

Justin Phillips

Assignment 3b

cs362 winter 2020

1/29/2020

Action Card class

The screenshot shows the PyCharm IDE with the `ActionCard` class defined in `Card.py`. The class has methods `__init__`, `use`, `augment`, and `play`. The `play` method includes a loop that calls `player.draw()`. The test results window shows that the tests passed, with coverage percentages of 33% for `test__init__`, 66% for `test_augment`, and 100% for `test_use`. The coverage report on the right indicates that 22% of files and 30% of lines are covered.

```
class ActionCard(Card):
    def __init__(self, name, cost, actions, cards, buys, coins):
        Card.__init__(self, name, "action", cost, 0, 0, -1)
        self.actions = actions
        self.cards = cards
        self.buys = buys
        self.coins = coins
    def use(self, player, trash):
        player.player.append(self)
        player.hand.remove(self)
    def augment(self, player):
        player.actions += self.actions
        player.buys += self.buys
        player.pursets += self.coins
        for i in range(self.cards):
            player.draw()
    def play(self, player, players, supply, trash):
        ComputerPlayer = turn() while self.actions > 0 and 'action' in self.actions:
            if playthis:
                if c
```

Test Results: 3 tests passed - 0 ms

testAction_Dominion.py::actionCardTest::test__init__ PASSED [33%]
testAction_Dominion.py::actionCardTest::test_augment PASSED [66%]
testAction_Dominion.py::actionCardTest::test_use PASSED [100%]

3 passed in 6.58s

Process finished with exit code 0

Based on the coverage test it looks like all the lines in the Action Card class are covered except the `player.draw()` that is called in the for loop in the `augment` test. To cover this I believe I could add another assert that will check if a card has been drawn, This will be done after I added code where the player calls `draw` from cards and is added to their cards.

Player Class

The screenshot shows the PyCharm IDE with the `Player` class defined in `testgameover_Dominion.py`. The class has methods `__init__`, `other`, and `stack`. The `__init__` method initializes the player's name, hand, deck, played cards, discard pile, and aside. The `other` method returns the sum of played cards, discard pile, and aside. The `stack` method returns the sum of the deck, hand, discard pile, and aside. The `ComputerPlayer` class is also defined, which uses the `Player` class and implements the `turn` method.

The test results show that the `testgameover_Dominion.py` file is 100% covered, while the `testAction_Dominion.py` file is 30% covered. The test results also show that the `testgameover_Dominion.py` file is 100% covered, while the `testAction_Dominion.py` file is 30% covered.

Element	Statistics, %
idea	
pytest_cache	
Dominion.py	30% lines covered
playDominion.py	not covered
README.md	
REPLDominion.py	not covered
test_Dominion.py	100% lines covered
testAction_Dominion.py	not covered
testDominion1.py	not covered
testDominion2.py	not covered
testgameover_Dominion.py	not covered
testUtility.py	not covered

The draw function in the player class seems to be covered 100% .

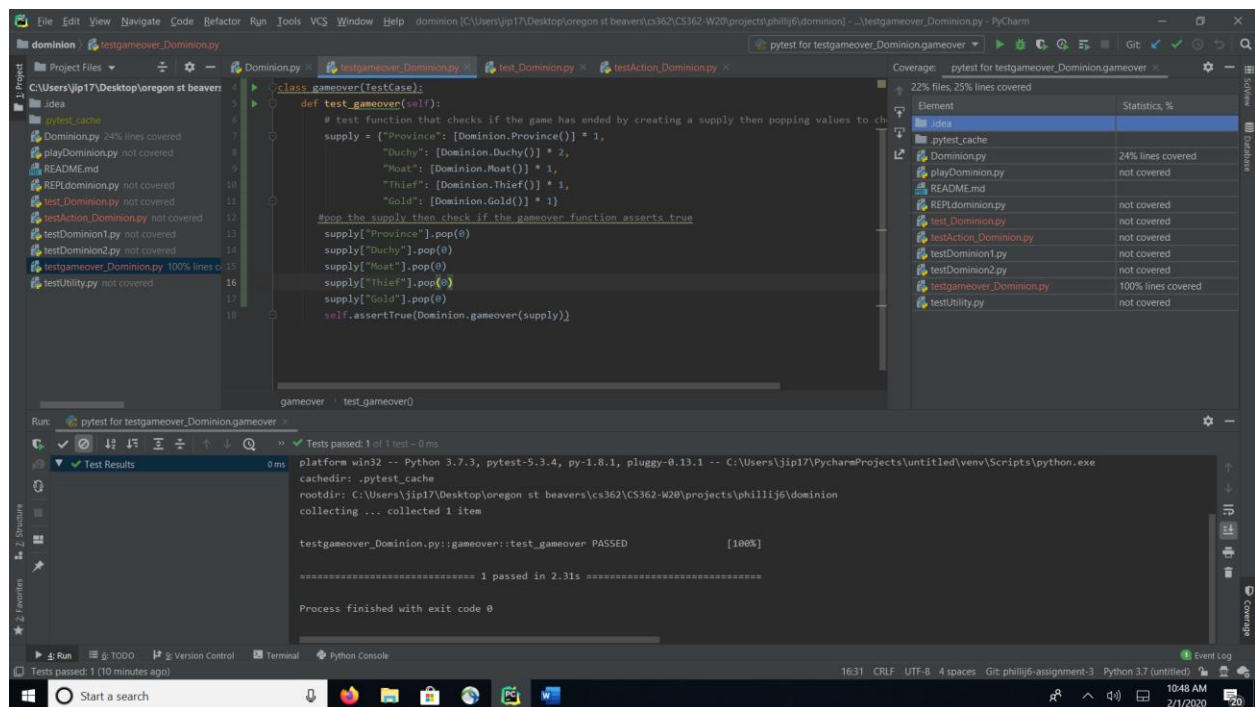
The screenshot shows the PyCharm IDE with the `Player` class defined in `testgameover_Dominion.py`. The class has methods `__init__`, `other`, `stack`, `action_balance`, `cardsummary`, `calcpoints`, and `gardens`. The `__init__` method initializes the player's name, hand, deck, played cards, discard pile, and aside. The `other` method returns the sum of played cards, discard pile, and aside. The `stack` method returns the sum of the deck, hand, discard pile, and aside. The `action_balance` method calculates the balance of the player's actions. The `cardsummary` method returns the summary of the player's cards. The `calcpoints` method calculates the player's points. The `gardens` method returns the number of gardens the player has.

The test results show that the `testgameover_Dominion.py` file is 100% covered, while the `testAction_Dominion.py` file is 30% covered. The test results also show that the `testgameover_Dominion.py` file is 100% covered, while the `testAction_Dominion.py` file is 30% covered.

Element	Statistics, %
idea	
pytest_cache	
Dominion.py	30% lines covered
playDominion.py	not covered
README.md	
REPLDominion.py	not covered
test_Dominion.py	100% lines covered
testAction_Dominion.py	not covered
testDominion1.py	not covered
testDominion2.py	not covered
testgameover_Dominion.py	not covered
testUtility.py	not covered

Also, the remainder of the tests `action_balance`, `cardsummary`, and `calcpoints` are all covered 100%

Game Over Test



The last function is to see if the game is over, this is checked by the `gameover` function and is also 100% covered except I am not sure why `assertTrue` doesn't have color next to it.