Hanlu Shi
31017835
FIT9131 Assignment B Test Strategy

Test strategy for PenguinFamily class

1) Test plan

1. create an object of PenguinFamily using default constructor

1.1 Create an object of PenguinFamily using the default constructor.

1.2 Verify that the familyChicks and familyEggs ArrayLists are empty.

1.3 Verify that the malePenguin and femalePenguin objects are null.

1.4 Verify that the idCode is an empty string.

2. create an object of PenguinFamily using non-default constructor

2.1 Create an object of PenguinFamily using the non-default constructor with valid values

2.2 with valid ArrayList for familyChicks , femalePenguin and malePenguin, valid idCode

2.3 with null/invalid values for parameters to test error handling

3. test all get methods

3.1 getFamilyChicks()

3.2 getFamilyEggs()

3.3 getFemalePenguin()

3.4 getIdCode()

3.5 getMalePenguin()

3.6 getSpecificChick()

3.7 getSpecificEgg()

4. test all set methods

4.1 setFamilyChicks() - valid and invalid ArrayList

4.2 setFamilyEggs() - valid and invalid ArrayList

4.3 setFemalePenguin() - valid and invalid Penguin

4.4 setIdCode() - valid and invalid string

4.5 setMalePenguin()

5. test other methods

5.1 breedEggsAndAlive() – verify

5.2 display() - verify print details of all familyChicks

2) Actual Tests

Test 1

Steps:

1.Create an object of PenguinFamily the default constructor

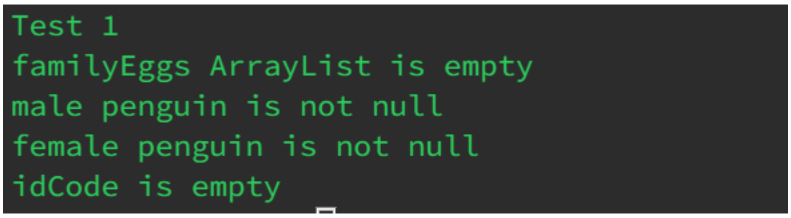
Expected results:

- familyChicks and familyEggs ArrayList should be empty

- male penguin and female penguin should not be null

- idCode should be blank

Actual results:



```
Test 1
familyEggs ArrayList is empty
male penguin is not null
female penguin is not null
idCode is empty
```

Test Passed!

Test 2

Steps:

1.Create an object of PenguinFamily with the non-default constructor with valid values

Test data:

- 2 objects of chick

- 2 objects of eggs

- 1 female penguin of penguin object

- 1 male penguin of penguin object

- idCode: "P002"

Expected results:

familyChicks.size() = 2

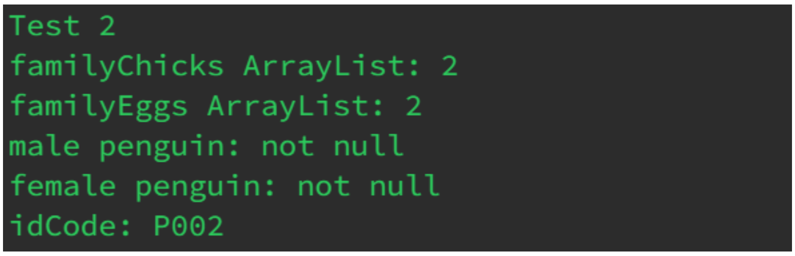
familyEggs.size() = 2

femalePenguin != null

malePenguin != null

idCode = "P002"

Actual results:



```
Test 2
familyChicks ArrayList: 2
familyEggs ArrayList: 2
male penguin: not null
female penguin: not null
idCode: P002
```

Test Passed!

Test 3: invalid Egg

Steps:

1.Create an object of PenguinFamily with the non-default constructor with invalid values

Test data:

- 1 object of chick

- -3 objects of eggs

- 1 female penguin of penguin object

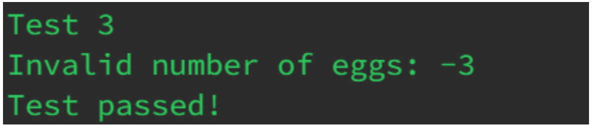
- 1 male penguin of penguin object

- idCode: "P002"

Expected results:

familyEggs.size() : Invalid number

Actual results:



```
Test 3
Invalid number of eggs: -3
Test passed!
```

Test Passed!

Test 4: invalid Chick Index

Steps:

1.Create an object of PenguinFamily with the non-default constructor with valid values

2.Call getSpecificChick(-1)

Test data:

- 1 object of chick

- 2 objects of eggs

- 1 female penguin of penguin object

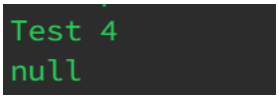
- 1 male penguin of penguin object

- idCode: "P002"

Expected results:

-return null

Actual results:



```
Test 4
null
```

Test Passed!

Test 5:

Description: Test the getFamilyChicks() method.

Steps:

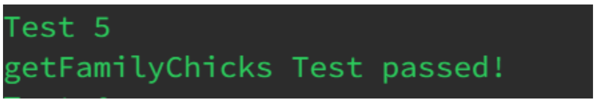
Create an object of PenguinFamily with a known ArrayList of familyChicks.

Call the getFamilyChicks() method.

Expected Results:

The returned ArrayList should be the same as the one set in the PenguinFamily object.

Actual results:



```
Test 5
getFamilyChicks Test passed!
```

Test Passed!

Test 6:

Description: Test the getFamilyEggs() method.

Steps:

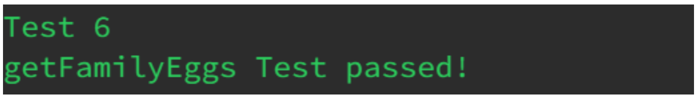
Create an object of PenguinFamily with a known ArrayList of familyEggs.

Call the getFamilyEggs() method.

Expected Results:

The returned ArrayList should be the same as the one set in the PenguinFamily object.

Actual results:



```
Test 6
getFamilyEggs Test passed!
```

Test Passed!

Test 7:

Description: Test the getFemalePenguin() method.

Steps:

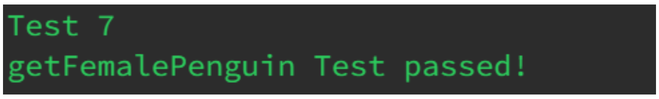
Create an object of PenguinFamily with a known femalePenguin object.

Call the getFemalePenguin() method.

Expected Results:

The returned femalePenguin object should be the same as the one set in the PenguinFamily object.

Actual results:



```
Test 7
getFemalePenguin Test passed!
```

Test Passed!

Test 8:

Description: Test the getIdCode() method.

Steps:

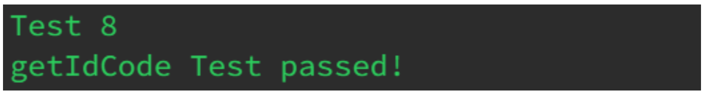
Create an object of PenguinFamily with a known idCode.

Call the getIdCode() method.

Expected Results:

The returned idCode should be the same as the one set in the PenguinFamily object.

Actual results:



```
Test 8
getIdCode Test passed!
```

Test Passed!

Test 9:

Description: Test the getMalePenguin() method.

Steps:

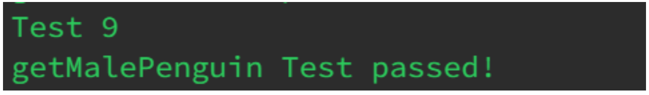
Create an object of PenguinFamily with a known malePenguin object.

Call the getMalePenguin() method.

Expected Results:

The returned malePenguin object should be the same as the one set in the PenguinFamily object.

Actual results:



```
Test 9
getMalePenguin Test passed!
```

Test Passed!

Test 10:

Description: Test the setFamilyChicks() method with a valid ArrayList.

Steps:

Create an object of PenguinFamily.

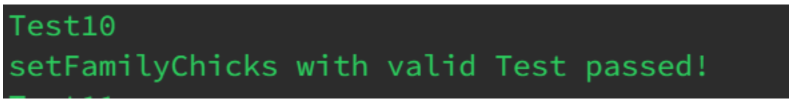
Create a valid ArrayList of Penguin objects.

Call the setFamilyChicks() method with the valid ArrayList.

Expected Results:

The familyChicks ArrayList in the PenguinFamily object should be updated with the provided ArrayList.

Actual results:



```
Test10
setFamilyChicks with valid Test passed!
```

Test Passed!

Test 11:

Description: Test the setFamilyChicks() method with a null ArrayList.

Steps:

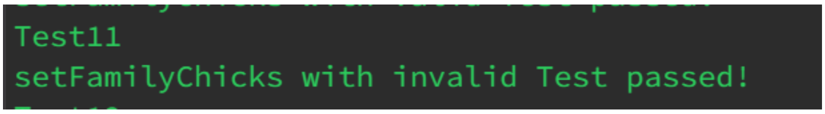
Create an object of PenguinFamily.

Call the setFamilyChicks() method with a null ArrayList.

Expected Results:

The familyChicks ArrayList in the PenguinFamily object should remain unchanged or be set to a default value, depending on the implementation's error handling.

Actual results:



```
Test11
setFamilyChicks with invalid Test passed!
```

Test Passed!

Test 12:

Description: Test the setFamilyEggs() method with a valid ArrayList.

Steps:

Create an object of PenguinFamily.

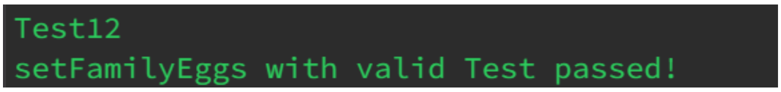
Create a valid ArrayList of Penguin objects.

Call the setFamilyEggs() method with the valid ArrayList.

Expected Results:

The familyEggs ArrayList in the PenguinFamily object should be updated with the provided ArrayList.

Actual results:



```
Test12
setFamilyEggs with valid Test passed!
```

Test Passed!

Test 13:

Description: Test the setFamilyEggs() method with a null ArrayList.

Steps:

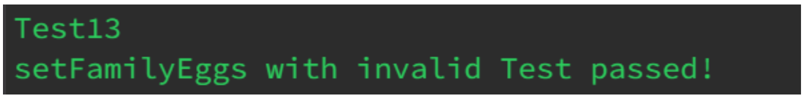
Create an object of PenguinFamily.

Call the setFamilyEggs() method with a null/invalid ArrayList.

Expected Results:

The familyEggs ArrayList in the PenguinFamily object should remain unchanged or be set to a default value, depending on the implementation's error handling.

Actual results:



```
Test13
setFamilyEggs with invalid Test passed!
```

Test Passed!

Test 14:

Description: Test the setFemalePenguin() method with a valid femalePenguin object.

Steps:

Create an object of PenguinFamily.

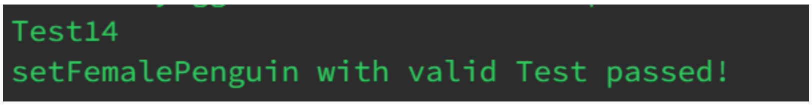
Create a valid femalePenguin object.

Call the setFemalePenguin() method with the valid femalePenguin object.

Expected Results:

The femalePenguin object in the PenguinFamily object should be updated with the provided femalePenguin object.

Actual results:



```
Test14
setFemalePenguin with valid Test passed!
```

Test Passed!

Test 15:

Description: Test the setFemalePenguin() method with a null/invalid femalePenguin object.

Steps:

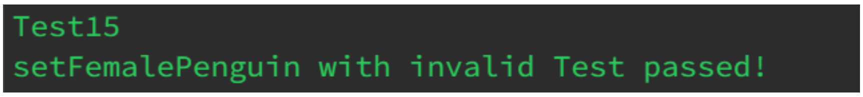
Create an object of PenguinFamily.

Call the setFemalePenguin() method with a null object.

Expected Results:

The femalePenguin object in the PenguinFamily object should remain unchanged or be set to a default value, depending on the implementation's error handling.

Actual results:



```
Test15
setFemalePenguin with invalid Test passed!
```

Test Passed!

Test 16:

Description: Test the setIdCode() method with a valid idCode.

Steps:

Create an object of PenguinFamily.

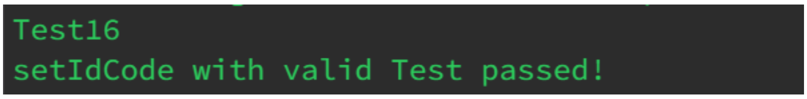
Create a valid idCode.

Call the setIdCode() method with the valid idCode.

Expected Results:

The idCode in the PenguinFamily object should be updated with the provided idCode.

Actual results:



```
Test16
setIdCode with valid Test passed!
```

Test Passed!

Test 17:

Description: Test the setIdCode() method with a null/invalid idCode.

Steps:

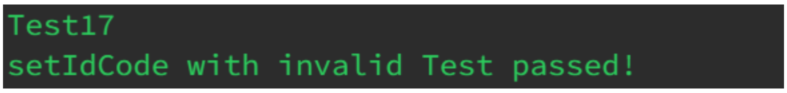
Create an object of PenguinFamily.

Call the setIdCode() method with a invalid idCode.

Expected Results:

The idCode in the PenguinFamily object should remain unchanged or be set to a default value, depending on the implementation's error handling.

Actual results:



```
Test17
setIdCode with invalid Test passed!
```

Test Passed!

Test 18:

Description: Test the setMalePenguin() method with a valid malePenguin object.

Steps:

Create an object of PenguinFamily.

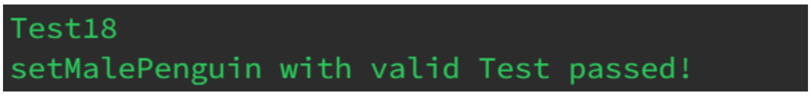
Create a valid malePenguin object.

Call the setMalePenguin() method with the valid malePenguin object.

Expected Results:

The malePenguin object in the PenguinFamily object should be updated with the provided malePenguin object.

Actual results:



```
Test18
setMalePenguin with valid Test passed!
```

Test Passed!

Test 19:

Description: Test the setMalePenguin() method with a null/invalid malePenguin object.

Steps:

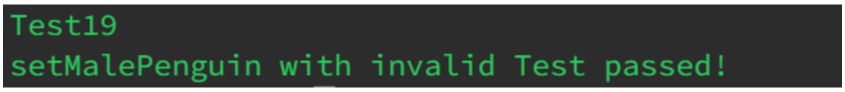
Create an object of PenguinFamily.

Call the setMalePenguin() method with a null object.

Expected Results:

The malePenguin object in the PenguinFamily object should remain unchanged or be set to a default value, depending on the implementation's error handling.

Actual results:



```
Test19
setMalePenguin with invalid Test passed!
```

Test Passed!

Test 20:

Description: Test Plan for breedEggsAndAlive() Method:

Test Case 1: Both male and female penguins are alive

Set malePenguin.aliveFlag and femalePenguin.aliveFlag to true.

Call breedEggsAndAlive() method.

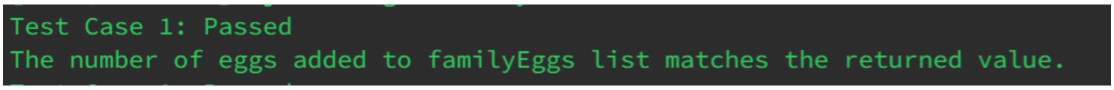
Verify that the method returns a value between 0 and 2 (inclusive).

Verify that the number of eggs added to familyEggs list matches the returned value.

Expected Results:

The number of eggs added to familyEggs list matches the returned value.

Actual results:



```
Test Case 1: Passed
The number of eggs added to familyEggs list matches the returned value.
```

Test Passed!

Test Case 2: Both male and female penguins are not alive

Set malePenguin.aliveFlag and femalePenguin.aliveFlag to false.

Add some existing chicks and eggs to familyChicks and familyEggs lists.

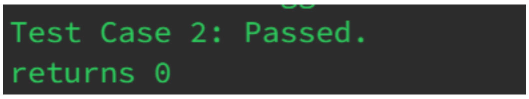
Call breedEggsAndAlive() method.

Verify that the method returns 0.

Expected Results:

returns 0

Actual results:



```
Test Case 2: Passed.
returns 0
```

Test Passed!

Test Case 3: Either male or female penguin is not alive

Set malePenguin.aliveFlag to true and femalePenguin.aliveFlag to false.

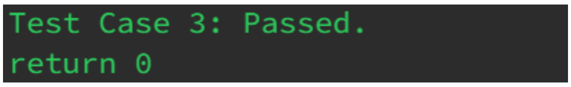
Call breedEggsAndAlive() method.

Verify that the method returns 0.

Expected Results:

returns 0

Actual results:



```
Test Case 3: Passed.
return 0
```

Test Passed!

Test 21:

Description: Test Plan for display() Method:

Test data:

- 3 objects of chick
- 2 objects of eggs
- 1 female penguin of penguin object
- 1 male penguin of penguin object

- idCode: "P010"

Expected results:

The details of the whole family:

Family ID: P010

Male Penguin Details:

Penguin [gender=M, alive Flag=true]

Female Penguin Details:

Penguin [gender=M, alive Flag=true]

Number of Chicks: 3

Chick 1 Details:

[Chick gender=M, chick age=1]

Chick 2 Details:

[Chick gender=F, chick age=1]

Chick 3 Details:

[Chick gender=M, chick age=1]

Number of Eggs: 2

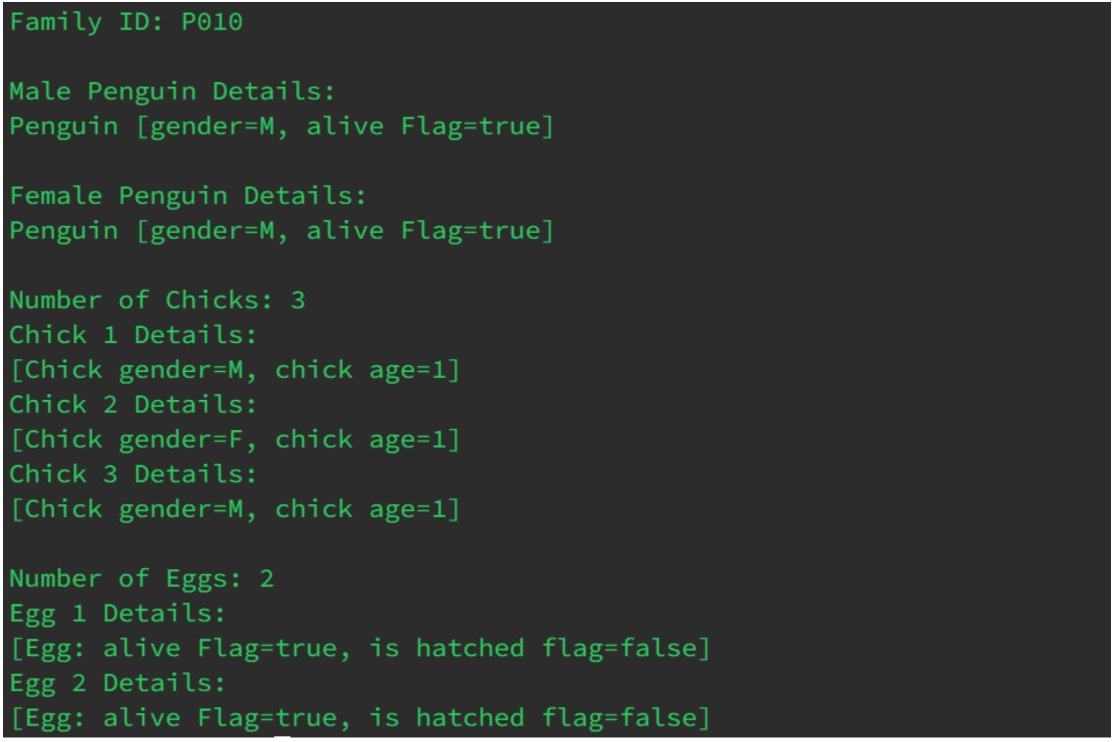
Egg 1 Details:

[Egg: alive Flag=true, is hatched flag=false]

Egg 2 Details:

[Egg: alive Flag=true, is hatched flag=false]

Actual results:



```
Family ID: P010

Male Penguin Details:
Penguin [gender=M, alive Flag=true]

Female Penguin Details:
Penguin [gender=M, alive Flag=true]

Number of Chicks: 3
Chick 1 Details:
[Chick gender=M, chick age=1]
Chick 2 Details:
[Chick gender=F, chick age=1]
Chick 3 Details:
[Chick gender=M, chick age=1]

Number of Eggs: 2
Egg 1 Details:
[Egg: alive Flag=true, is hatched flag=false]
Egg 2 Details:
[Egg: alive Flag=true, is hatched flag=false]
```

Test Passed!