# GameUtil : WarriorAttack(Warrior warrior, AttackType attackType, Direction firstDirection, Direction secondDirection = Direction.INVALID\_DIRECTION)

Paths:

1-2-3(T)-4-5(T)-6-9(T)-10(T)-11(F)-14-16-17(T)-18-19  
1-2-3(F)-12  
1-2-3(T)-4-5(F)-7(T)-8-9(F)-19  
1-2-3(T)-4-5(F)-7(T)-8-9(T)-10(T)-11(T)-12  
1-2-3(T)-4-5(F)-7(T)-8-9(T)-10(T)-11(F)-14-16-17(F)-19

Selected path: 1-2-3(T)-4-5(T)-6-9(T)-10(T)-11(F)-14-16-17(T)-18-19

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Interpretation** |
| 1 | <warrior,attackType,firstDirection,secondDirection> |  |
| 2 | warriorPosition = warrior.Position |  |
| 3(T) | board.IsPositionInsideBoundary(warriorPosition, firstDirection, secondDirection) | board.IsPositionInsideBoundary(warriorPosition, firstDirection, secondDirection) |
| 4 | attackPosition = board.GetPositionInADirection(warriorPosition, firstDirection, secondDirection) opponent = findWarriorAtARoom(attackPosition.X, attackPosition.Y); | attackPosition = board.GetPositionInADirection(warriorPosition, firstDirection, secondDirection) |
| 5(T) | warrior is MeleeWarrior | warrior is MeleeWarrior |
| 6 | warrior.MeleePower = GetAddedPowerWithinRange(warrior, 0.5, PowerType.MELEE\_POWER) warrior.MagicPower = GetAddedPowerWithinRange(warrior, 0.125, PowerType.MAGIC\_POWER) |  |
| 9(T) | opponent != null | opponent != null |
| 10(T) | attackType == AttackType.MeleeAttack | attackType == AttackType.MeleeAttack |
| 11(F) | secondDirection != Direction.INVALID\_DIRECTION | secondDirection != Direction.INVALID\_DIRECTION |
| 14 | meleeDamage = opponent.MeleePower - (opponent.DefensePercentage \* opponent.MeleePower)  opponent.HitPoints -= meleeDamage |  |
| 16 | opponent.DefensePercentage = GetAddedPowerWithinRange(opponent, 0.25, PowerType.DEFENCE\_PERCENTAGE) |  |
| 17(T) | opponent.HitPoints <= 0 | opponent.HitPoints <= 0 |
| 18 | killAnOpponent(warrior, opponent) |  |
| 19 | return true |  |

Path predicate expression

|  |
| --- |
| board.IsPositionInsideBoundary(warriorPosition, firstDirection, secondDirection) ≡ true -- (1) |
| warrior is MeleeWarrior ≡ true(2) |
| opponent != null ≡ true (3) |
| attackType == AttackType.MeleeAttack ≡ true (4) |
| secondDirection != Direction.INVALID\_DIRECTION ≡ false (5) |
| opponent.HitPoints <= 0 ≡ true (6) |

Test Data

|  |
| --- |
| warrior = new MeleeWarrior() |
| warrior.Position = new Position {2, 2} |
| attackType = AttackType.MeleeAttack |
| firstDirection = Direction.EAST |
| secondDirection = Direction.INVALID\_DIRECTION |
| attackPosition = new Position {2,3} |
| opponent = new MeleeWarrior() |
| opponent.HitPoints = 0.001 |

Test Case before control flow testing: AWarriorShouldBeAbleToAttackAnOpponent. This test cases checks whether an opponent still exists in the attackPosition after the attack.

Test Case after control flow testing: AWarriorShouldBeAbleToKillAnOpponent. This test case checks whether an opponent still exists after the HitPoints gets below 0.

# Weapon : constructor

Paths:

1-2-3(T)-4

1-2-3(F)-4

Selected Path: 1-2-3(T)-4

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Interpretation** |
| **1** | <weaponType> |  |
| **2** | this.weaponType = weaponType | this.weaponType = weaponType |
| **3(T)** | this.weaponType == WeaponType.Staff | this.weaponType == WeaponType.Staff |
| **4** | this.Power = Util.random.Next(3,7) | this.Power >= 3 and < 7 |
| **5** | this.Power = Util.random.Next(2,5) | this.Power >= 2 and < 5 |

**Path Predicate Expression**

|  |
| --- |
| this.weaponType == WeaponType.Staff === true |

**Input Vector (Test Data)**

|  |
| --- |
| weaponType = weaponType.Staff |

No change in ShouldSetPowerBetweenThreeAndSixForStaffWeaponTypeWhenConstructed and ShouldSetPowerBetweenTwoAndFourForSwordWeaponTypeWhenConstructed test cases.

# Warrior: addWeapon(Weapon weapon)

Paths:

1-2(F)-4-5(F)-7(F)-9

1-2(T)-3

1-2(F)-4-5(T)-6-9

1-2(F)-4-5(F)-7(T)-8-9

Selected Path: 1-2(F)-4-5(F)-7(T)-8-9

**Predicate Interpretation Table**

|  |  |  |
| --- | --- | --- |
| **Node** | **Description** | **Interpretation** |
| **1** | <weapon> |  |
| **2(F)** | this.weapon != null |  |
| **4** | this.weapon = weapon |  |
| **5(F)** | weapon.WeaponType == WeaponType.Sword | weapon.WeaponType == WeaponType.Sword |
| **7(T)** | weapon.WeaponType == WeaponType.Staff | weapon.WeaponType == WeaponType.Staff |
| **8** | this.MagicPower += this.weapon.Power | this.MagicPower = this.MagicPower + this.weapon.Power |
| **9** | return void |  |

**Path Predicate Expression**

|  |
| --- |
| weapon.WeaponType == WeaponType.Sword === false  weapon.WeaponType == WeaponType.Staff === true |

**Input Vector (Test Data)**

|  |
| --- |
| weapon.WeaponType = WeaponType.Staff |

No change in ShouldIncreaseMeleePowerWhenASwordIsPicked test case.

# Board:

GetPositionInADirection(Position position, Direction firstDirection, Direction secondDirection = Direction.INVALID\_DIRECTION)

Paths:

1-2-3(T)-4-5

1-2-3(F)-4-5

Selected Path: 1-2-3(T)-4-5

|  |  |  |
| --- | --- | --- |
| Node | Description | Interpretation |
| 1 | <position, firstDirection, secondDirection> |  |
| 2 | deltaPosition = GetMovementDelta(firstDirection)  newPosition = new Position { X = position.X + deltaPosition.X, Y = position.Y + deltaPosition.Y } |  |
| 3(T) | secondDirection != Direction.INVALID\_DIRECTION | secondDirection != Direction.INVALID\_DIRECTION |
| 4 | newDelta = GetMovementDelta(secondDirection)  newPosition.X += newDelta.X  newPosition.Y += newDelta.Y |  |
| 5 | return newPosition |  |

**Path Predicate Expression**

|  |
| --- |
| secondDirection != Direction.INVALID\_DIECTION === true |

**Input Vector (Test Data)**

|  |
| --- |
| secondDirection = Directio.EAST |

No change in ShouldGetAPositionInADirectionInTheBoundaryPosition and ShouldGetAPositionInADirectionOutsideTheBoundaryPosition test cases.