**AirLock Master Test Plan**

**Please Note:** this is not a ‘master test plan’ as asked for in Part A. It contains no analysis of risks, and no prioritisation of test areas – specifically static reviews of requirements and specifications, design docs, or any planning related to user acceptance testing.

Having said that, the most critical risk in an airlock system is that a spacecraft suffers a catastrophic loss of pressure to the interior of the spacecraft. Even stranding someone outside is preferrable to killing everyone inside the spacecraft. So – Sorry Dave.

This ‘test plan’ document specifies what is desired in regard to unit, integration, system operation, scenario testing and functional testing of the AirLock system.

It is suggested that you implement and unit test classes in parallel. I.E. either write unit tests for a method and then implement the method to satisfy the unit tests (as in Test Driven Development), or write the method, and then implement the unit tests to ensure it meets the specifications.

In the following section, for each class, methods to be tested are listed in priority order. It is left to you to decide if, and how much to test each method, according to your assessment of what is possible and desirable within the constraints of submitting the assignment by the due date.

Remember it is just as important to test that classes work correctly when error conditions occur, as that they work correctly during normal operation.

Unit Testing

1. PressureSensor.
   1. constructor
      1. ensure that the constructor throws an ‘InvalidPressureException’ if an attempt is made to initialise the sensor with a negative pressure (< 0.0)
      2. ensure that the constructor returns a valid fully initialised PressureSensor object. (i.e. no other methods should fail unexpectedly or give unspecified results if the constructor is successful)
   2. getPressure
      1. ensure that getPressure returns the initial pressure set in the constructor
      2. ensure that getPressure returns any updated pressure set by setPressure
   3. setPressure
      1. ensure that setPressure throws an ‘InvalidPressureException’ if an attempt is made to set a negative pressure (< 0.0)
      2. ensure that setPressure updates the initial pressure set by the constructor

The aim of testing PressureSensor is to make sure that the sensor returns either the initial pressure or the pressure most recently set.

Integration Testing

You are not asked to perform isolated unit testing for Door or AirLock. Instead, you are asked to perform integration testing both for Door-PressureSensor and AirLock-Door-PressureSensor.

The aim of integration testing is to ensure that Pressure sensors work correctly with Doors, and that both Doors and PressureSensors work correctly with AirLocks.

1. Door
   1. Constructor
      1. Ensure that the constructor throws a DoorException if the supplied sensors are not valid implementations of the IPressureSensor interface.
      2. Ensure that the constructor throws an DoorException if the initial Door state is set to OPEN but the difference between internal and external pressures is greater than TOLERANCE.
      3. Ensure that if no exceptions are thrown, that the constructor returns a valid fully initialised Door. (i.e. no other methods should fail unexpectedly or give unspecified results if the constructor is successful)
   2. open
      1. ensure that openDoor throws a DoorException if open() is called when the Door is already open
      2. ensure that openDoor throws an DoorException if the Door is closed and the difference between external and internal pressures is greater than TOLERANCE
      3. ensure that if no exceptions are thrown that the door is opened (i.e. that the Door’s state becomes OPEN)
   3. close
      1. ensure that closeDoor throws a DoorException if close() is called when the Door is already closed
      2. ensure that if no exception is thrown that the Door’s state becomes CLOSED
   4. getExternalPressure
      1. Ensure that getExternalPressure returns the correct pressure value associated with the external pressure sensor set by the constructor
   5. getInternalPressure
      1. Ensure that getInternalPressure returns the correct pressure value associated with the internal pressure sensor set by the constructor
   6. isOpen
      1. Ensure that isOpen returns true when the Door is OPEN and false when it is CLOSED
   7. isClosed
      1. Ensure that isClosed returns true when the Door is CLOSED and false when it is OPEN

The aim of testing Door is to make sure that the door does not open if internal and external pressures are unequal but does open if they are equal. ‘Internal’ is always towards the inside of the airlock. ‘External’ for the inner door refers to the interior of the spacecraft. ‘External’ for the outer door refers to the exterior of the spacecraft.

1. AirLock
   1. Constructor
      1. Ensure that a valid fully initialised AirLock is returned. (Both airlock state and operation mode are initialised correctly)
      2. Ensure that initial airlock state is set to SEALED if both doors are CLOSED, and otherwise UNSEALED
      3. Ensure that initial operational mode is set to MANUAL.
   2. openOuterDoor
      1. Ensure that openOuterDoor throws an AirLockException if openOuterDoor is called while the outer door is already open.
      2. Ensure that if operation mode is AUTO and the inner door is open then an attempt is made to close the inner door
      3. Ensure that if operation mode is AUTO and after the inner door is closed then an attempt is made to equalise pressures with the external environment
      4. Ensure that if operation mode is AUTO and after the inner door is closed and pressure has been equalised with the external environment that an attempt is made to open the outer door
      5. Ensure that if operation mode is MANUAL then an attempt is made to open the outer door
      6. Ensure that if no ex exceptions are thrown that the airlock state becomes UNSEALED
      7. Ensure that if any exceptions are thrown and the airlock was SEALED when openOuterDoor was called, that the airlock remains SEALED.
      8. Ensure that all DoorExceptions are caught and then rethrown encapsulated in AirLockExceptions
   3. openInnerDoor
      1. Ensure that openInnerDoor throws an AirLockException if openInnerDoor is called while the inner door is already open.
      2. Ensure that if operation mode is AUTO and the outer door is open then an attempt is made to close the outer door
      3. Ensure that if operation mode is AUTO and after the outer door is closed then an attempt is made to equalise pressures with the internal cabin
      4. Ensure that if operation mode is AUTO and after the outer door is closed and pressure has been equalised with the internal cabin that an attempt is made to open the inner door
      5. Ensure that if operation mode is MANUAL then an attempt is made to open the inner door
      6. Ensure that if no ex exceptions are thrown that the airlock state becomes UNSEALED
      7. Ensure that if any exceptions are thrown and the airlock was SEALED when openInnerDoor was called, that the airlock remains SEALED.
      8. Ensure that all DoorExceptions are caught and then rethrown encapsulated in AirLockExceptions
   4. closeOuterDoor
      1. Ensure that closeOuterDoor attempts to close the outer door
      2. Ensure that if the inner door is also closed, closeOuterDoor sets the airlock state to SEALED
      3. Ensure that all DoorExceptions are caught and then rethrown encapsulated in AirLockExceptions
   5. closeInnerDoor
      1. Ensure that closeInnerDoor attempts to close the inner door
      2. Ensure that if the outer door is also closed, closeInnerDoor sets the airlock state to SEALED
      3. Ensure that all DoorExceptions are caught and then rethrown encapsulated in AirLockExceptions
   6. equaliseWithEnvironmentPressure
      1. ensure that equaliseWithEnvironmentPressure throws an ‘AirlockNotSealedException’ if it is called while the airlock state is not SEALED
      2. ensure that if no exception is thrown that the lock pressure is set to the same as the exterior environment pressure (outer doors external pressure)
   7. equaliseWithCabinPressure
      1. ensure that equaliseWithCabinPressure throws an ‘AirlockNotSealedException’ if it is called while the airlock state is not SEALED
      2. ensure that if no exception is thrown that the lock pressure is set to the same as the interior cabinpressure (inner doors external pressure)
   8. toggleOperationMode
      1. ensure that toggleOperationModethrows an ‘AirLockException’ if it is called while the airlock is not SEALED
      2. ensure that if no exception is thrown that the airlock’s operational mode is switched. (from MANUAL to AUTO or AUTO to MANUAL)
   9. isInManualMode
      1. ensure that isInManualMode returns TRUE if the airlocks operational mode is MANUAL and false if it in AUTO operation mode
   10. isInAutoMode
       1. ensure that isInAutoMode returns TRUE if the airlocks operational mode is AUTO and false if it in MANUAL operation mode
   11. isOuterDoorClosed
       1. ensure that isOuterDoorClosed returns true if the outer door is closed and false if it is open
   12. isInnerDoorClosed
       1. ensure that isInnerDoorClosed returns true if the outer door is closed and false if it is open

The aim of testing for AirLock is to ensure that under no circumstances do both doors open when exterior environment and interior cabin pressures are unequal.

In AUTO mode, both doors should never be simultaneously open.

Toggling between Auto and Manual mode should only be possible when both doors are shut.

System Level Testing (Functional Acceptance Tests)

For system operation testing, you must write Functional Acceptance Tests (FATs) that set up different starting conditions from which to test the airlock’s functionality in both MANUAL and AUTO mode in a variety of scenarios.

Scenario testing puts system operations together in a sequence to enact a use case such as ‘enter spacecraft’ or ‘exit spacecraft’.

Required test scenarios:

1. Pass through airlock in auto mode from inside to outside when external environment pressure is less than internal cabin pressure
2. Pass through airlock in auto mode from outside to inside when external environment pressure is greater than internal cabin pressure
3. Pass through airlock from inside to outside in manual mode when external environment pressure is greater than internal cabin pressure
4. Pass through airlock from outside to inside in manual mode when external environment pressure is less than internal cabin pressure

Steps involved in transiting the airlock in auto mode from outside through to the cabin for all initial pressure settings with both doors closed are as follows:

1. Open the outer door
2. Open the inner door
3. Close the inner door

At the end of this process, the airlock should be SEALED, the cabin pressure should be unchanged, and the airlock pressure should be the same as the cabin pressure.

Steps involved in transiting the airlock in manual mode from outside through to the cabin for all initial pressure settings with both doors closed are as follows:

1. Equalise lock pressure with external environment pressure
2. Open the outer door
3. Close the outer door
4. Equalise lock pressure with internal cabin pressure
5. Open the inner door
6. Close the inner door

At the end of this process, the airlock should be SEALED, the cabin pressure should be unchanged, and the airlock pressure should be the same as the cabin pressure.