Because Player class is an abstract class in my design, an instance of Player can not be created directly. Instead, I will create an instance of Player's subclass, for example, Forward class.

Test Plan

1. Create a Forward object with the default constructor.
2. Create a Forward object with the first non-default constructor:
    - with valid field values
    - with invalid field values
3. Create a Forward object with the second non-default constructor:
    - with valid field values
    - with invalid field values
4. Test display/toString method.
5. Test all get methods:
    - Test getBehinds()
    - Test getFieldPosition()
    - Test getGoals()
    - Test getIsInjured()
    - Test getIsReported()
    - Test getIsStar()
    - Test getKicks()
    - Test getPass()
    - Test getPlayerName()
    - Test getSeasonGoals()
    - Test getTeam()
6. Test all set methods:
    - Test setBehinds()
        - with valid field values
        - with invalid field values
    - Test setFieldPosition()
        - with valid field values
        - with invalid field values
    - Test setGoals()
        - with valid field values
        - with invalid field values
    - Test setIsInjured()
        - with valid field values
        - with invalid field values
    - Test setIsReported()
        - with valid field values
        - with invalid field values
    - Test setIsStar()
        - with valid field values
        - with invalid field values

- Test setKicks()
  - with valid field values
  - with invalid field values
- Test setPass()
  - with valid field values
  - with invalid field values
- Test setPlayerName()
  - with valid field values
  - with invalid field values
- Test setSeasonGoals()
  - with valid field values
  - with invalid field values
- Test setTeam()
  - with valid field values
  - with invalid field values

7. Test other methods:
   - Test addABehind()
   - Test addAGoal()
   - Test addAKick()
   - Test addAPass()
   - Test calculatePercent()
   - Test getReported()
   - Test getInjured()

The actual tests

Test 1

Create a Forward object with the default constructor.

Test data:

Expected results:

playerName: “default name”
fieldPosition: “default position”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0
pass = 0
isStar: false
isInjured: false
isReported: false

Actual results:

```
Create a Forward object with the default constructor
playerName   : default name
fieldPosition: default position
seasonGoals  : 0
team         : null
kicks        : 0
goals        : 0
behinds      : 0
pass         : 0
isStar       : false
isInjured    : false
isReported   : false
```

Test passed!

Test 2.1

Create a Forward object with the first non-default constructor with valid field values.

Test data:

playerName: “playerA1”
fieldPosition: “Forward”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0
pass = 0
isStar: false
isInjured: false
isReported: false

Expected results:

playerName: “playerA1”
fieldPosition: “Forward”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0
pass = 0
isStar: false
isInjured: false
isReported: false

Actual results:

```
Create a Forward object with the first non-default constructor with valid field values
playerName   : playerA1
fieldPosition: Forward
seasonGoals  : 0
team         : null
kicks        : 0
goals        : 0
behinds      : 0
pass         : 0
isStar       : false
isInjured    : false
isReported   : false
```

Test passed!

Test 2.1 b.1

Create a Forward object with the first non-default constructor with invalid field values.

Test data:

playerName: null
fieldPosition: “Forward”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0
pass = 0
isStar: false
isInjured: false
isReported: false

Expected results:

“Name cannot be null or empty”

Actual results:

```
Create a Forward object with the first non-default constructor with invalid field values
Name cannot be null or empty
```

Test passed!

Test 2.2

Create a Forward object with the second non-default constructor with valid field values.

Test data:

playerName: “playerA1”
fieldPosition: “Forward”
seasonGoals: 0
team: null

Expected results:

playerName: “playerA1”
fieldPosition: “Forward”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0
pass = 0
isStar: false
isInjured: false
isReported: false

Actual results:

```
Create a Forward object with the second non-default constructor with valid field values
playerName   : playerA1
fieldPosition: Forward
seasonGoals  : 0
team         : null
kicks        : 0
goals        : 0
behinds      : 0
pass         : 0
isStar       : false
isInjured    : false
isReported   : false
```

Test passed!

Test 2.2 b.1

Create a Forward object with the second non-default constructor with invalid field values.

Test data:

playerName: null
fieldPosition: “Forward”
seasonGoals: 0
team: null

Expected results:

“Name cannot be null or empty”

Actual results:

```
Create a Forward object with the second non-default constructor with invalid field values
Name cannot be null or empty
```

Test passed!

Test 2.3

Test display method.

Test data:

playerName: “playerA1”
fieldPosition: “Forward”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0

pass = 0
isStar: false
isInjured: false
isReported: false

Expected results:

playerName: “playerA1”
fieldPosition: “Forward”
seasonGoals: 0
team: null
kicks: 0
goals = 0
behinds = 0
pass = 0
isStar: false
isInjured: false
isReported: false

Actual results:

```
Test display method
playerName   : playerA1
fieldPosition: Forward
seasonGoals  : 0
team         : null
kicks        : 0
goals        : 0
behinds      : 0
pass         : 0
isStar       : false
isInjured    : false
isReported   : false
```

Test passed!

Test 2.4

Test getPlayerName method.

Test data:

playerName: “playerA1”

Expected results:

playerName: “playerA1”

Actual results:

```
Test getPlayerName method
playerName: playerA1
```

Test passed!

Test 2.5

Test setPlayerName method with valid values.

Test data:

playerName: “playerA1”

Expected results:

Argument is valid: true

Actual results:

```
Test setPlayerName method with valid argument
Argument is valid: true
```

Test passed!

Test 2.5 b.1

Test setPlayerName method with invalid values.

Test data:

playerName: null

Expected results:

Argument is valid: false

Actual results:

```
Test setPlayerName method with invalid argument
Argument is valid: false
```

Test passed!