**Bloque 1: NIVEL 2 CAJA BLANCA: IWT42 Grupo 6**

***Nombres:*  
Adrián Infante Pérez**

**Manuel Guerra Solís**

INDICE

[Nivel 2: Caja Blanca 3](#_Toc182865040)

[1. Presentación grafos de ejecución los métodos, ciclomática y los caminos independientes encontrados. 3](#_Toc182865041)

[Shot 3](#_Toc182865042)

[Alien 4](#_Toc182865043)

[Bomb 5](#_Toc182865044)

[Player 6](#_Toc182865045)

[Board 8](#_Toc182865046)

[2. Conjunto de casos de prueba por cada método probado, entradas a probar y salidas esperadas 15](#_Toc182865047)

[3. Implementaciones en Junit 5 de los casos de prueba definidos 15](#_Toc182865048)

[4. Los resultados obtenidos a la finalización de esta fase, casos de prueba NO han sido superados y casos de prueba que hayan sido superados exitosamente. 15](#_Toc182865049)

# Nivel 2: Caja Blanca

## Presentación grafos de ejecución los métodos, ciclomática y los caminos independientes encontrados.

Adjhfadnfkeafkajhdfkajhfksajdhf

### Shot

* private void initShot(int x, int y)

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 1.

Caminos Independientes:

1. src/main/resources/images/shot.png → new ImageIcon(shotImg) → setImage(ii.getImage()) → int H\_SPACE = 6 → setX(x + H\_SPACE) → int V\_SPACE = 1 → setY(y - V\_SPACE)

### Alien

* public Alien(int x, int y)

Diagrama

Descripción generada automáticamente con confianza media

Complejidad ciclomática = 5

Caminos indepentienes:

1. x > Commons.BOARD\_WIDTH → this.x = Commons.BOARD\_WIDTH → y > Commons.BOARD\_HEIGHT → this.y = Commons.BOARD\_HEIGHT → bomb = new Bomb(x, y) → src/main/resources/images/alien.png → new ImageIcon(alienImg) → setImage(ii.getImage())
2. x < 0 → this.x = 0 → y < 0 → this.y = 0 → bomb = new Bomb(x, y) → src/main/resources/images/alien.png → new ImageIcon(alienImg) → setImage(ii.getImage())
3. x < 0 → this.x = 0 → y >= 0 → else → this.x = x → this.y = y → bomb = new Bomb(x, y) → src/main/resources/images/alien.png → new ImageIcon(alienImg) → setImage(ii.getImage())
4. x dentro del rango → y > Commons.BOARD\_HEIGHT → this.y = Commons.BOARD\_HEIGHT → bomb = new Bomb(x, y) → src/main/resources/images/alien.png → new ImageIcon(alienImg) → setImage(ii.getImage())
5. x dentro del rango → y >= 0 → else → this.x = x → this.y = y → bomb = new Bomb(x, y) → src/main/resources/images/alien.png → new ImageIcon(alienImg) → setImage(ii.getImage())

### Bomb

* public Bomb(int x, int y)cc

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 2

Caminos independientes:

1. setDestroyed(true) → x <= Commons.BOARD\_WIDTH && y <= Commons.BOARD\_HEIGHT → this.x += x → this.y += y → src/main/resources/images/bomb.png → new ImageIcon(bombImg) → setImage(ii.getImage())
2. setDestroyed(true) → else → this.x = Commons.BOARD\_WIDTH → this.y = Commons.BOARD\_HEIGHT → src/main/resources/images/bomb.png → new ImageIcon(bombImg) → setImage(ii.getImage())

### Player

* public void act()

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 4

Caminos independientes:

1. x += dx → x < 2 → x < Commons.BOARD\_WIDTH - 2 \* width
2. x += dx → x >= 2 → x = 2 → x < Commons.BOARD\_WIDTH - 2 \* width
3. x += dx → x >= 2 → x = 2 → x >= Commons.BOARD\_WIDTH - 2 \* width → x = Commons.BOARD\_WIDTH - 2 \* width
4. x += dx → x < 2 → x >= Commons.BOARD\_WIDTH - 2 \* width → x = Commons.BOARD\_WIDTH - 2 \* width

* public void keyPressed(KeyEvent e)///public void keyReleased(KeyEvent e)

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 3

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 3

Caminos encontrados:

1. key = e.getKeyCode → key == KeyEvent.VK\_LEFT → dx = 2
2. key = e.getKeyCode → key == KeyEvent.VK\_RIGHT → dx = 2
3. key = e.getKeyCode → no acción
4. key = e.getKeyCode → key == KeyEvent.VK\_LEFT → dx = 0
5. key = e.getKeyCode → key == KeyEvent.VK\_RIGHT → dx = 0
6. key = e.getKeyCode → no acción

### Board

* private void gameInit()

Diagrama

Descripción generada automáticamente

Compleejidad ciclomática = 3

Caminos independientes:

1. aliens = new ArrayList → i = 0 → j = 0 → crear Alien → add(alien) → j = 1 → crear Alien → add(alien) → ... → j = 5 → crear Alien → add(alien) → i = 1 → repetir hasta i = 3 → player = new Player → shot = new Shot
2. aliens = new ArrayList → i = 0 → no entra en j → i = 1 → no entra en j → repetir hasta i = 3 → player = new Player → shot = new Shot
3. aliens = new ArrayList → player = new Player → shot = new Shot

* private void update()

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 2

Caminos independientes:

1. deaths == Commons.CHANCE → inGame = false → timer.stop() → message = "Game won!" → this.player.act() → update\_shots() → update\_aliens() → update\_bomb()
2. deaths != Commons.CHANCE → this.player.act() → update\_shots() → update\_aliens() → update\_bomb()

* private void update\_shots()

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 4

Caminos independientes:

1. this.shot.isVisible() → shotX = this.shot.getX() → shotY = this.shot.getY() → para cada alien → alien.isVisible() → shotX >= alienX → shotX <= alienX + ALIEN\_WIDTH → shotY >= alienY → shotY <= alienY + ALIEN\_HEIGHT → new ImageIcon(explImg) → alien.setImage(ii.getImage()) → alien.setDying(true) → deaths-- → this.shot.die() → y = this.shot.getY() → y -= 4 → y < 0 → this.shot.die()
2. this.shot.isVisible() → shotX = this.shot.getX() → shotY = this.shot.getY() → para cada alien → alien.isVisible() → no se ejecuta el bloque interno → y = this.shot.getY() → y -= 4 → y < 0 → this.shot.die()
3. this.shot.isVisible() → no se ejecuta el bloque interno → fin
4. this.shot.isVisible() → shotX = this.shot.getX() → shotY = this.shot.getY() → para cada alien → no impacta → y = this.shot.getY() → y -= 4 → y >= 0 → this.shot.setY(y)

* private void update\_bomb()

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 6

Caminos independientes encontrados:

1. generator.nextInt(15) → shot == Commons.CHANCE → alien.isVisible() → bomb.isDestroyed() → bomb.setDestroyed(false) → bomb.setX(alien.getX()) → bomb.setY(alien.getY()) → bombX = bomb.getX() → bombY = bomb.getY() → playerX = this.player.getX() → playerY = this.player.getY() → this.player.isVisible() → !bomb.isDestroyed() → bombX >= playerX → bombX <= playerX + Commons.PLAYER\_WIDTH → bombY >= playerY → bombY <= playerY + Commons.PLAYER\_HEIGHT → new ImageIcon(explImg) → this.player.setImage(ii.getImage()) → this.player.setDying(false) → bomb.setDestroyed(true)
2. generator.nextInt(15) → shot != Commons.CHANCE o alien.isVisible() o bomb.isDestroyed() es falso → bombX = bomb.getX() → bombY = bomb.getY() → playerX = this.player.getX() → playerY = this.player.getY() → this.player.isVisible() → !bomb.isDestroyed() → bombX >= playerX → bombX <= playerX + Commons.PLAYER\_WIDTH → bombY >= playerY → bombY <= playerY + Commons.PLAYER\_HEIGHT → no se ejecuta el bloque de colisión
3. !bomb.isDestroyed() → bomb.setY(bomb.getY() + 1) → bomb.getY() >= Commons.GROUND - Commons.BOMB\_HEIGHT → bomb.setDestroyed(false)
4. !bomb.isDestroyed() → bomb.setY(bomb.getY() + 1) → bomb.getY() < Commons.GROUND - Commons.BOMB\_HEIGHT

* private void update\_aliens()

Diagrama

Descripción generada automáticamente

Complejidad ciclomática = 8

Caminos independientes encontrados:

1. x <= BOARD\_WIDTH - BORDER\_RIGHT → direction != -1 → direction = 0 → Iterator<Alien> i1 → setY(a2.getY() + GO\_DOWN) → Iterator<Alien> i1 → repite para cada alien → Iterator<Alien> it → if (alien.isVisible()) → int y = alien.getY() → y > GROUND - ALIEN\_HEIGHT → inGame = false → message = "Invasion!" → alien.act(direction)
2. x <= BOARD\_WIDTH - BORDER\_RIGHT → direction == -1 → no se actualiza direction → Iterator<Alien> i1 → setY(a2.getY() + GO\_DOWN) → Iterator<Alien> i1 → repite para cada alien → Iterator<Alien> it → if (alien.isVisible()) → int y = alien.getY() → y > GROUND - ALIEN\_HEIGHT → inGame = false → message = "Invasion!" → alien.act(direction)
3. x <= BORDER\_LEFT → direction != 1 → direction = 1 → Iterator<Alien> i2 → setX(a.getY() + GO\_DOWN) → Iterator<Alien> i2 → repite para cada alien → Iterator<Alien> it → if (alien.isVisible()) → int y = alien.getY() → y > GROUND - ALIEN\_HEIGHT → inGame = false → message = "Invasion!" → alien.act(direction)
4. x <= BORDER\_LEFT → direction == 1 → no se actualiza direction → Iterator<Alien> i2 → setX(a.getY() + GO\_DOWN) → Iterator<Alien> i2 → repite para cada alien → Iterator<Alien> it → if (alien.isVisible()) → int y = alien.getY() → y > GROUND - ALIEN\_HEIGHT → inGame = false → message = "Invasion!" → alien.act(direction)
5. x > BOARD\_WIDTH - BORDER\_RIGHT y x > BORDER\_LEFT → direction no cambia → Iterator<Alien> it → if (alien.isVisible()) → int y = alien.getY() → y > GROUND - ALIEN\_HEIGHT → inGame = false → message = "Invasion!" → alien.act(direction)
6. x > BOARD\_WIDTH - BORDER\_RIGHT y x > BORDER\_LEFT → direction no cambia → Iterator<Alien> it → if (alien.isVisible()) no se ejecuta → no se mueve
7. x <= BOARD\_WIDTH - BORDER\_RIGHT → direction = 0 → Iterator<Alien> i1 → setY(a2.getY() + GO\_DOWN) → Iterator<Alien> i1 → repite para cada alien → Iterator<Alien> it → if (alien.isVisible()) no se ejecuta → no se mueve
8. x <= BORDER\_LEFT → direction = 1 → Iterator<Alien> i2 → setX(a.getY() + GO\_DOWN) → Iterator<Alien> i2 → repite para cada alien → Iterator<Alien> it → if (alien.isVisible()) no se ejecuta → no se mueve

* AAAAAAAa

## Conjunto de casos de prueba por cada método probado, entradas a probar y salidas esperadas

## Implementaciones en Junit 5 de los casos de prueba definidos

void testUpdateWhenConditionsMet() throws InterruptedException {

    board.getTimer().stop();

    board.setDeaths(Commons.CHANCE);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertFalse(board.isInGame());

    assertEquals("Game Won!", board.getMessage());

  }

  @Test

  void testUpdateWhenConditionsNotMet() throws InterruptedException {

    board.getTimer().stop();

    board.setDeaths(Commons.CHANCE - 1);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertTrue(board.isInGame());

    assertNotEquals("Game Won!", board.getMessage());

  }

  @Test

  void testUpdateShotsShotNotVisible() throws InterruptedException {

    board.getTimer().stop();

    Shot shot = new Shot();

    shot.die(); // Shot is not visible

    board.setShot(shot);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertFalse(shot.isVisible());

  }

  @Test

  void testUpdateShotsVisibleAlienVisibleHit() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    Shot shot = new Shot();

    shot.setX(alien.getX());

    shot.setY(alien.getY());

    board.setShot(shot);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertTrue(alien.isDying());

    assertFalse(shot.isVisible());

  }

  @Test

  void testUpdateShotsVisibleAlienVisibleMiss() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    Player player = board.getPlayer();

    Shot shot = new Shot();

    alien.setX(Commons.BORDER\_LEFT);

    alien.setY(Commons.BOARD\_HEIGHT);

    shot.setX(player.getX());

    shot.setY(player.getY());

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertTrue(shot.isVisible());

    assertFalse(alien.isDying());

  }

  @Test

  void testUpdateAliensMoveDownAtRightBoundary() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    alien.setX(Commons.BOARD\_WIDTH - Commons.BORDER\_RIGHT); // At right boundary

    alien.setY(50);

    board.setDirection(1); // Moving right

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertEquals(50 + Commons.GO\_DOWN, alien.getY()); // Alien moves downward

    assertEquals(0, board.getDirection()); // Direction reset

  }

  @Test

  void testUpdateAliensMoveDownAtLeftBoundary() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    alien.setX(Commons.BORDER\_LEFT);

    alien.setY(50);

    board.setDirection(-1); // Moving left

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertEquals(50 + Commons.GO\_DOWN, alien.getY());

    assertEquals(1, board.getDirection());

  }

  @Test

  void testUpdateAliensGameOver() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    alien.setX(100);

    alien.setY(Commons.GROUND - Commons.ALIEN\_HEIGHT);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertFalse(board.isInGame()); // Game ends

    assertEquals("Invasion!", board.getMessage());

  }

  @Test

  void testUpdateAliensMoveRight() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    alien.setX(50); // Well within the board

    alien.setY(50);

    board.setDirection(1); // Moving right

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertEquals(52, alien.getX()); // Alien moves right

    assertEquals(1, board.getDirection()); // Direction remains unchanged

    assertTrue(board.isInGame()); // Game is still ongoing

  }

  @Test

  void testUpdateAliensMoveLeft() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    alien.setX(Commons.BORDER\_LEFT + 2);

    alien.setY(50);

    board.setDirection(-1); // Moving left

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertEquals(Commons.BORDER\_LEFT, alien.getX());

    assertEquals(-1, board.getDirection());

    assertTrue(board.isInGame());

  }

  @Test

  void testUpdateBombCreatedForVisibleAlien() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    Alien.Bomb bomb = alien.getBomb();

    bomb.setDestroyed(true);

    board.setPlayer(new Player());

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertFalse(bomb.isDestroyed());

    assertEquals(alien.getX(), bomb.getX());

    assertEquals(alien.getY(), bomb.getY());

  }

  @Test

  void testUpdateBombHitsPlayer() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    Alien.Bomb bomb = alien.getBomb();

    bomb.setDestroyed(false);

    Player player = board.getPlayer();

    player.setX(50);

    player.setY(50);

    bomb.setX(50);

    bomb.setY(50);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertTrue(player.isDying());

    assertTrue(bomb.isDestroyed());

  }

  @Test

  void testUpdateBombReachesGround() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    Alien.Bomb bomb = alien.getBomb();

    bomb.setDestroyed(false);

    bomb.setY(Commons.GROUND - Commons.BOMB\_HEIGHT);

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertTrue(bomb.isDestroyed());

  }

  @Test

  void testUpdateBombContinuesFalling() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    Alien.Bomb bomb = alien.getBomb();

    bomb.setDestroyed(false); // Bomb is active

    bomb.setY(50); // Bomb is mid-air

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertEquals(51, bomb.getY()); // Bomb moves one step downward

    assertFalse(bomb.isDestroyed()); // Bomb is still active

  }

  @Test

  void testUpdateBombNotCreatedForInvisibleAlien() throws InterruptedException {

    board.getTimer().stop();

    Alien alien = board.getAliens().get(0);

    alien.die();

    Alien.Bomb bomb = alien.getBomb();

    bomb.setDestroyed(true); // Bomb not created initially

    board.getTimer().start();

    Thread.sleep(Commons.DELAY);

    assertTrue(bomb.isDestroyed()); // Bomb remains destroyed

  }

}

## Los resultados obtenidos a la finalización de esta fase, casos de prueba NO han sido superados y casos de prueba que hayan sido superados exitosamente.

### Métodos si pasan test

* void testUpdateWhenConditionsNotMet()
* void testUpdateShotsShotNotVisible()
* void testUpdateShotsVisibleAlienVisibleMiss()
* void testUpdateBombNotCreatedForInvisibleAlien()

### Métodos NO pasan test

* void testUpdateWhenConditionsMet()
  + Se esperaba false, y ha llegado true
* void testUpdateShotsVisibleAlienVisibleHit()
  + Se esperaba false, y ha llegado true
* void testUpdateAliensMoveDownAtRightBoundary()
  + Se esperaba 65 y ha llegado 50
* void testUpdateAliensMoveDownAtLeftBoundary()
  + S Se esperaba 65 y ha llegado 50
* void testUpdateAliensGameOver()
  + Se esperaba false, y ha llegado true
* void testUpdateAliensMoveRight()
  + Se esperaba 52, y ha llegado 50
* void testUpdateAliensMoveLeft()
  + Se esperaba 5 y ha llegado 7
* void testUpdateBombCreatedForVisibleAlien()
  + Se esperaba false, y ha llegado true
* void testUpdateBombHitsPlayer()
  + Se esperaba true y ha llegado false
* void testUpdateBombReachesGround()
  + Se esperaba true y ha llegado false
* void testUpdateBombContinuesFalling()
  + Se esperaba 51 y ha llegado 50