White-Box Testing Evidence

Unit testing. All dynamic, functional tests.

Test Class	Test Name	Test ID	Description	Result
CharacterTest	charactersWithSamePositionShouldCollide	1.1.1	Creates one character and checks it collides with itself.	PASS
	touchingCharactersShouldCollide	1.1.2	Creates two characters than are just touching to ensure they collide. This is the most likely situation to occur in game.	PASS
	nonTouchingCharactersShouldNotCollide	1.1.3	Creates two characters that should not collide and checks this is the case.	PASS
	getCenterOnCharacterWithPositivePosition	1.2.1	Tests the calculation for getCenter(), on a character in the top right quadrant. It assumes the character sprite is 32x32, as it should be.	PASS
	getCenterOnCharacterWithNegativePosition	1.2.2	Negative test as no character should ever have a negative position in the x or y direction. Again testing the calculation for getCenter().	PASS
	getDirectionInTopLeftQuadrant	1.3.1	Testing the getDirection method for each possible quadrant. All of which are possible as the direction is a bearing relative to the characters center.	PASS
	getDirectionInBottomLeftQuadrant	1.3.2		PASS
	getDirectionInBottomRightQuadrant	1.3.3		PASS
	getDirectionInTopRightQuadrant	1.3.4		PASS

	charactersTakeSpecifiedDamage	1.4	Calling takeDamage() on a character and testing if the expected hit points are lost from the player.	PASS
	getDirNormVectorToNegativePosition	1.5.1	Testing that getDirNormVector calculates the correct normalized vector from the characters center to a positive coordinate.	PASS
	getDirNormVectorToPositivePosition	1.5.2	Testing that getDirNormVector calculates the correct normalized vector from the characters center to a negative coordinate.	PASS
PlayerTest	playerPositionResetsWhenRespawned	2.1	Check that the original position of the player is the same as the position after moving it, then respawning it.	PASS
	playerDoesDamageToZombieWhenAtMaxRange	2.2.1	Creates a player and a zombie where the zombie is the maximum range away from the player in the direction that the player is facing. Then the player attacks the zombie and we check that the health decreases.	PASS
	playerDoesDamageToZombieWhenInRange	2.2.2	The same as above but the zombie's distance from the player is less than the maximum range.	PASS
	playerDoesNoDamageToZombieWhenOutOfRange	2.2.3	The same as above but the zombie's distance from the player is greater than the maximum range. In this case the player should not do damage to the player.	PASS
	playerTypesHaveDifferentHealth	2.3.1	Save the health of a nerdy student in a variable then respawn the player as a sporty student and check that they have a different amount of hit points.	PASS
	playerTypesHaveDifferentSpeed	2.3.2	Save the speed of a nerdy student in a variable then respawn the player as a sport student and check that they have different speed.	PASS
ZombieTest	zombieDoesDamageToPlayerWhenAtMaxRange	3.1.1	The same as 2.1.1 but switch player and zombie positions and checking that the zombie does damage to the player.	PASS

	zombieDoesDamageToPlayerWhenInRange	3.1.2	The same as 2.1.2 but switch player and zombie positions and checking that the zombie does damage to the player.	PASS
	zombieDoesNoDamageToPlayeWhenOutOfRange	3.1.3	The same as 2.1.3 but switch player and zombie positions and checking that the zombie does no damage to the player.	PASS
	zombieCannotAttackBeforeCooldownComplete	3.2.1	The zombie tries to attack the player twice in rapid succession. The player should only take damage from the first attack.	PASS
	zombieCanAttackAfterCooldownComplete	3.2.2	The zombie tries to attack the player twice but with a pause longer than the zombies cooldown time between the attacks. The player should take damage from both attacks.	PASS
PowerUpTest	powerUpHealthAddsHPToPlayer	4.1	Reduces players health then activates a health power up and checks that the players health goes up by the amount specified by Constant.HEALUP.	PASS
	powerUpSpeedIncreasesPlayersSpeed	4.2.1	Compares the players speed before and after activating a speed power up to make to sure the speed increases by the amount specified by Constant.SPEEDUP.	PASS
	powerUpSpeedDeactivatesAfter10s	4.2.2	Compares the players speed before activating it and 11 seconds after it has been activated to make sure the speed is the same as the original speed.	PASS
	powerUpSpeedDoesNotDeactivateBefore 10s	4.2.3	Compares the player speed before activating and 9s after activating to make sure the speed is still different.	PASS
	powerUpSpeedDeactivateMethodResestsPlayerSpeed	4.2.4	Tests that the speed power ups effect can be cancelled at anytime by calling deactivate0 manually.	PASS
	playerCannotPickUpFarAwayPowerUp	4.3.1	Checks the player can’t pick up a power up that is out of reach (must be overlapping) by using the overlapsPlayer() method of the PowerUp class.	PASS

	playerCanPickUpClosePowerUp	4.3.2	Checks the player can pick up a power up that is in reach (must be overlapping) by using the overlapsPlayer() method of the PowerUp class.	PASS
	powerUpImmunityStopsThePlayerTakingDamage	4.4.1	Activates an immunity power up and calls takeDamage on the player. Checks that the players health before and after takeDamage remains the same.	PASS
	powerUpImmunityDeactivatesAfter5s	4.4.2	Activates an immunity power up and calls takeDamage before and after 5 seconds. Checks the the player only lost hit points from takeDamage called after 5 seconds.	PASS
	powerUpImmunityDeactivateMethodCancelsImmunity	4.4.3	Activates an immunity power up and calls takeDamage on the player before and after calling deactivate on the power up. Checks the player only lost hit points from the takeDamage called after deactivate.	PASS