Link to repository: https://github.com/ks-moss/472-2023-G3.git

Task 1 Coverage before testing

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	3% (4/110)	1% (10/624)	1% (28/2274)
tudelft	3% (4/110)	1% (10/624)	1% (28/2274)
jpacman	3% (4/110)	1% (10/624)	1% (28/2274)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	0% (0/26)	0% (0/156)	0% (0/690)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	0% (0/12)	0% (0/90)	0% (0/238)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationExcep	0% (0/1)	0% (0/2)	0% (0/4)

Is the coverage good enough? No, it's not good enough. Let's increase the coverage.

Task 2

PlayerTest.java

```java
package nl.tudelft.jpacman.level;

import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

no usages  new *
public class PlayerTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    1 usage
    private Player ThePlayer = Factory.createPacMan();

    no usages  new *
    @Test
    void testIsAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( expected: true); }
}
```

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4/26)	6% (10/156)	3% (26/700)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationExcep	0% (0/1)	0% (0/2)	0% (0/4)

Task 2.1

src/main/java/nl/tudelft/jpacman/npc/ghost/GhostFactory.java

First Method

GhostFactoryTest.java

```java
package nl.tudelft.jpacman.npc.ghost;

import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

no usages  new *
public class GhostFactoryTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);

    no usages  new *
    @Test
    void testGhostFactory(){
        System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
        //System.out.println("Value of createClyde() : " + tempGhost.createClyde());
        //System.out.println("Value of createPinky() : " + tempGhost.createPinky());
    }
}
```

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	23% (26/110)	11% (74/624)	9% (230/2318)
tudelft	23% (26/110)	11% (74/624)	9% (230/2318)
jpacman	23% (26/110)	11% (74/624)	9% (230/2318)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4/26)	6% (10/156)	3% (26/700)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	46% (42/90)	54% (142/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationExcep	0% (0/1)	0% (0/2)	0% (0/4)

Second Method

GhostFactoryTest.java

```java
package nl.tudelft.jpacman.npc.ghost;

import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

no usages   new *
public class GhostFactoryTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    2 usages
    GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);

    no usages   new *
    @Test
    void testGhostFactory(){
        System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
        System.out.println("Value of createClyde() : " + tempGhost.createClyde());
        //System.out.println("Value of createPinky() : " + tempGhost.createPinky());
    }
}
```

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	25% (28/110)	12% (80/624)	10% (250/2318)
tudelft	25% (28/110)	12% (80/624)	10% (250/2318)
jpacman	25% (28/110)	12% (80/624)	10% (250/2318)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4/26)	6% (10/156)	3% (26/700)
npc	50% (10/20)	19% (18/94)	11% (54/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	46% (42/90)	54% (142/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationExce	0% (0/1)	0% (0/2)	0% (0/4)

Third Method

GhostFactoryTest.java

```java
package nl.tudelft.jpacman.npc.ghost;

import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

no usages   new *
public class GhostFactoryTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    3 usages
    GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);

    no usages   new *
    @Test
    void testGhostFactory(){
        System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
        System.out.println("Value of createClyde() : " + tempGhost.createClyde());
        System.out.println("Value of createPinky() : " + tempGhost.createPinky());
    }
}
```

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	27% (30/110)	13% (86/624)	11% (258/2318)
tudelft	27% (30/110)	13% (86/624)	11% (258/2318)
jpacman	27% (30/110)	13% (86/624)	11% (258/2318)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4/26)	6% (10/156)	3% (26/700)
npc	60% (12/20)	25% (24/94)	12% (62/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	46% (42/90)	54% (142/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Forth Method

GhostFactoryTest.java

```java
package nl.tudelft.jpacman.npc.ghost;

import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

no usages   new *
public class GhostFactoryTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    4 usages
    GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);

    no usages   new *
    @Test
    void testGhostFactory(){
        System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
        System.out.println("Value of createClyde() : " + tempGhost.createClyde());
        System.out.println("Value of createPinky() : " + tempGhost.createPinky());
        System.out.println("Value of createInky() : " + tempGhost.createInky());
    }
}
```

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	29% (32/110)	14% (92/624)	11% (266/2318)
tudelft	29% (32/110)	14% (92/624)	11% (266/2318)
jpacman	29% (32/110)	14% (92/624)	11% (266/2318)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4/26)	6% (10/156)	3% (26/700)
npc	70% (14/20)	31% (30/94)	14% (70/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	46% (42/90)	54% (142/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Task 3

Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?

The results obtained from the JaCoCo coverage tool differ from those previously obtained through IntelliJ in a prior task. The coverage results within IntelliJ are contingent upon various factors such as the type of coverage tool used, its configuration, the build process utilized, and the extent of test coverage within the codebase.

Did you find helpful the source code visualization from JaCoCo on uncovered branches?

Affirmative, I discovered that the JaCoCo coverage tool provides a visualization of the source code, specifically highlighting any uncovered branches. This information can be readily accessed as demonstrated in the following illustration:

jpacman

jpacman

Element
default
nl.tudelft.jpacman
nl.tudelft.jpacman.board
nl.tudelft.jpacman.game
nl.tudelft.jpacman.level
nl.tudelft.jpacman.npc
nl.tudelft.jpacman.npc.ghost
nl.tudelft.jpacman.points
nl.tudelft.jpacman.sprite
nl.tudelft.jpacman.ui

Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?

I found the JaCoCo coverage report to be preferable due to its convenient accessibility of information regarding uncovered branches, as well as its well-structured format. The report also employs a clear and effective color-coding scheme, utilizing red and white to display the coverage percentage as demonstrated below:

jpacman Sessions

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.game		87%		60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.level		67%		57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.sprite		87%		59%	29	70	10	113	4	38	0	5
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
Total	1,211 of 4,694	74%	293 of 637	54%	292	590	228	1,039	50	268	6	47

Created with JaCoCo 0.8.3.201901230119