Test Plan

1. Create a Player object with the default constructor.
2. Create a Player object with the non-default constructor.
    1. with valid field values
    2. with invalid field values
3. Test all get methods:
    1. *getCharge()*
    2. *getConsumed()*
    3. *getEscaped()*
    4. *getName()*
    5. *getTurn()*
4. Test all set methods:
    1. *setCharge()*
        a) with valid field values
        b) with invalid field values
    2. *setConsumed()*
        a) with valid field values
        b) with invalid field values
    3. *setEscaped()*
        a) with valid field values
        b) with invalid field values
    4. *setName()*
        a) with valid field values
        b) with invalid field values
    5. *setTurn()*
        c) with valid field values
        d) with invalid field values

5. Test all other methods:
    1. *chargeDecrease()*
        a) with valid field values
        b) with invalid field values
    2. *chargeIncrease()*
        a) with valid field values
        b) with invalid field values
    3. *display()*
    4. *incrementConsumed()*
    5. *turnIncrease()*

Test 1

Create a Player object with the default constructor.

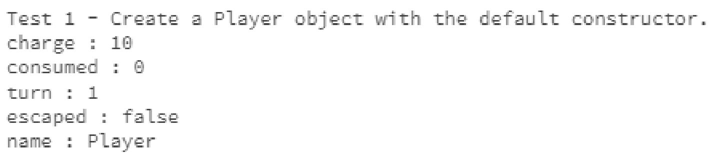
Test data:

- charge = 10
- consumed = 0
- turn = 1
- escaped = false
- name = “Player”

Expected results:

- charge : 10
- consumed : 0
- turn : 1
- escaped : false
- name : “Player”

Actual results: **Pass**

- 

```
Test 1 - Create a Player object with the default constructor.
charge : 10
consumed : 0
turn : 1
escaped : false
name : Player
```

Test 2.1

Create a Player object with the non-default constructor with valid fields.

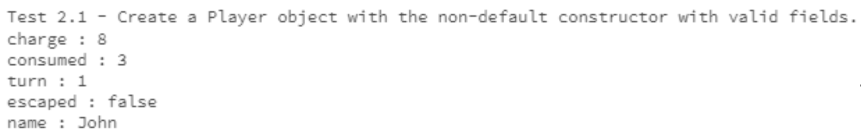
Test data:

- charge = 8
- consumed = 3
- turn = 1
- escaped = false
- name = “John”

Expected results:

- charge : 8
- consumed : 3
- turn : 1
- escaped : false
- name : “John”

Actual results: **Pass**

- 

```
Test 2.1 - Create a Player object with the non-default constructor with valid fields.
charge : 8
consumed : 3
turn : 1
escaped : false
name : John
```

Test 2.2ai

Create a Player object with the non-default constructor with invalid fields.

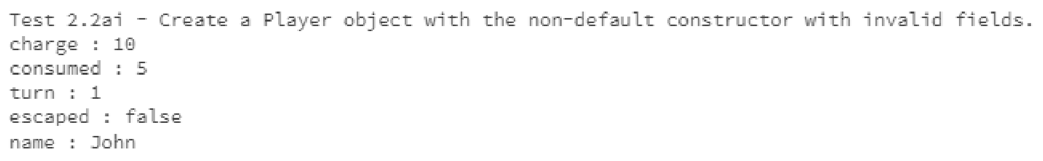
Test data:

- charge = -1
- consumed = 5
- turn = 1
- escaped = false
- name = “John”

Expected results:

- charge : 10
- consumed : 5
- turn : 1
- escaped : false
- name : “John”

Actual results: Pass

- 

```
Test 2.2ai - Create a Player object with the non-default constructor with invalid fields.
charge : 10
consumed : 5
turn : 1
escaped : false
name : John
```

Test 2.2aii

Create a Player object with the non-default constructor with invalid fields.

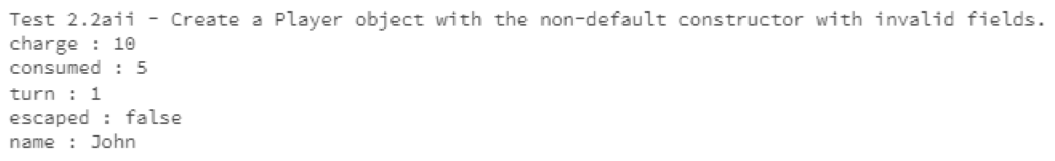
Test data:

- charge = 25
- consumed = 5
- turn = 1
- escaped = false
- name = “John”

Expected results:

- charge : 10
- consumed : 5
- turn : 1
- escaped : false
- name : “John”

Actual results: Pass

- 

```
Test 2.2aii - Create a Player object with the non-default constructor with invalid fields.
charge : 10
consumed : 5
turn : 1
escaped : false
name : John
```

<u>Test 2.2b</u>

Create a Player object with the non-default constructor with invalid fields.

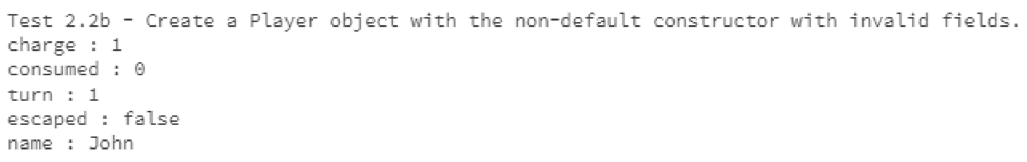
<u>Test data:</u>

- charge = 1
- consumed = -5
- turn = 1
- escaped = false
- name = “John”

<u>Expected results:</u>

- charge : 1
- consumed : 0
- turn : 1
- escaped : false
- name : “John”

<u>Actual results:</u> Pass

- 

  ```
  Test 2.2b - Create a Player object with the non-default constructor with invalid fields.
  charge : 1
  consumed : 0
  turn : 1
  escaped : false
  name : John
  ```

<u>Test 2.2c</u>

Create a Player object with the non-default constructor with invalid fields.

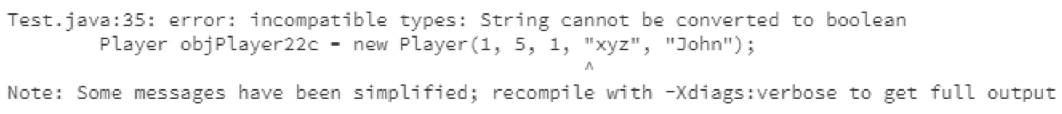
<u>Test data:</u>

- charge = 1
- consumed = 5
- turn = 1
- escaped = “xyz”
- name = “John”

<u>Expected results:</u>

- Compilation error

<u>Actual results:</u> Pass

- Compilation error
- 

  ```
  Test.java:35: error: incompatible types: String cannot be converted to boolean
          Player objPlayer22c = new Player(1, 5, 1, "xyz", "John");
                                                    ^
  Note: Some messages have been simplified; recompile with -Xdiags:verbose to get full output
  ```

<u>Test 2.2di</u>

Create a Player object with the non-default constructor with invalid fields.

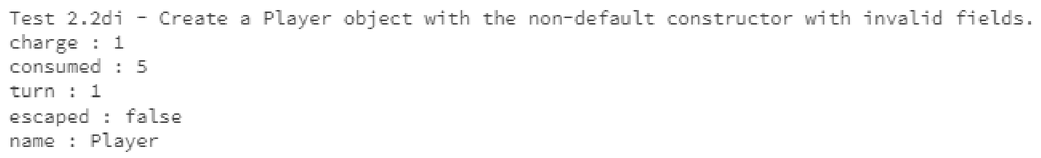
<u>Test data:</u>

- charge = 1
- consumed = 5
- turn = 1
- escaped = false
- name = “Jo”

<u>Expected results:</u>

- charge : 1
- consumed : 5
- turn : 1
- escaped : false
- name : “Player”

<u>Actual results:</u> Pass

- 

```
Test 2.2di - Create a Player object with the non-default constructor with invalid fields.
charge : 1
consumed : 5
turn : 1
escaped : false
name : Player
```

<u>Test 2.2dii</u>

Create a Player object with the non-default constructor with invalid fields.

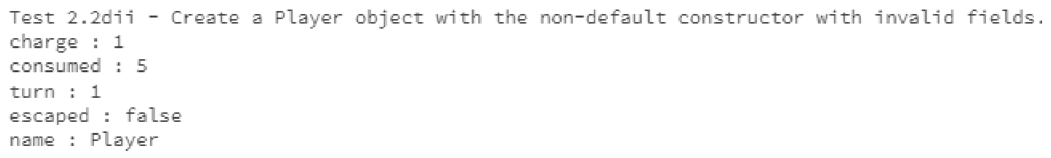
<u>Test data:</u>

- charge = 1
- consumed = 5
- turn = 1
- escaped = false
- name = “John Doe John Doe”

<u>Expected results:</u>

- charge : 1
- consumed : 5
- turn : 1
- escaped : false
- name : “Player”

<u>Actual results:</u> Pass

- 

```
Test 2.2dii - Create a Player object with the non-default constructor with invalid fields.
charge : 1
consumed : 5
turn : 1
escaped : false
name : Player
```

<u>Test 2.2e</u>

Create a Player object with the non-default constructor with valid fields.

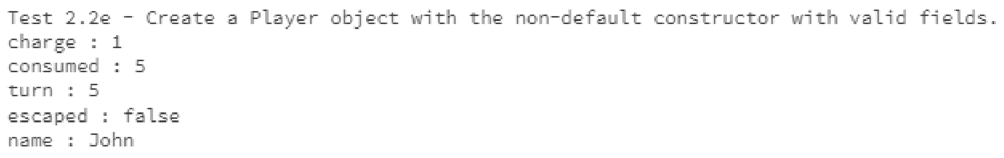
<u>Test data:</u>

- charge = 1
- consumed = 5
- turn = 5
- escaped = false
- name = “John”

<u>Expected results:</u>

- charge : 1
- consumed : 5
- turn : 5
- escaped : false
- name : “John”

<u>Actual results:</u> Pass

- 

```
Test 2.2e - Create a Player object with the non-default constructor with valid fields.
charge : 1
consumed : 5
turn : 5
escaped : false
name : John
```

<u>Test 2.2f</u>

Create a Player object with the non-default constructor with invalid fields.

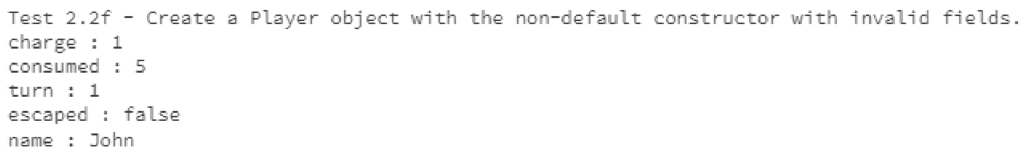
<u>Test data:</u>

- charge = 1
- consumed = 5
- turn = -5
- escaped = false
- name = “John”

<u>Expected results:</u>

- charge : 1
- consumed : 5
- turn : 1
- escaped : false
- name : “John”

<u>Actual results:</u> Pass

- 

```
Test 2.2f - Create a Player object with the non-default constructor with invalid fields.
charge : 1
consumed : 5
turn : 1
escaped : false
name : John
```

Test 3.1

Test *getCharge()*

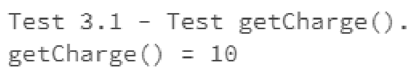
Test data:

- charge = 10

Expected results:

- getCharge() = 10

Actual results: **Pass**

- 

```
Test 3.1 - Test getCharge().
getCharge() = 10
```

Test 3.2

Test *getConsumed()*

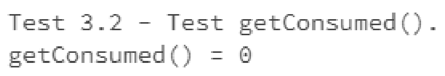
Test data:

- consumed = 0

Expected results:

- getConsumed() = 0

Actual results: **Pass**

- 

```
Test 3.2 - Test getConsumed().
getConsumed() = 0
```

Test 3.3

Test *getEscaped()*

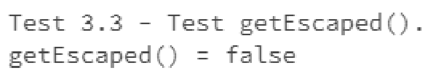
Test data:

- escaped = false

Expected results:

- getEscaped = false

Actual results: **Pass**

- 

```
Test 3.3 - Test getEscaped().
getEscaped() = false
```

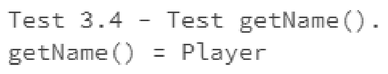
<u>Test 3.4</u>

Test *getName()*

<u>Test data:</u>
- name = “Player”

<u>Expected results:</u>
- getName = “Player”

<u>Actual results:</u> **Pass**

- 
```
Test 3.4 - Test getName().
getName() = Player
```

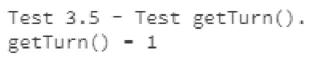
<u>Test 3.5</u>

Test *getTurn()*

<u>Test data:</u>
- turn = 1

<u>Expected results:</u>
- turn = 1

<u>Actual results:</u> **Pass**

- 
```
Test 3.5 - Test getTurn().
getTurn() = 1
```

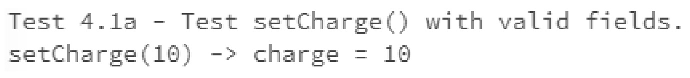
<u>Test 4.1a</u>

Test *setCharge()* with valid fields.

<u>Test data:</u>
- charge = 10

<u>Expected results:</u>
- charge = 10

<u>Actual results:</u> **Pass**

- 
```
Test 4.1a - Test setCharge() with valid fields.
setCharge(10) -> charge = 10
```

<u>Test 4.1bi</u>

Test *setCharge()* with invalid fields.

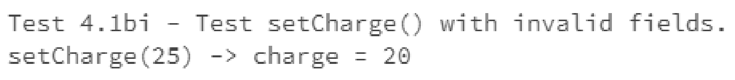
<u>Test data:</u>

- charge = 25

<u>Expected results:</u>

- charge = 20

<u>Actual results:</u> **Pass**

- 

```
Test 4.1bi - Test setCharge() with invalid fields.
setCharge(25) -> charge = 20
```

<u>Test 4.1bii</u>

Test *setCharge()* with invalid fields.

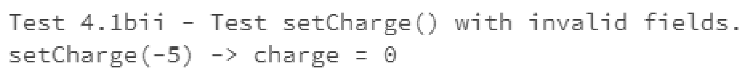
<u>Test data:</u>

- charge = -5

<u>Expected results:</u>

- charge = 0

<u>Actual results:</u> **Pass**

- 

```
Test 4.1bii - Test setCharge() with invalid fields.
setCharge(-5) -> charge = 0
```

<u>Test 4.2a</u>

Test *setConsumed()* with valid fields.

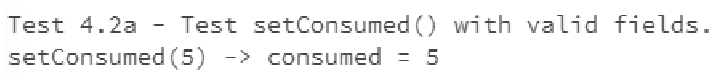
<u>Test data:</u>

- consumed = 5

<u>Expected results:</u>

- consumed = 5

<u>Actual results:</u> **Pass**

- 

```
Test 4.2a - Test setConsumed() with valid fields.
setConsumed(5) -> consumed = 5
```

Test 4.2b

Test *setConsumed ()* with invalid fields.

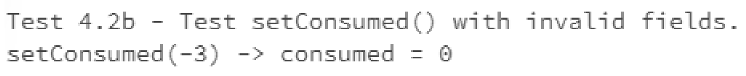
Test data:

- consumed = -3

Expected results:

- consumed = 0

Actual results: **Pass**

- 

```
Test 4.2b - Test setConsumed() with invalid fields.
setConsumed(-3) -> consumed = 0
```

Test 4.3a

Test *setEscaped()* with valid fields.

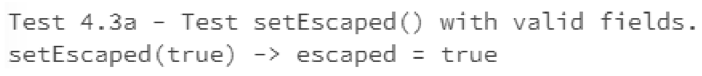
Test data:

- escaped = true

Expected results:

- escaped = true

Actual results: **Pass**

- 

```
Test 4.3a - Test setEscaped() with valid fields.
setEscaped(true) -> escaped = true
```

Test 4.3b

Test *setEscaped()* with invalid fields.

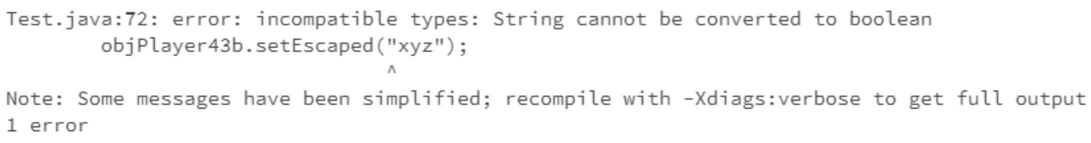
Test data:

- escaped = “xyz”

Expected results:

- Compilation error

Actual results: **Pass**

- Compilation error
- 

```
Test.java:72: error: incompatible types: String cannot be converted to boolean
        objPlayer43b.setEscaped("xyz");
                                ^
Note: Some messages have been simplified; recompile with -Xdiags:verbose to get full output
1 error
```

Test 4.4a

Test *setName()* with valid fields.

Test data:

- name = “John Doe”

Expected results:

- name = “John Doe”

Actual results: **Pass**

-
  ```
  Test 4.4a - Test setName() with valid fields.
  setName to John Doe -> name = John Doe
  ```

Test 4.4bi

Test *setName()* with invalid fields.

Test data:

- name = “Jo”

Expected results:

- name = “Player”

Actual results: **Pass**

-
  ```
  Test 4.4bi - Test setName() with invalid fields.
  setName to Jo -> name = Player
  ```

Test 4.4bii

Test *setName()* with invalid fields.

Test data:

- name = “John Doe John Doe”

Expected results:

- name = “Player”

Actual results: **Pass**

-
  ```
  Test 4.4bii - Test setName() with invalid fields.
  setName to John Doe John Doe -> name = Player
  ```

<u>Test 4.5a</u>

Test *setTurn()* with valid fields.

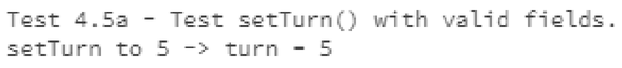
<u>Test data:</u>

- turn = 5

<u>Expected results:</u>

- turn = 5

<u>Actual results:</u> **Pass**

- 
```
Test 4.5a - Test setTurn() with valid fields.
setTurn to 5 -> turn = 5
```

<u>Test 4.5b</u>

Test *setTurn()* with invalid fields.

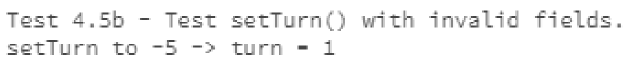
<u>Test data:</u>

- turn = -5

<u>Expected results:</u>

- turn = 1

<u>Actual results:</u> **Pass**

- 
```
Test 4.5b - Test setTurn() with invalid fields.
setTurn to -5 -> turn = 1
```

<u>Test 5.1a</u>

Test *chargeDecrease()* with valid fields.

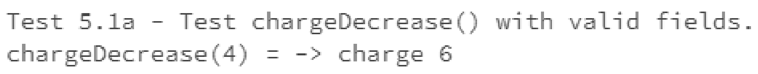
<u>Test data:</u>

- charge = 10
- decrease = 4

<u>Expected results:</u>

- charge = 6

<u>Actual results:</u> **Pass**

- 
```
Test 5.1a - Test chargeDecrease() with valid fields.
chargeDecrease(4) = -> charge 6
```

Test 5.1b

Test *chargeDecrease()* with invalid fields.

Test data:

- charge = 10
- decrease = 15

Expected results:

- charge = 0

Actual results: **Pass**

-
  ```
  Test 5.1b - Test chargeDecrease() with invalid fields.
  chargeDecrease(15) = -> charge 0
  ```

Test 5.2a

Test *chargeIncrease()* with valid fields.

Test data:

- charge = 10
- increase = 4

Expected results:

- charge = 14

Actual results: **Pass**

-
  ```
  Test 5.2a - Test chargeIncrease() with valid fields.
  chargeIncrease(4) = -> charge 14
  ```

Test 5.2b

Test *chargeIncrease()* with invalid fields.

Test data:

- charge = 10
- increase = 15

Expected results:

- charge = 20

Actual results: **Pass**

-
  ```
  Test 5.2b - Test chargeIncrease() with invalid fields.
  chargeIncrease(15) = -> charge 20
  ```

<u>Test 5.3</u>

Test *display()*

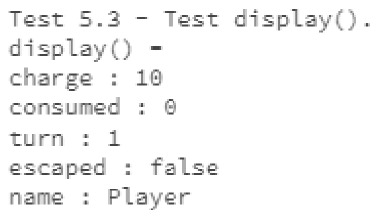
<u>Test data:</u>

- charge = 10
- consumed = 0
- turn = 1
- escaped = false
- name = “Player”

<u>Expected results:</u>

- “Charge : 10
  Consumed : 0
  Turn : 1
  Escaped : false
  Name : Player”

<u>Actual results:</u> **Pass**

- 

```
Test 5.3 - Test display().
display() -
charge : 10
consumed : 0
turn : 1
escaped : false
name : Player
```

<u>Test 5.4</u>

Test *incrementConsumed()*

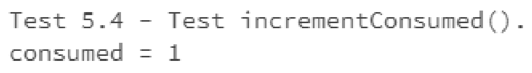
<u>Test data:</u>

- consumed = 0

<u>Expected results:</u>

- consumed = 1

<u>Actual results:</u> **Pass**

- 

```
Test 5.4 - Test incrementConsumed().
consumed = 1
```

Test 5.5

Test *turnIncrease()*

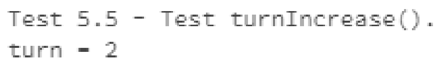
Test data:

- turn = 1

Expected results:

- turn = 2

Actual results: **Pass**

- 

```
Test 5.5 - Test turnIncrease().
turn = 2
```