent() && direction.get().equals(Direction.EAST);  }  /\*\*  \* 间隔大于8 的情况  \*/  @Test  public void testGT8 () {  String[] map = {  "##############",  "#P C#",  "##############"  };  reParser(map);  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  Optional<Direction> direction = clyde.nextAiMove();  //预期行为时向西靠近  assert direction.isPresent() && direction.get().equals(Direction.WEST);  }  /\*\*  \* 不可到达的情况  \*/  @Test  public void unAccessible() {  String[] map = {  "##############",  "#P##########C#",  "##############"  };  reParser(map);  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  Optional<Direction> direction = clyde.nextAiMove();  //不可达，返回空的方向  assert !direction.isPresent();  }  /\*\*  \* 路线曲折大于8的情况  \*/  @Test  public void routeNotLineGT8() {  String[] map = {  "##############",  "#P # # # C#",  "## # ## #",  "## # #",  "## # ## #",  "## # #",  "## # ## #",  "## # #",  "##############"  };  reParser(map);  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  Optional<Direction> direction = clyde.nextAiMove();  //向南方向  assert direction.isPresent() && direction.get().equals(Direction.SOUTH);  }  /\*\*  \*路线曲折小于8  \*/  @Test  public void routeLineLT8() {  String[] map = {  "##############",  "#P # # # #",  "## #C ## #",  "## # #",  "##############"  };  reParser(map);  Clyde clyde = Navigation.findUnitInBoard(Clyde.class, level.getBoard());  Optional<Direction> direction = clyde.nextAiMove();  //向南方向的反方向即北方向  assert direction.isPresent() && direction.get().equals(Direction.NORTH);  }  }  测试结果：    Jacco |

### 五、结论分析与体会

感觉这一次的实验还是挺难的，我也是在别人的帮助下搞定的，希望后面可以继续提升自己的能力

### 六、仓库地址

<https://github.com/hyb1041739742/Software-Testing>