

## Test Strategy

### Class diagram

Ship
shipName: String xPos: int yPos: int noOfHitsMade: int noOfHitsNeeded: int
Ship() Ship(shipName: String, xPos: int, yPos: int, noOfHitsMade: int, noOfHitsNeeded: int ) getShipName(): String getXpos(): int getYpos(): int getNoOfHitsMade(): int getnoOfHitsNeeded(): int setShipName(shipName: String): void setXpos(xPos: int): void setYpos(yPos: int): void setNoOfHitsMade(noOfHitsMade: int): void setnoOfHitsNeeded(noOfHitsNeeded: int): void madeIncrement(): void display(): void

### Test Plan

1. Create a Ship object with default constructor
2. Create s Ship object with non-default constructor
  - a. With valid field values
  - b. With invalid field values
3. Test all the get methods
4. Test all the set methods
  - a. With valid arguments
  - b. With invalid arguments
5. Test the display method

## Test 1

Create a ship object with default constructor

Test Data:

- No input

Expected Results:

- shipName: ""
- xPos: 0
- yPos: 0
- noOfHitsMade: 0
- noOfHitsNeeded: 0

Actual Results:

ship1 : Ship

private String shipName	""
private int xPos	0
private int yPos	0
private int noOfHitsMade	0
private int noOfHitsNeeded	0

Inspect

Get

Show static fields

Close

## Test 2

- a. Create a ship object with non-default constructor with valid values

### Test Data:

- shipName: "Titan"
- xPos: 3
- yPos: 1
- noOfHitsMade: 2
- noOfHitsNeeded: 4

### Expected Results

- shipName: "Titan"
- xPos: 3
- yPos: 1
- noOfHitsMade: 2
- noOfHitsNeeded: 4

### Actual Results

ship1 : Ship

private String shipName	"Titan"	Inspect  Get
private int xPos	3	
private int yPos	1	
private int noOfHitsMade	2	
private int noOfHitsNeeded	4	

Show static fieldsClose

b. Create a ship object with non-default constructor with invalid values

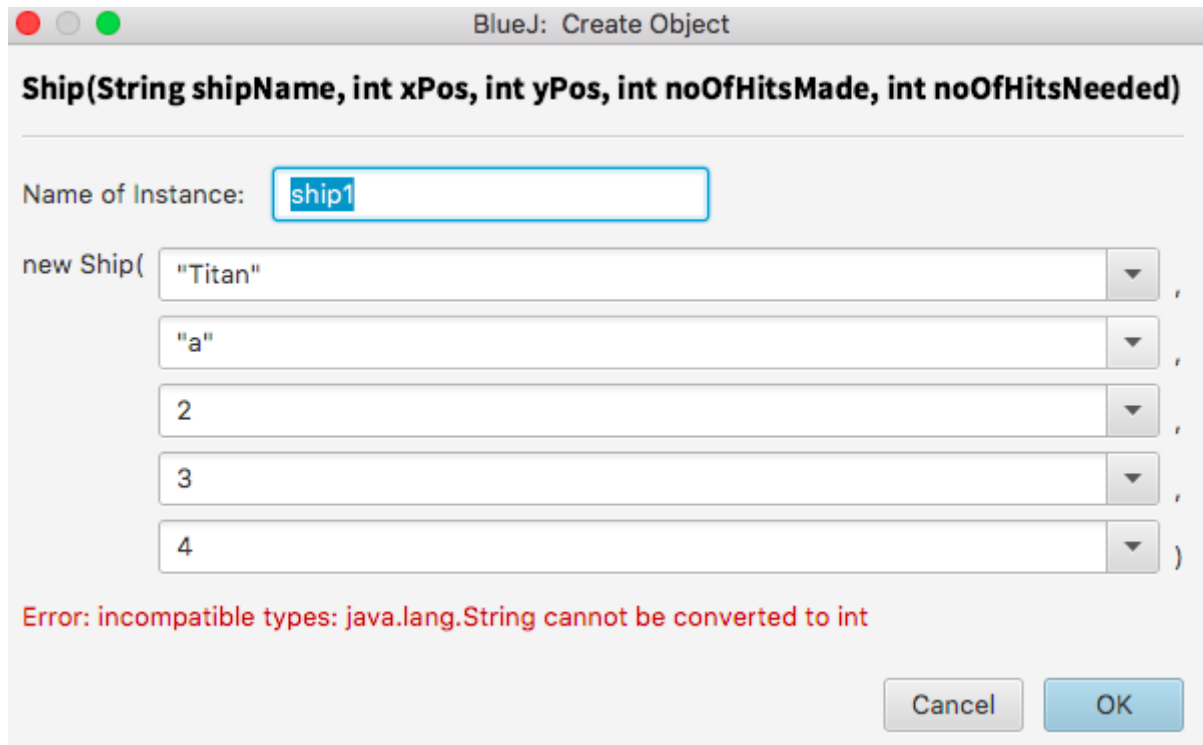
Test Data:

- shipName: "Titan"
- xPos: "a"
- yPos: 2
- noOfHitsMade: 3
- noOfHitsNeeded: 4

Expected Results

- shipName: "Titan"
- xPos: Error, Incompatible types
- yPos: 2
- noOfHitsMade: 3
- noOfHitsNeeded: 4

Actual results:



The image shows a screenshot of the BlueJ 'Create Object' dialog box for the `Ship` class. The title bar reads 'BlueJ: Create Object'. The main title is `Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)`. Below this, there is a field for 'Name of Instance:' with the value 'ship1'. The 'new Ship(' line is followed by five input fields: 'Titan', 'a', '2', '3', and '4'. Each field has a dropdown arrow on its right. The fields are separated by commas, and the last field is followed by a closing parenthesis. At the bottom, there is a red error message: 'Error: incompatible types: java.lang.String cannot be converted to int'. There are 'Cancel' and 'OK' buttons at the bottom right.

BlueJ: Create Object

**Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)**

Name of Instance:

new Ship(  ,  
 ,  
 ,  
 ,  
 )

Error: incompatible types: java.lang.String cannot be converted to int

Cancel OK

c.

Test Data:

- shipName: "Titan"
- xPos: 1
- yPos: "a"
- noOfHitsMade: 3
- noOfHitsNeeded: 4

Expected Results

- shipName: "Titan"
- xPos: 1
- yPos: Error, Incompatible types
- noOfHitsMade: 3
- noOfHitsNeeded: 4

BlueJ: Create Object

**Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)**

Name of Instance:

new Ship(  ,  
 ,  
 ,  
 ,  
 )

Error: incompatible types: java.lang.String cannot be converted to int

Cancel OK

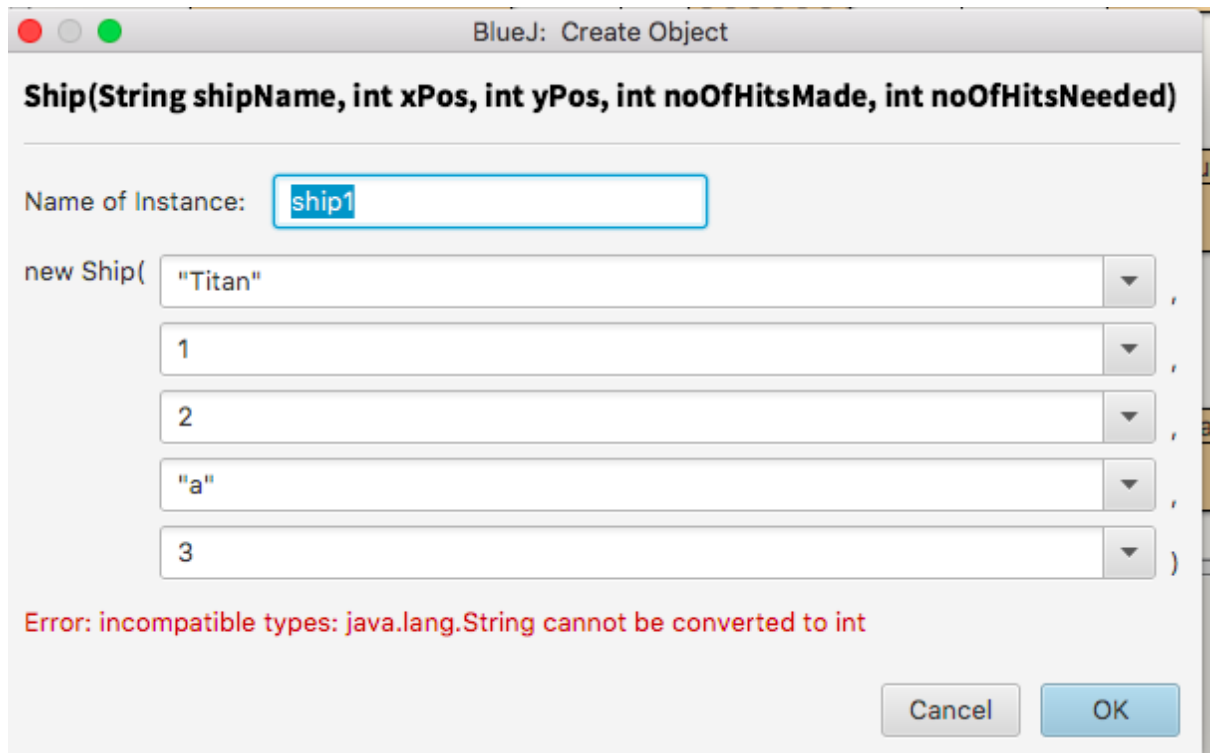
d.

Test Data:

- shipName: "Titan"
- xPos: 1
- yPos: 2
- noOfHitsMade: "a"
- noOfHitsNeeded: 4

Expected Results

- shipName: "Titan"
- xPos: 1
- yPos: 2
- noOfHitsMade: Error, Incompatible types
- noOfHitsNeeded: 4



The image shows a Java IDE window titled "BlueJ: Create Object". Inside, there is a dialog box for creating a new object of the class `Ship`. The dialog box has a title bar with standard Mac OS window controls (red, yellow, green buttons) and the title "BlueJ: Create Object". The main title of the dialog is **Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)**. Below the title, there is a field for "Name of Instance:" with the text "ship1" entered. Below this, there is a section for "new Ship(" followed by five input fields, each with a dropdown arrow. The first field contains "Titan", the second contains "1", the third contains "2", the fourth contains "a", and the fifth contains "3". The fields are separated by commas, and the last field is followed by a closing parenthesis. At the bottom of the dialog, there is a red error message: "Error: incompatible types: java.lang.String cannot be converted to int". At the bottom right, there are two buttons: "Cancel" and "OK".

BlueJ: Create Object

**Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)**

Name of Instance:

new Ship(  ,  
 ,  
 ,  
 ,  
 )

Error: incompatible types: java.lang.String cannot be converted to int

Cancel OK

e.

Test Data:

- shipName: "Titan"
- xPos: 1
- yPos: 2
- noOfHitsMade: 3
- noOfHitsNeeded: "a"

Expected Results

- shipName: "Titan"
- xPos: 1
- yPos: 2
- noOfHitsMade: 3
- noOfHitsNeeded: Error, Incompatible types

The image shows a BlueJ 'Create Object' dialog box. The title bar reads 'BlueJ: Create Object'. The main heading is 'Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)'. Below this, there is a field for 'Name of Instance:' with the value 'ship1'. Underneath, the constructor is being built: 'new Ship(' followed by five input fields. The first field contains '"Titan"', the second '1', the third '2', the fourth '3', and the fifth '"a"'. Each field has a dropdown arrow on its right. The fields are separated by commas, and the entire expression ends with a closing parenthesis ')'. At the bottom of the dialog, a red error message states: 'Error: incompatible types: java.lang.String cannot be converted to int'. At the very bottom right are 'Cancel' and 'OK' buttons.

BlueJ: Create Object

**Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)**

Name of Instance:

new Ship(  ,  
 ,  
 ,  
 ,  
 )

Error: incompatible types: java.lang.String cannot be converted to int

Cancel OK

### Test 3

#### a. getShipName()

Without setting any value

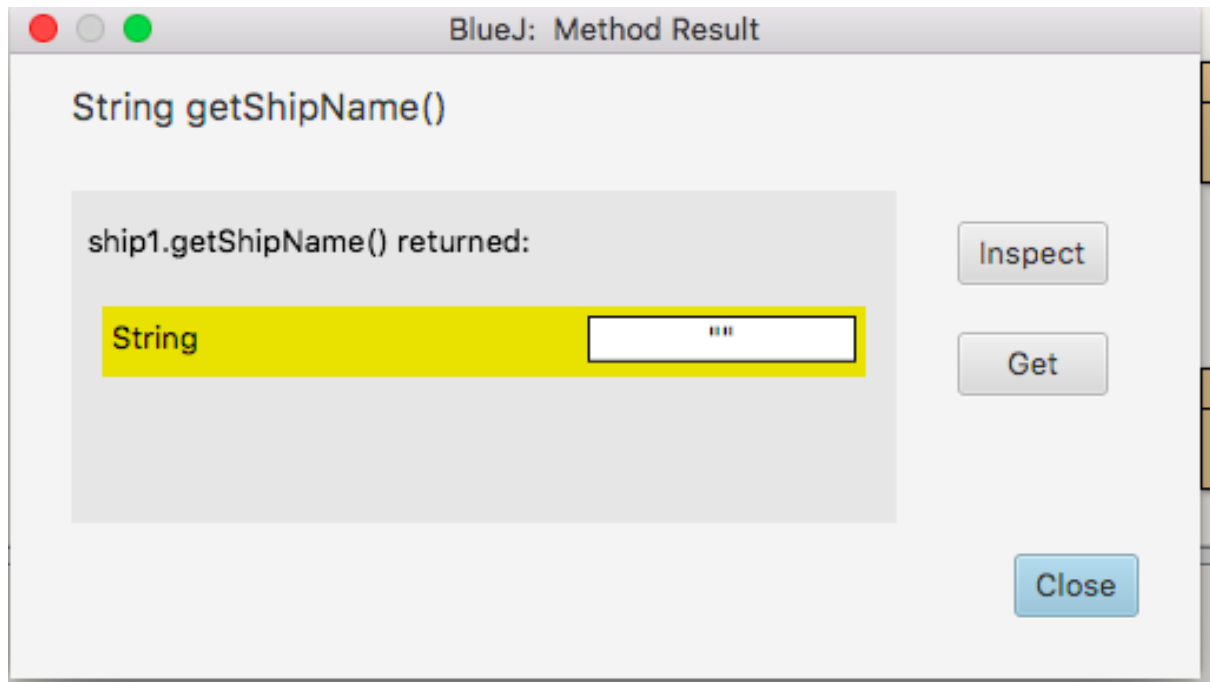
Test Data:

- No input

Expected Results:

- getShipName() = ""

Actual results:





**b. getXpos()**

Without setting any value

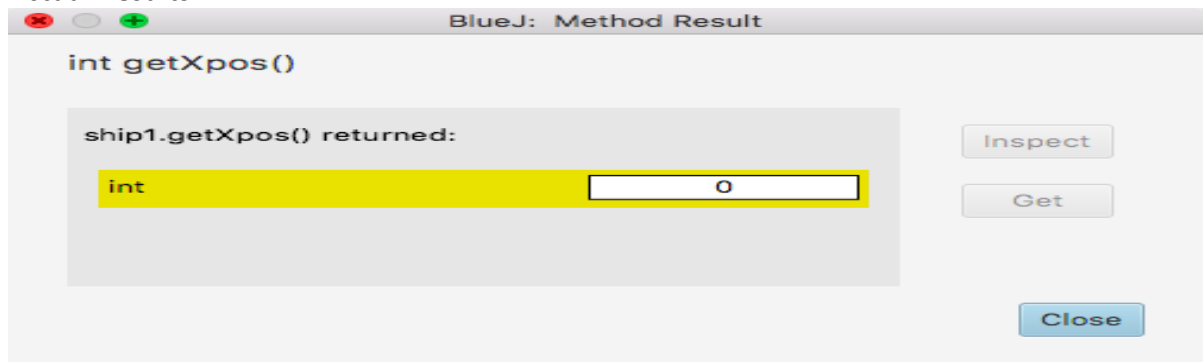
Test Data:

- No input

Expected Results:

- getXpos(): 0

Actual Results:



**c. getYpos()**

Without setting any value

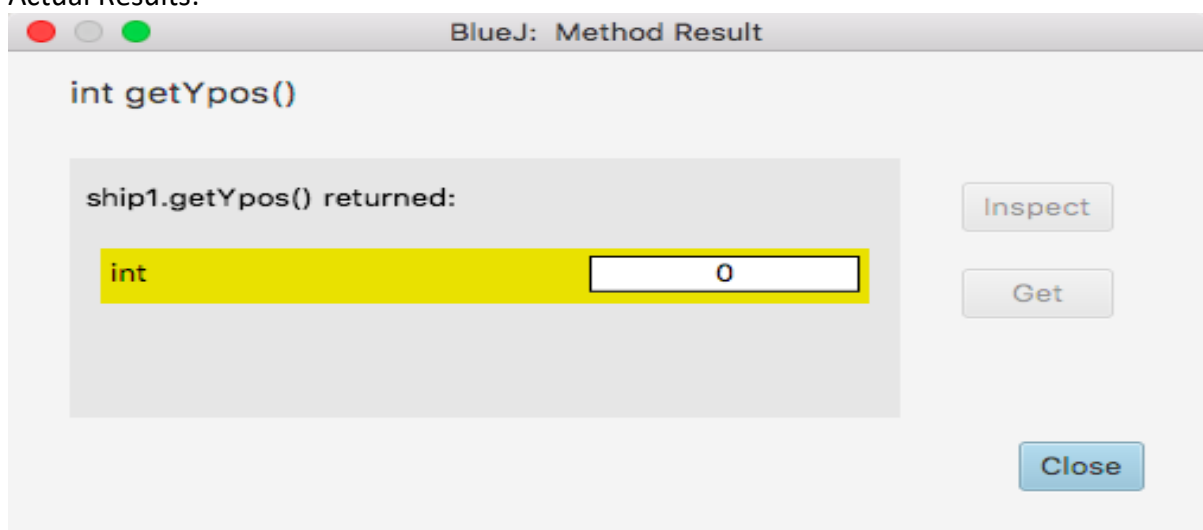
Test Data:

- No input

Expected Results:

- getYpos() : 0

Actual Results:



d. `getNoOfHitsMade()`

Without setting any value

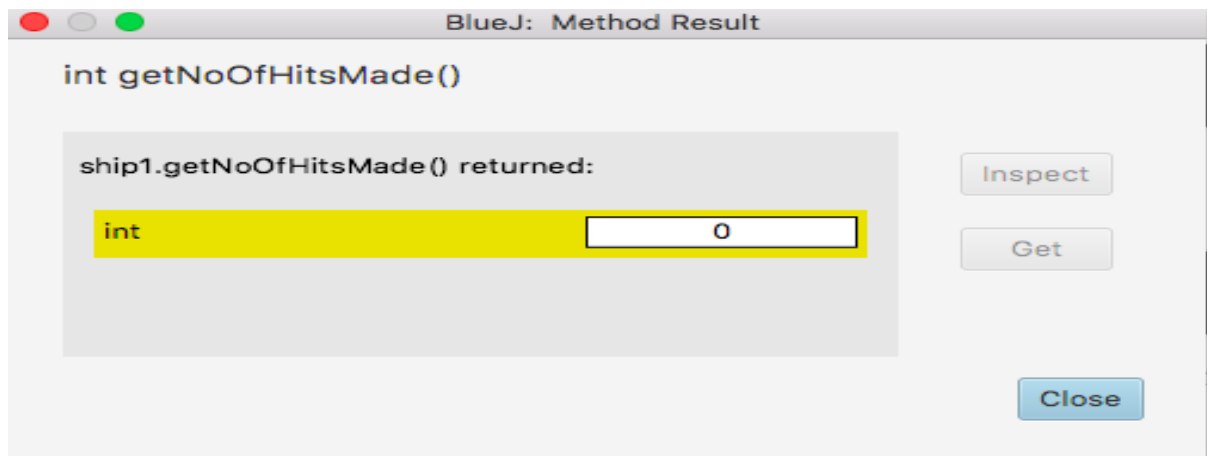
Test Data:

- No input

Expected Results:

- 0

Actual Result:



e. `getnoOfHitsNeeded()`

Without setting any value

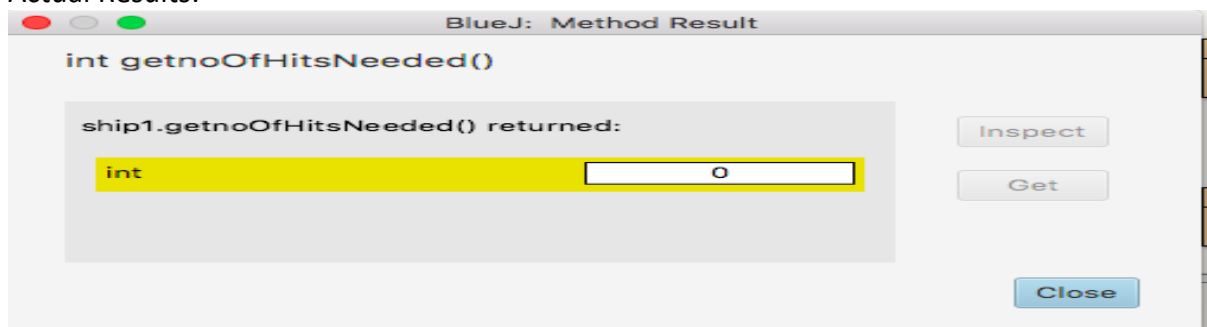
Test Data:

- No input

Expected Results:

- 0

Actual Results:



Test 4

a. setXpos()

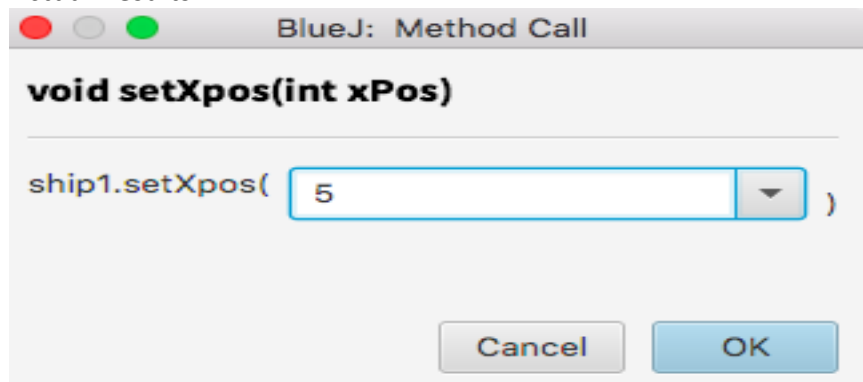
Test Data:

- 5

Expected Results:

- 5

Actual Results



b. setXpos()

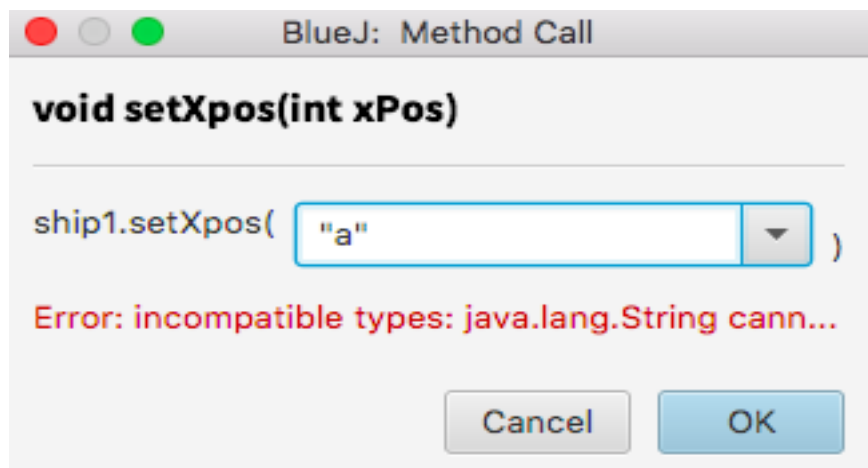
Test Data:

- "a"

Expected Results :

- Error, incompatible types

Actual Results



c. setYpos()

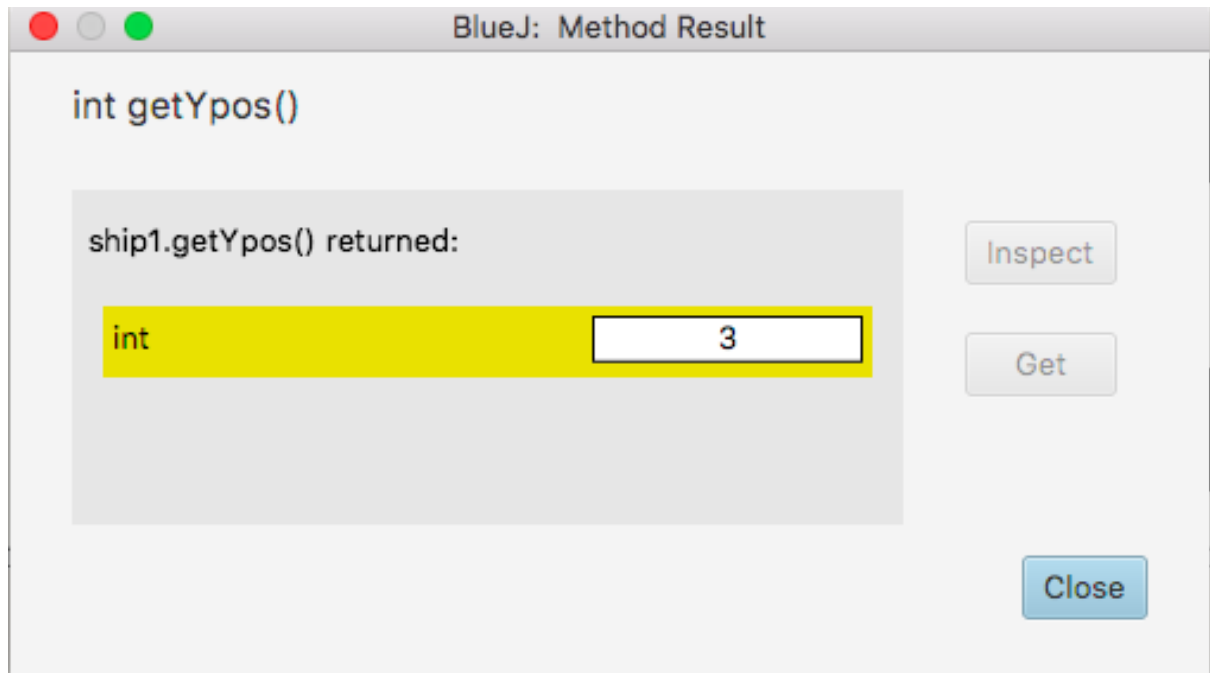
Test Data:

- 3

Expected Results:

- 3

Actual Results



d. setYpos()

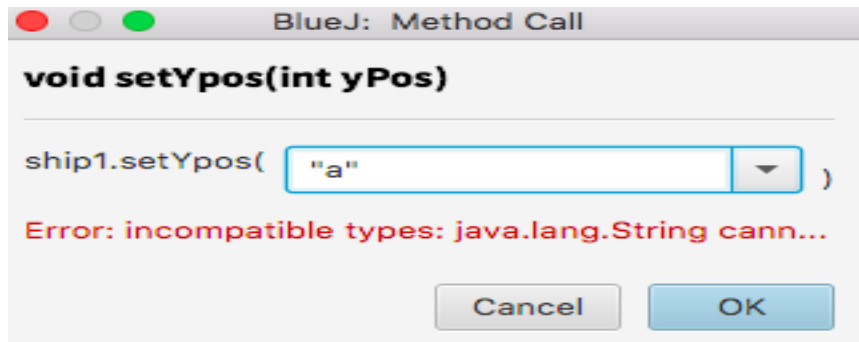
Test Data:

- "a"

Expected Results:

- Error, incompatible type

Actual Results



e. `setNoOfHitsMade()`

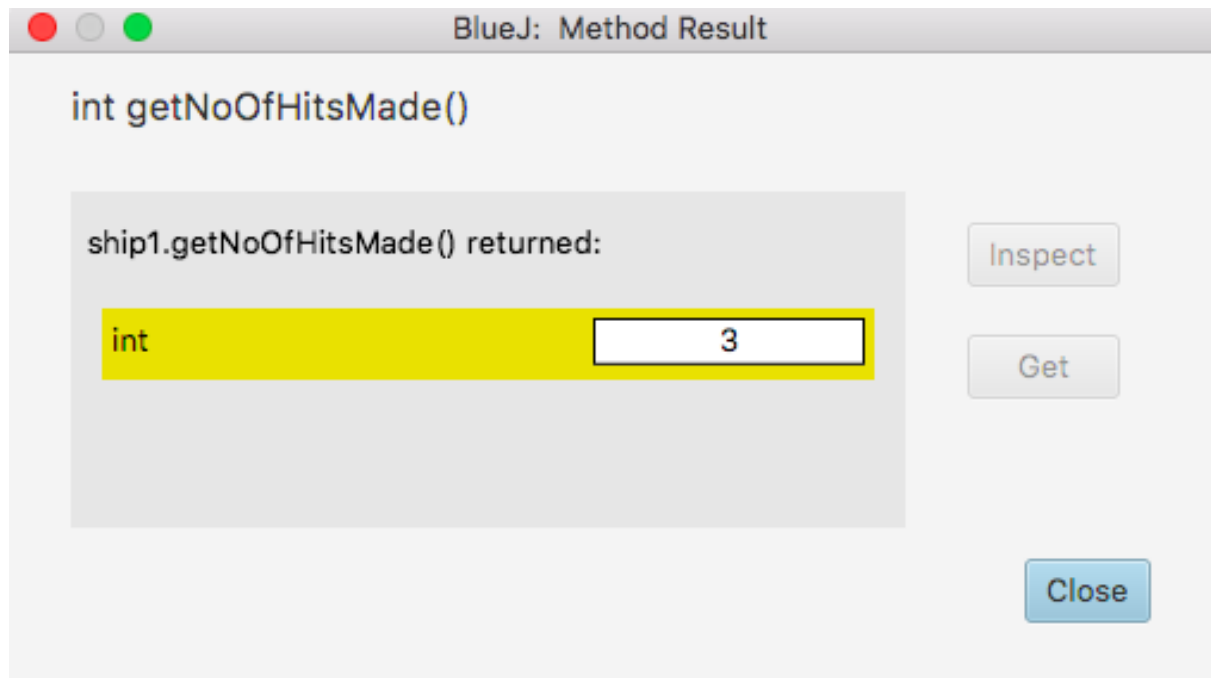
Test Data:

- 3

Expected Results:

- 3

Actual Results



f. `setNoOfHitsMade()`

Test Data:

- "a"

Expected Results:

- Error, incompatible type

Actual Results



g. setnoOfHitsNeeded()

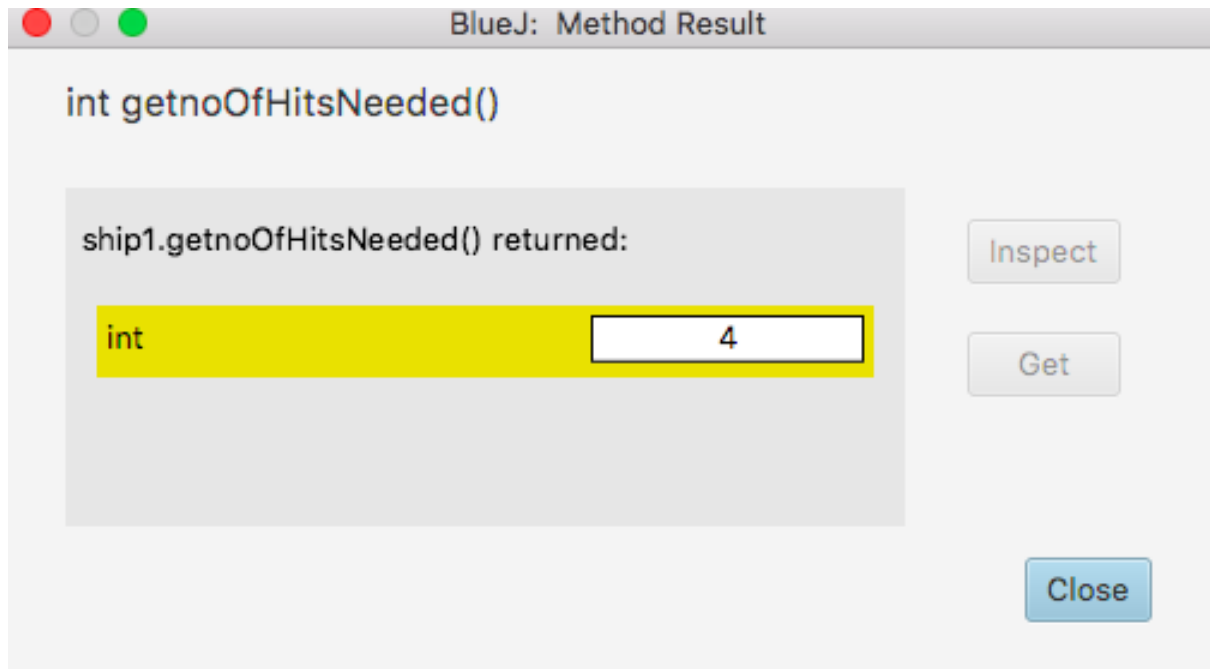
Test Data:

- 4

Expected Results:

- 4

Actual Results



h. setnoOfHitsNeeded

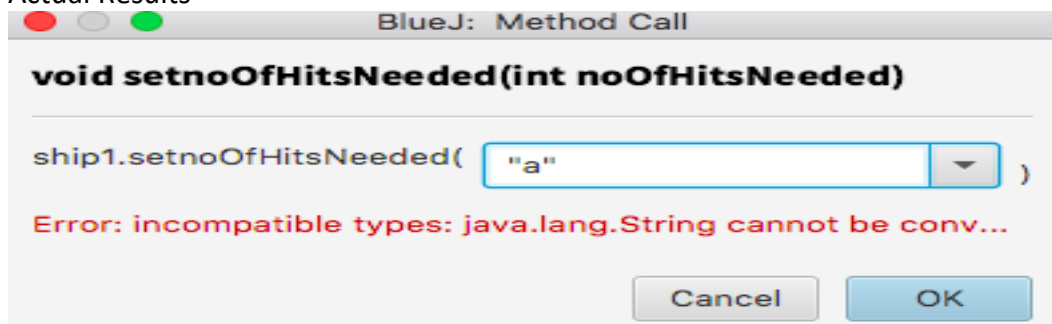
Test Data:

- "a"

Expected Results:

- Error, incompatible type

Actual Results



## Test 5

### a. display()

Created object using default constructor and then call display method.

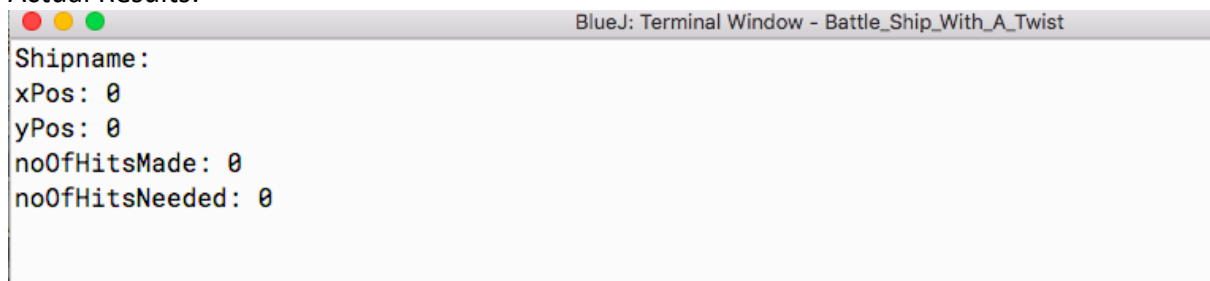
Test Data:

- No input

Expected Results:

- Shipname:
- xPos: 0
- yPos: 0
- noOfHitsMade: 0
- noOfHitsNeeded: 0

Actual Results:

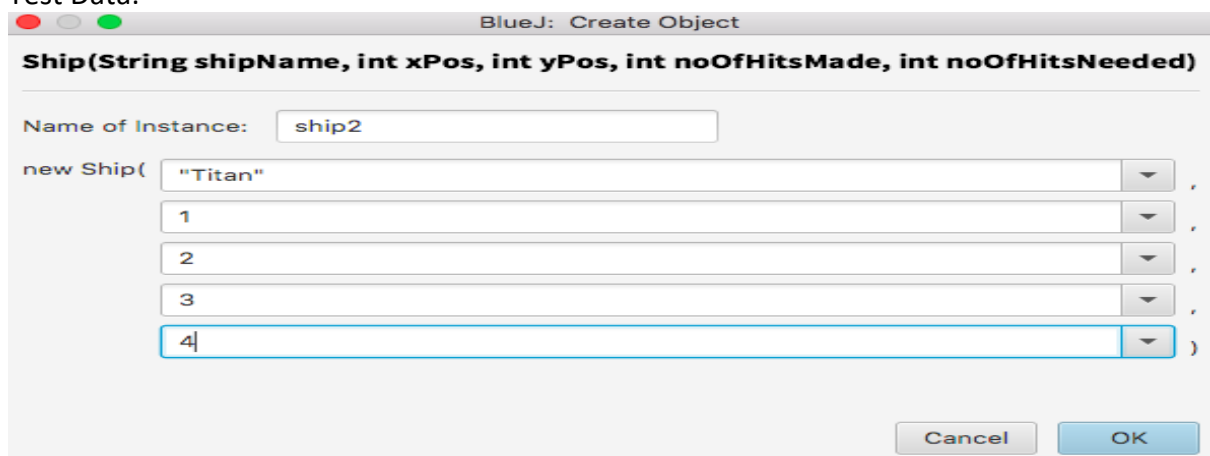


```
Shipname:
xPos: 0
yPos: 0
noOfHitsMade: 0
noOfHitsNeeded: 0
```

### b. display()

Created object using non- default constructor and then call display method.

Test Data:



BlueJ: Create Object

**Ship(String shipName, int xPos, int yPos, int noOfHitsMade, int noOfHitsNeeded)**

Name of Instance:

new Ship(  ,  
           ,  
           ,  
           ,  
           )


Cancel OK

Expected Results:

- Shipname: Titan
- xPos: 1

- yPos: 2
- noOfHitsMade: 3
- noOfHitsNeeded: 4

Actual Results:

A screenshot of a terminal window titled "BlueJ: Terminal Window - Battle\_Ship\_With\_A\_Twist". The window has a standard macOS-style title bar with red, yellow, and green window control buttons. The terminal content displays the following text:

```
Shipname: Titan  
xPos: 1  
yPos: 2  
noOfHitsMade: 3  
noOfHitsNeeded: 4
```