



| Testing gears | Input | expected value |
|---|---|---|
| Instantiate a headgear with negative effects | Headgear(.TypeOfEffect.NEGATIVE) | {Ability.CONSTITUTION: x}(-3<=x<0) |
| | headgear.use() | |
| Instantiate a headgear with positive effects | Headgear(.TypeOfEffect.POSITIVE) | {Ability.CONSTITUTION: x}(0<x<=5) |
| | headgear.use() | |
| Instantiate a small belt with negative effects | Belt(.TypeOfEffect.NEGATIVE, TypeOfBelt.SMALL) | {Ability.CONSTITUTION: -1},{Ability.CHARISMA, -1} |
| | belt.use(); | |
| | getUit(); | 1 |
| Instantiate a small belt with positive effects | Belt(.TypeOfEffect.POSITIVE, TypeOfBelt.SMALL) | {Ability.CONSTITUTION: 1},{Ability.CHARISMA, 1} |
| | belt.use(); | |
| | getUit(); | 1 |
| Instantiate a medium belt with negative effects | Belt(.TypeOfEffect.NEGATIVE, TypeOfBelt.MEDIUM) | {Ability.CONSTITUTION: -2},{Ability.CHARISMA, -2} |
| | belt.use(); | |
| | getUit(); | 2 |
| Instantiate a medium belt with positive effects | Belt(.TypeOfEffect.POSITIVE, TypeOfBelt.MEDIUM) | {Ability.CONSTITUTION: 2},{Ability.CHARISMA, 2} |
| | belt.use(); | |
| | getUit(); | 2 |
| Instantiate a large belt with negative effects | Belt(.TypeOfEffect.NEGATIVE, TypeOfBelt.LARGE) | {Ability.CONSTITUTION: -4},{Ability.CHARISMA, -4} |
| | belt.use(); | |
| | getUit(); | 4 |
| Instantiate a medium belt with positive effects | Belt(.TypeOfEffect.POSITIVE, TypeOfBelt.LARGE) | {Ability.CONSTITUTION: 4},{Ability.CHARISMA, 4} |
| | belt.use(); | |
| | getUit(); | 4 |
| Instantiate a potion with negative effects | Potion(.TypeOfEffect.NEGATIVE) | {a random Ability: x}(-3<x<0) |
| | potion.use(); | |
| Instantiate a potion with positive effects | Potion(.TypeOfEffect.POSITIVE) | {a random Ability: x}(0<x<=10) |
| | potion.use(); | |
| Instantiate a footwear with negative effects | Footwear(.TypeOfEffect.NEGATIVE) | {Ability.DEXTERITY: x}(-3< x<0) |
| | footwear.use(); | |
| Instantiate a footwear with positive effects | Footwear(.TypeOfEffect.POSITIVE) | {Ability.DEXTERITY: x}(0<x <=5) |
| | footwear.use(); | |
| | | |
| Testing EquipmentBagModel | | |

| | | |
|--|---|--|
| Instantiate a equipment bag and testing whether it contains 25% equipment with negative effects | EquipmentBagModel() | count |
| | int count = 0; | ==EquipmentBagModel. |
| | for(){ all the equipment, if it has negative effects , count ++} | SIZE * EquipmentBagModel.NEGATIVEPERCENT; |
| testing getEquipment() and make sure there are 20 items returned and the number of each type of items can not be greater than the total count of that type in the bag. | Equipmentbag playerbag = bag.getEquipment() | hnum + sbnum + mbnum + lbnum + fnum ==20; |
| | int hnum=0, pnum=0, sbnum=0, mbnum=0, lbnum=0,fnum=0; | hnum<= headgearNum, pnum <= potionNum, sbnum <= |
| | for(){ all the equipment in the playerBag, count the number of each type equipment} | smallBeltNum, mbnum <= mediumBeltNum, lbnum <= largeBeltNum, fnum <= footwearNum |
| testing getEquipment() when there is no equipment or the number of equipment < 20 | Equipmentbag playerbag = bag.getEquipment() | IllegalStateException |
| | | |
| | | |
| Testing weapons | | |
| Instantiate a Katanas | Katanas() | x (4<=x <=6) |
| | getDamage(null) | |
| Instantiate a Broadsword | Broadsword() | x (6<=x <=10) |
| | getDamage(null) | |
| Instantiate a TwoHandedSword, testing getDamage with a player whose strength < 14 | TwoHandedSword() | x(4<=x<=6) |
| | getDamage(10),10 means the strength of the player | |
| Instantiate a TwoHandedSword, testing getDamage with a player whose strength > 14 | TwoHandedSword() | x(8<=x<=12) |
| | getDamage(16),16 means the strength of the player | |
| Instantiate a Axes | Axe() | x (6<=x <=10) |
| | getDamage(null) | |
| Instantiate a Flail, testing getDamage with a player whose dexterity < 14 | Flail() | x(4<=x<=6) |
| | getDamage(10),10 means the dexterity of the player | |
| Instantiate a Flail, testing getDamage with a player whose dexterity > 14 | Flail() | x(8<=x<=12) |
| | getDamage(16),16 means the dexterity of the player | |
| | | |
| Testing ArmoryModel | | |
| | | |

| | | |
|--|---|--|
| Instantiate an ArmoryModel and testing if there are 5 types of weapons in the armory and each type has at least one weapon | ArmoryModel() | katanas.size()>=1, broadswords.size()>=1,t woHandedSwords.size()>=1,flails.size()>=1, axes.size()>=1 |
| testing requestWeapon() | Weapon weapon = requestWeapon() | weapon != null |
| testing requestWeapon() when there is no weapon in the Armory | Weapon weapon = requestWeapon() | IllegalStateException |
| | | |
| Testing Battle | | |
| Instantiate Battle | Battle battle = new Battle() | it can create two players, a equipment bag, an armory and each player has 20 gears, 1 weapon and each player has basic |
| | | |
| Testing rollDice() | Battle battle = new Battle() int x = battle.rollDice(); | 1 <= x <=6 |
| Testing assignCategories() | Battle battle = new Battle() Player player = new Player() battle.assignCategories(play er) | 2 <= player.constituion + player.strength + player.charisma + player.dexterity <= 24 |
| Testing getTurn(),the first player should have greater charisma than the other | Battle battle = new Battle() Player current = battle.getTurn() | current.charisma > the other's charisma |
| Testing getTurn() after playerA has attacked | Player current = getTurn() | current.getName().equals (playerB.getName()); |
| Testing isGameOver() after the game isn't over | boolean flag = battle.isGameOver(); | FALSE |
| Testing isGameOver() after the game is over | boolean flag = battle.isGameOver(); | TRUE |
| Testing getWinner() when playerA wins the game | Player winner = battle.getWinner() | winner.getName.equals(playerA.getName()); |
| Testing getWinner() when playerB wins the game | Player winner = battle.getWinner() | winner.getName.equals(playerB.getName()); |

| | | |
|---|---|--|
| Testing getWinner() when it's a tied game | Player winner = battle.getWinner() | winner == null |
| Testing getWinner() when the game hasn't been over | Player winner = battle.getWinner() | winner == null |
| Testing playerA.attack() when the game is over | playerA.attack() | IllegalStateException |
| Testing playerB.attack() when the game is over | playerB.attack() | IllegalStateException |
| | | |
| Testing Player | | |
| Testing constructor with the name Tom, basicStrength 10, basicConstitution 9, basicDexterity 5, basicCharisma 8 | Player player = new Player("Tom", 10, 9,5,8) | player.getName().equals("Tom");player.getBasicStrength()==10;player.getBasicConstitution==9;player.getBasicDexterity== 5; player.getBasicCharisma == 8 |
| Testing player use a potion with positive effect; | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be added the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getPotions().get(0).use(); | |
| Testing player use a potion with negative effect; | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be subtracted the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getPotions().get(0).use(); | |
| Testing player use a headwear with negative effect; | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be subtracted the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getHeadwear().get(0).use(); | |
| Testing player uses a headwear with positive effect; | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be added |
| | EquipmentBag bag = player.bag; | |

| | | |
|---|---|--|
| with positive effect, | Map<Ability, Integer> effects=bag.getHeadwear(). get(0).use(); | the value in the Map. |
| Testing player uses a headwear when he has used one | Player player = new Player("Tom", 10, 9,5,8) | IllegalStateException |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getHeadwear(). get(0).use(); | |
| Testing player uses a pair of foot wears with postive effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be added the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getFootwears(). get(0).use(); | |
| Testing player uses a pair of foot wears with negative effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be subtracted the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getFootwears(). get(0).use(); | |
| Testing player uses a pair of footwear when he has used one | Player player = new Player("Tom", 10, 9,5,8) | IllegalStateException |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getFootwears(). get(0).use(); | |
| Testing player uses a small belt with postive effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be added the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getSmallBelt().g et(0).use(); | |
| Testing player uses a small belt with negative effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be subtracted the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getSmallBelt().g et(0).use(); | |
| | Player player = new | |

| | | |
|--|---|---|
| Testing player uses a medium belt with positive effects | EquipmentBag bag = player.bag; | each Ability listed in the Map should be added the value in the Map. |
| | Map<Ability, Integer> effects=bag.getMediumBelt().get(0).use(); | |
| Testing player uses a medium belt with negative effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be subtracted the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getLargeBelt().get(0).use(); | |
| Testing player uses a large belt with positive effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be added the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getLargeBelt().get(0).use(); | |
| Testing player uses a large belt with negative effects | Player player = new Player("Tom", 10, 9,5,8) | each Ability listed in the Map should be subtracted the value in the Map. |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getMediumBelt().get(0).use(); | |
| Testing player uses a small belt when he has used 10 units of belts | Player player = new Player("Tom", 10, 9,5,8) | IllegalStateException |
| | EquipmentBag bag = player.bag; | |
| | Map<Ability, Integer> effects=bag.getBelts().get(0).use(); | |
| Testing player getStrikingPower(), if there are 20 strength in total after calling equip() | Player player = new Player("Tom", 10, 9,5,8) | 21<=x<=30 |
| | player.equip(); | |
| | int x = player.getStrikingPower() | |
| Testing getAvoidanceAbility() if | Player player = new Player("Tom", 10, 9,5,8) | |
| | | |

| | | |
|--|---|-------------|
| Testing getAvoidanceAbility(), if there are 15 dexterity in total after calling equip() | player.equip(); | 16<=x<=21 |
| | int x = player.getAvoidanceAbility() | |
| Testing getPotentialDamage(), if there are 21 strength in total after calling equip() and the player uses a Axe as a weapon | Player player = new Player("Tom", 10, 9,5,8) | 27<=x<=31 |
| | player.equip(); | |
| | int x = player.getPotentialDamage()) | |
| Testing getPotentialDamage(), if there are 21 strength in total after calling equip() and the player uses Katanas as a weapon | Player player = new Player("Tom", 10, 9,5,8) | 25<=x<=27 |
| | player.equip(); | |
| | int x = player.getPotentialDamage()) | |
| Testing getPotentialDamage(), if there are 20 strength in total after calling equip() and the player uses a Broadsword as a weapon | Player player = new Player("Tom", 10, 9,5,8) | 26<=x<=30 |
| | player.equip(); | |
| | int x = player.getPotentialDamage()) | |
| Testing getPotentialDamage(), if there are 10 strength in total after calling equip() and the player uses a TwoHandedSwords as a weapon but the player doesn't have enough strength to use it(strenght < 14) | Player player = new Player("Tom", 10, 9,5,8) | 14 <=x <=16 |
| | player.equip(); | |
| | int x = player.getPotentialDamage()) | |
| Testing getPotentialDamage(), if there are 20 strength in total after calling equip() and the player uses a TwoHandedSwords as a weapon | Player player = new Player("Tom", 10, 9,5,8) | 28<=x <=32 |
| | player.equip(); | |
| | int x = player.getPotentialDamage()) | |

| | | |
|---|--|--|
| Testing getPotentialDamage(), if there are 10 strength and 10 dexterity in total after calling equip() and the player uses a Flail as a weapon but the player doesn't have enough dexterity to use it(dexterity < 14) | Player player = new Player("Tom", 10, 9,5,8) | 14 <=x <=16 |
| | player.equip(); | |
| | int x = player.getPotentialDamage() | |
| Testing getPotentialDamage(), if there are 20 strength in total after calling equip() and the player uses a Flail as a weapon | Player player = new Player("Tom", 10, 9,5,8) | 28<=x <=32 |
| | player.equip(); | |
| | int x = player.getPotentialDamage() | |
| Testing attack(), if the striking power of PlayerA is greater than the avoidandce ability of the PlayerB, and the potential damage of PlayerA is 28 and the | playerA.attack(playerB) | playerB.getHealth() should have been deducated 8 |
| Testing attack(), if the striking power of PlayerA is smaller than the avoidandce ability of the PlayerB | playerA.attack(playerB) | no effect |
| Testing attack(), if the striking power of PlayerA is greater than the avoidandce ability of the PlayerB but they have the same value in potential damage and constitution respectively | playerA.attack(playerB) | no effect |
| Testing the constructor of Player | Player("Tom", 0,0,0,0) | IllegalArgumentExcetion |
| | Player("Tom", 0,10,10,10) | IllegalArgumentExcetion |
| | Player("Tom", 10,0,10,10) | IllegalArgumentExcetion |
| | Player("Tom", 10,10,0,10) | IllegalArgumentExcetion |
| | Player("Tom", 10,10,10,0) | IllegalArgumentExcetion |
| | Player("Tom", -1,-1,-1,-1) | IllegalArgumentExcetion |
| | Player("Tom", -1,10,10,10) | IllegalArgumentExcetion |
| | Player("Tom", 10,-1,10,10) | IllegalArgumentExcetion |
| | Player("Tom", 10,10,-1,10) | IllegalArgumentExcetion |
| | Player("Tom", 10,10,10,-1) | IllegalArgumentExcetion |