**Cave Class Test Strategy**

**Test 1**

Create a Cave object with the default constructor.

**Test data:**

* Default constructor with no parameters

**Expected results:**

* id: 0
* north: 0
* east: 0
* south: 0
* west: 0
* creature: null

**Actual results:**

id : 0

north : 0

east : 0

south : 0

west : 0

creature : null

✅ Test Passed!

**Test 2**

Create a Cave object with the non-default constructor using valid field values.

**Test data:**

* id: 5
* north: 10
* east: 15
* south: 0
* west: 100 (Mount Api)

**Expected results:**

* id: 5
* north: 10
* east: 15
* south: 0
* west: 100
* creature: null

**Actual results:**

id : 5

north : 10

east : 15

south : 0

west : 100

creature : null

✅ Test Passed!

**Test 3**

Test the setter methods of the Cave class with valid values.

**Test data:**

* setId(10)
* setNorth(20)
* setEast(30)
* setSouth(40)
* setWest(50)

**Expected results:**

* id: 10
* north: 20
* east: 30
* south: 40
* west: 50

**Actual results:**

id : 10

north : 20

east : 30

south : 40

west : 50

✅ Test Passed!

**Test 4**

Test the setter methods of the Cave class with invalid values.

**Test data:**

* Set valid values first:
  + setId(10)
  + setNorth(20)
  + setEast(30)
  + setSouth(40)
  + setWest(50)
* Then set invalid values:
  + setId(-5)
  + setNorth(-10)
  + setEast(-20)
  + setSouth(-30)
  + setWest(-40)

**Expected results:**

* id: 10 (unchanged)
* north: 20 (unchanged)
* east: 30 (unchanged)
* south: 40 (unchanged)
* west: 50 (unchanged)

**Actual results:**

id : 10

north : 20

east : 30

south : 40

west : 50

✅ Test Passed!

**Test 5.1**

Test the hasExitToMountApi method with no exit to Mount Api.

**Test data:**

* Cave with no connection to Mount Api (no value of 100)
* id: 1, north: 2, east: 3, south: 4, west: 5

**Expected results:**

* hasExitToMountApi(): false

**Actual results:**

hasExitToMountApi(): false

✅ Test Passed!

**Test 5.2**

Test the hasExitToMountApi method with north exit to Mount Api.

**Test data:**

* Cave with north connection to Mount Api (north = 100)
* id: 2, north: 100, east: 3, south: 4, west: 5

**Expected results:**

* hasExitToMountApi(): true

**Actual results:**

hasExitToMountApi(): true

✅ Test Passed!

**Test 5.3**

Test the hasExitToMountApi method with east exit to Mount Api.

**Test data:**

* Cave with east connection to Mount Api (east = 100)
* id: 3, north: 1, east: 100, south: 4, west: 5

**Expected results:**

* hasExitToMountApi(): true

**Actual results:**

hasExitToMountApi(): true ✅ Test Passed!

**Test 5.4**

Test the hasExitToMountApi method with south exit to Mount Api.

**Test data:**

* Cave with south connection to Mount Api (south = 100)
* id: 4, north: 1, east: 2, south: 100, west: 5

**Expected results:**

* hasExitToMountApi(): true

**Actual results:**

hasExitToMountApi(): true

✅ Test Passed!

**Test 5.5**

Test the hasExitToMountApi method with west exit to Mount Api.

**Test data:**

* Cave with west connection to Mount Api (west = 100)
* id: 5, north: 1, east: 2, south: 3, west: 100

**Expected results:**

* hasExitToMountApi(): true

**Actual results:**

hasExitToMountApi(): true

✅ Test Passed!

**Test 6.1**

Test adding a creature to a cave.

**Test data:**

* Create a cave: id: 1, north: 2, east: 3, south: 4, west: 5
* Add an Orc named "TestOrc"

**Expected results:**

* getCreature() returns the Orc object
* getCreature().getName() returns "TestOrc"

**Actual results:**

getCreature() type: Orc

getCreature().getName(): TestOrc

✅ Test Passed!

**Test 6.2**

Test changing a creature in a cave.

**Test data:**

* Using the same cave from Test 6.1
* Change creature to a Troll named "TestTroll"

**Expected results:**

* getCreature() returns the Troll object
* getCreature().getName() returns "TestTroll"

**Actual results:**

getCreature() type: Troll

getCreature().getName(): TestTroll

✅ Test Passed!

**Test 6.3**

Test removing a creature from a cave.

**Test data:**

* Using the same cave from Test 6.2
* Remove creature by setting it to null

**Expected results:**

* getCreature() returns null

**Actual results:**

getCreature(): null

✅ Test Passed!

**Test 7.1**

Test toString method with an empty cave.

**Test data:**

* Create a cave with no connections: id: 1, north: 0, east: 0, south: 0, west: 0

**Expected results:**

* toString() contains "Cave 1"
* No direction information since there are no connections

**Actual results:**

toString(): Cave 1:

✅ Test Passed!

**Test 7.2**

Test toString method with a cave with connections.

**Test data:**

* Create a cave with connections: id: 2, north: 3, east: 0, south: 5, west: 100

**Expected results:**

* toString() contains "Cave 2"
* toString() contains "North → Cave 3"
* toString() contains "South → Cave 5"
* toString() contains "West → Mount Api"

**Actual results:**

toString(): Cave 2: North → Cave 3, South → Cave 5, West → Mount Api

✅ Test Passed!

**Test 7.3**

Test toString method with a cave containing a creature.

**Test data:**

* Create a cave: id: 3, north: 0, east: 4, south: 0, west: 2
* Add a Goblin named "TestGoblin"

**Expected results:**

* toString() contains "Cave 3"
* toString() contains "East → Cave 4"
* toString() contains "West → Cave 2"
* toString() contains "TestGoblin"
* toString() contains "Goblin"

**Actual results:**

toString(): Cave 3: East → Cave 4, West → Cave 2 [Contains: TestGoblin (Goblin)]

✅ Test Passed!

I also have a class, which uses Java assertions to systematically verify the functionality of the Cave component. Assertions were chosen for thorough testing because they:

* Provide clear, concise test cases with descriptive error messages
* Can be easily enabled or disabled using the Java -ea flag when running it
* Identify failures immediately at the exact point where tests fail, which reduces debugging time
* Integrate directly with the code, making tests self-documenting
* Require no external testing frameworks or dependencies as the assertion feature is built directly into the Java language

**Results of class:**

=== Cave Class Test Suite ===

--- Testing Constructor and Getters ---

Default constructor - Cave ID: 0

Non-default constructor - Cave ID: 5

Constructor and getter tests passed.

--- Testing Setters ---

Set ID: 10

After setting negative ID: 10

Set north: 20

Set east: 30

Set south: 40

Set west: 50

After setting negative directions - North: 20, East: 30, South: 40, West: 50

Setter tests passed.

--- Testing Creature Operations ---

Set creature: TestOrc

Changed creature: TestTroll

Removed creature

Creature operations tests passed.

--- Testing hasExitToMountApi ---

Cave with no exit to Mount Api: false

Cave with north exit to Mount Api: true

Cave with east exit to Mount Api: true

Cave with south exit to Mount Api: true

Cave with west exit to Mount Api: true

hasExitToMountApi tests passed.

--- Testing toString ---

Empty cave toString: Cave 1:

Connected cave toString: Cave 2: North → Cave 3, South → Cave 5, West → Mount Api

Cave with creature toString: Cave 3: East → Cave 4, West → Cave 2 [Contains: TestGoblin (Goblin)]

toString tests passed.

All tests completed.

It does provide more error messaging if tests fail but since all the tests passed, it only prints what was explicitly told to print through the statements. When an assertion fails in Java, it throws an AssertionError with an optional message, for example:

assert cave1.getId() == 0 : "Default ID should be 0";

Each test contains these messages, which will appear in the event a test fails.