EECS 3311W20 - SIMODYSSEY2 REPORT

Team Member	Name	EECS Login	Signature
Member 1:	Christopher Boyd	chris360	

1. Requirements for SimOdyssey2

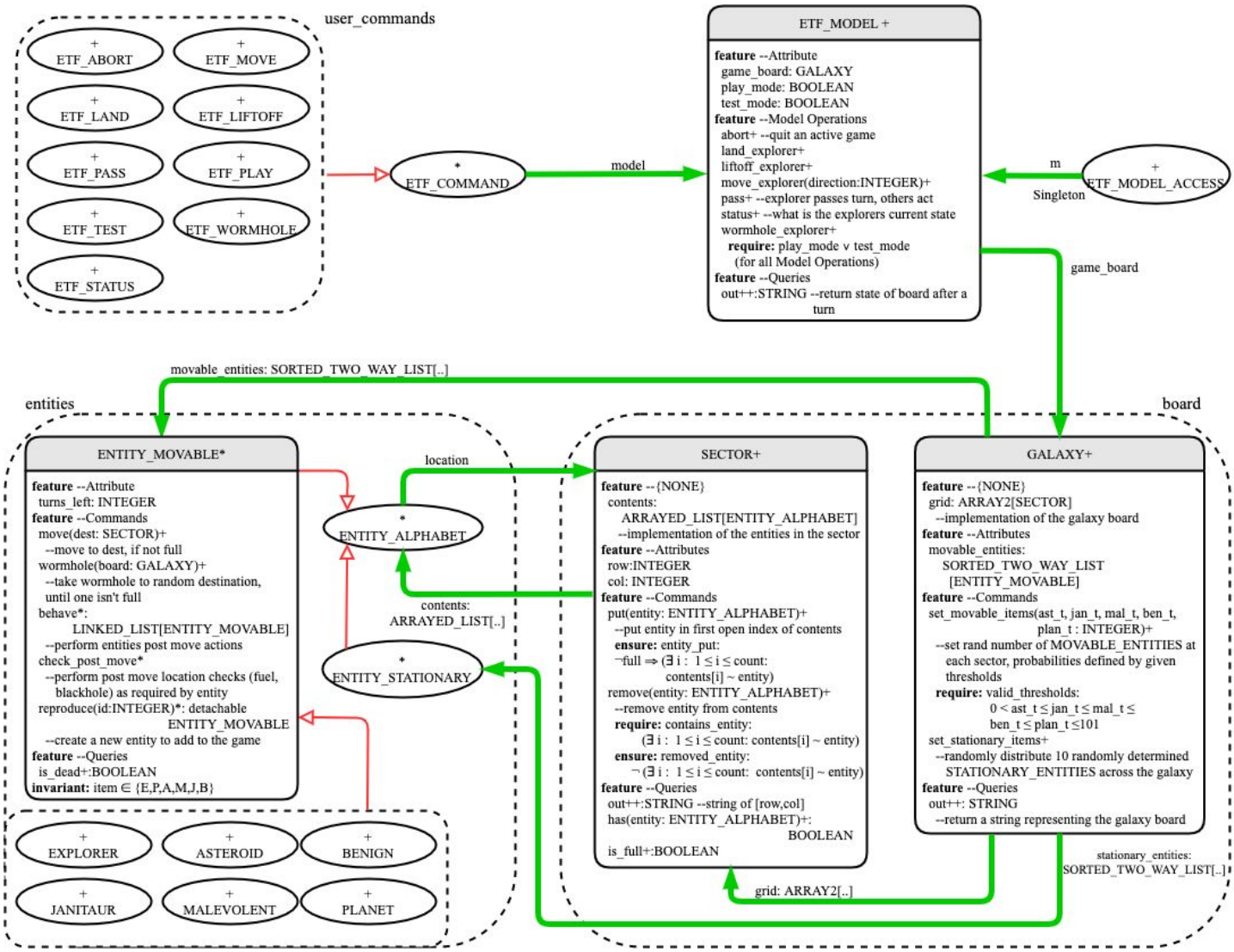
To create the business logic for a game titled “SimOdyssey2”, a follow up to the basic but beloved SimOdyssey. The premise of the game is that a future Earth requires a simulator to train a new generation of space explorers. SimOdyssey began this process with a training module that allowed the explorer to seek out new worlds in a relatively empty and safe galaxy. SimOdyssey2 must build upon this foundation but introduces the explorer to new entities and dangers in the galaxy.

Given a base set of galactic entities and a standardized sequence for how they will act, my task was to implement these entities along with creating a design that allows for the easy addition of new entities. These new interactive entities all follow a basic sequence of: move, check destination for other entities, potentially reproduce, followed by performing some entity-specific behaviours. SimOdyssey2 implements the entities Benign, Malevolent, Janitaur and Asteroid not seen in SimOdyssey (see section 7.2 for details). The key design goal of SimOdyssey2 was to enable expansion of the entities within the game with minimal effort through a generalized pattern for entities to act within and communicate with each other.

At this stage, the goal of SimOdyssey2 was to create testable game logic. As such, the game interaction is implemented through a command line interface that can be run in either interactive or batch-mode to allow for thorough testing. This interaction is defined by a given grammar for the user interface (see section 7.3). This command interaction can be easily incorporated into any graphical interface in the future, without requiring any changes to the developed and tested game logic.

2. Architecture

2.1. BON Diagram



2.2. Design Overview

2.2.1. Abstraction of Interface

As the goal of this design is to develop and test the game logic for SimOdyssey2, not to implement the final interface, the interface must be abstracted away from game logic decisions. The user_commands cluster accomplishes this. In future development an immersive graphical environment need only use this interface and benefit from the existing, proven, game logic.

2.2.2. Extendable Entities

The importance of expanding the type and quantity of entities represented within the SimOdyssey2 game was demonstrated by the growth from the initial SimOdyssey. The planned growth for future versions has been implemented by defining a sequence of action types (move, check, reproduce and behave) within ENTITY_MOVABLE. By inheriting from this base class the system will support any future entities that are required to better train space explorers.

2.2.3. Information Hiding

Within the board cluster, both SECTOR and GALAXY classes have the current implementation of their key systems hidden from the client's view of the class. The GALAXY is implemented using a 2-dimensional array of SECTORS which, in turn, is implemented by an ARRAYED_LIST of ENTITY_ALPHABET. The implementation of both GALAXY and SECTOR can be modified without any effect on clients making use of either class.

2.2.4. Reliability

To ensure stability, Eiffel's Design by Contract system has been leveraged to prevent invalid inputs from corrupting data. This can be seen throughout the project by performing pre and post command checks to guarantee validity of the design.

2.2.5. Simplicity

A simple system is an easy system to maintain. To increase maintainability, each class supports a minimum set of features that are relevant only for that class to function. For example, GALAXY is composed of a grid of SECTORS which, in turn, each hold a set of ENTITY_ALPHABET. Therefore, GALAXY does not deal with the mechanics of moving said entities about the SECTORs. When required, such as in the out feature, GALAXY need only request the detailed information regarding each SECTOR for specific details about it's contents.

3. Table of Modules

3.1. Model Interface

1	ETF_MODEL_ACCESS	Responsibility: Provide singleton access to the ETF_MODEL class	Alternative: N/A
	Concrete	Secret: None	

1.1	ETF_MODEL	Responsibility: Primary game logic. Use GALAXY to initialize the game. Receives input commands and builds output STRING that the user interacts with. Initiates move, check, reproduce and behave commands of ENTITY_MOVABLES.	Alternative: N/A
	Concrete	Secret: Methods related to game logic that are used internally to ETF_MODEL and are not called by client classes. Also several internal attributes related to the game state and a list of ENTITY_MOVABLE that died during the previous turn.	

3.2. Board Cluster

2	board	Responsibility: Set of classes related to the storage implementation of game items.	Alternative: N/A
	Cluster	Secret: N/A	

2.1	GALAXY	Responsibility: board game creation and initial placement of all entities within SECTORs. Build STRING representing current board state.	Alternative: Implement with dynamic allocation when SECTOR becomes non-empty
	Concrete	Secret: board grid implemented with an ARRAY2[SECTOR]. See 1.1.1	

2.1.1	SECTOR	Responsibility: Store ENTITY_ALPHABET. Provides an interface to add/remove ENTITY_ALPHABET. Query types of entities in the SECTOR. Build STRING representing contents of the SECTOR.	Alternative: Use place-holder entities instead of detachable
	Concrete	Secret: Contents implemented with an ARRAYED_LIST[detachable ENTITY_ALPHABET]	

3.3. Entities Cluster

3	entities	Responsibility: Set of classes related to the entities stored in each SECTOR.	Alternative: N/A
	Cluster	Secret: N/A	

3.1	ENTITY_ALPHABET	Responsibility: Defines base requirements for entities in the game: id (unique INTEGER), item (set of CHARACTER), and location (SECTOR). Implements equality and less than for comparing and sorting any descendent classes.	Alternative: N/A
	Abstract	Secret: None	

3.1.1	ENTITY_MOVABLE	Responsibility: Provides base requirements for entities that can travel and act in the game. Defines deferred method interfaces for: check_post_move, reproduce and behave.	Alternative: Further abstract into more subclasses, such as uses fuel or not
	Abstract	Secret: None	

3.1.1.1	ASTEROID	Responsibility: Implement ASTEROID, which randomly travels the galaxy, does not reproduce and behaves by seeking out other ENTITY_MOVABLEs to destroy. Represented by an ‘A’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.1.2	BENIGN	Responsibility: Implement BENIGN, which randomly travels the galaxy. Reproduces every other turn. Behaviour is to protect EXPLORER from MALEVOLENT and destroy MALEVOLENT within current SECTOR. Represented by a ‘B’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.1.3	EXPLORER	Responsibility: Implement EXPLORER, which is the user controlled entity. Moves around the galaxy seeking out planets to land on and explore for life. If life is found the game is won and over. Represented by an ‘E’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.1.4	JANITAUR	Responsibility: Implement JANITAUR, which randomly travels the galaxy. Reproduces every third turn. Behaviour is to remove ASTEROIDs from the galaxy by transporting them into a WORMHOLE, where they are destroyed. Represented by a ‘J’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.1.5	MALEVOLENT	Responsibility: Implement MALEVOLENT, which randomly travels the galaxy. Reproduces every other turn. Behaviour is to attack the EXPLORER if there are no BENIGNs present and the EXPLORER is not landed. Each attack takes 1 life from EXPLORER. Represented by a ‘M’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.1.6	PLANET	Responsibility: Implement PLANET, which randomly travels the galaxy. Does not reproduce. Behaviour is to orbit a star if there is one in the SECTOR. Once orbiting, it will not move again. If the PLANET is orbiting a YELLOW_DWARF, it has a 50% chance of supporting life. Represented by a ‘P’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.2	ENTITY_STATIONARY	Responsibility: Provides base requirements for entities that do not move once placed in a SECTOR. Each SECTOR can only have one ENTITY_STATIONARY.	Alternative: Further abstract into more subclasses.
	Abstract	Secret: None	

3.1.2.1	STAR	Responsibility: Provides base requirements for entities that have a luminosity and provide fuel to entities that require it.	Alternative: N/A
	Abstract	Secret: None	

3.1.2.1.1	YELLOW_DWARF	Responsibility: Implements a STAR with luminosity of 2. Represented by a ‘Y’ on the board	Alternative: N/A
	Concrete	Secret: None	

3.1.2.1.2	BLUE_GIANT	Responsibility: Implements a STAR with luminosity of 5. Represented by a ‘*’ on the board.	Alternative: N/A
	Concrete	Secret: None	

3.1.2.2	BLACKHOLE	Responsibility: Implements a BLACKHOLE, which kills every entity that enters it’s SECTOR.	Alternative: N/A
	Concrete	Secret: None	

3.1.2.3	WORMHOLE	Responsibility: Implements a WORMHOLE, ENTITY_MOVABLEs can enter a WORMHOLE from a SECTOR to any randomly determined SECTOR without using fuel	Alternative: N/A
	Concrete	Secret: None	

3.4. Support Cluster

4	support	Responsibility: Set of classes that provide services such as default values to multiple classes within the system.	Alternative: N/A
	Cluster	Secret: N/A	

4.1	RANDOM_GENERATOR_ACCESS	Responsibility: Provides singleton access to RANDOM_GENERATOR	Alternative: N/A
	Concrete	Secret: None	

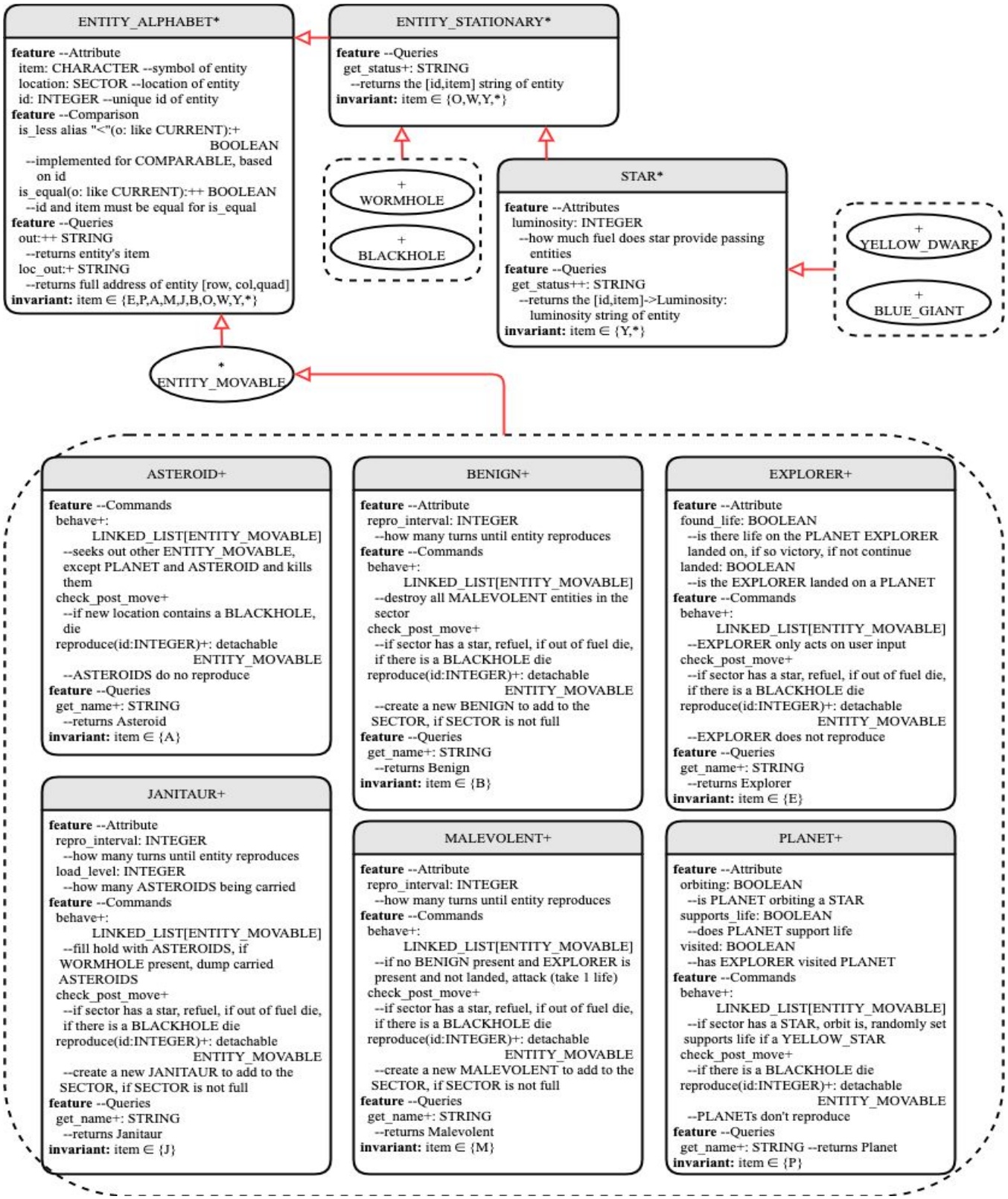
4.1.1	RANDOM_GENERATOR	Responsibility: Provides consistent number generation for testing game logic	Alternative: N/A
	Concrete	Secret: n_seed: RANDOM, used to generate random numbers	

4.2	SHARED_INFORMATION_ACCESS	Responsibility: Provide singleton access to SHARED_INFORMATION	Alternative: N/A
	Concrete	Secret: None	

4.2.1	SHARED_INFORMATION	Responsibility: Provides a central storage location for default game values such as board size and initial number of ENTITY_STATIONARY in the game	Alternative: N/A
	Concrete	Secret: None	

4. Entities Cluster Design Descriptions

4.1. ENTITY_ALPHABET and Descendent Classes BON Diagram



4.2. ENTITY_ALPHABET and Descendent Classes Design Decisions

4.2.1. Benefits of the implemented system

The key module within SimOdyssey2 that provides for future expansion and simplicity of game logic is the Entities Cluster. The Entities Cluster encompasses the deferred ENTITY_ALPHABET, ENTITY_MOVABLE and ENTITY_STATIONARY classes and their implemented descendents. In particular, the deferred class ENTITY_MOVABLE provides the ability for any future interactive entities to be added to the game provided they can be distilled to the check, reproduce and behave function routines.

By leveraging dynamic-typing a new entity can inherit from ENTITY_MOVABLE and implement actions through these check, reproduce and behave functions and be completely integrated into the game. While the ETF_MODEL provides the overall game logic and structure by controlling the sequence of events, the events themselves are defined and processed dynamically from within the Entities cluster. This dynamic-typing also enables an implemented class to not perform one of the three defined functions by implementing an empty function, such as EXPLORER for the reproduce and behave actions (see 4.2.2. for this design decision).

Further, if there is a desire to change how an implemented entity acts within the game there is a single location to modify the code and the rest of the system will continue to function as expected.

Finally, this general architecture of a base game piece further extended and redefined to perform a myriad of tasks can be applied to other games with nothing in common to SimOdyssey2 other than being a game with interactive pieces, thus promoting code-reuse.

4.2.2. Alternatives considered

The initial sketch of the ENTITY_ALPHABET descendents contained considerably more abstraction. This included an additional deferred class inherited from ENTITY_MOVABLE called ENTITY_BEHAVIOUR as the EXPLORER class does not behave without user commands. Additionally, since the BENIGN, MALEVOLENT and JANITAURs are the only ones to reproduce the initial design had them abstracted further with a deferred class, ENTITY_REPRODUCE, inheriting from the proposed ENTITY_BEHAVIOUR class.
On paper this design appeared desirable as it kept the classes with shared actions combined under their own deferred class. In reality this level of abstraction created many issues at the ETF_MODEL level of the code. Instead of iterating through a list of ENTITY_MOVABLE types and dynamically calling the correct check, reproduce and behave methods the entity type had to be tested at each iteration. For example, if the entity was an EXPLORER, perform the check while skipping the reproduce and behave calls. Whereas an ASTEROID would perform the check, skip the reproduce before performing the behave action. Every type of implemented entity would have to be checked for with this design. Any changes to the design or adding new entity types would require significant design changes throughout the project.

5. Contracts in the SECTOR Class

Contracts are used throughout the implementation of SimOdyssey2, however, some key contracts occur within the SECTOR class as it is the class adding and removing elements from data structures without using default iterator access.

5.1. Modify Data Structure Contracts

5.1.1. SECTOR.put (entity: ENTITY_ALPHABET)

ensure entity_put: not is_full implies contents.has (entity)

The ‘ensure’ contract guarantees that if the put function was called on a non-empty SECTOR the entity will be placed within the SECTOR. This contract is important because, as implemented, a SECTOR can have ‘empty’ locations that at the implementation level are Void. As such, the implementation is not as simple as using extend on the array of entities within the SECTOR.

5.1.2. SECTOR.remove (entity: ENTITY_ALPHABET)

require contains_entity: has (entity)

The ‘require’ contract is vital because the implementation makes use of the idex_of method to get the index of the entity and set that location to Void. Without the contract client classes would not be aware of this and usage could cause the system to crash by accessing attempting to set an invalid array index to Void.

ensure removed_entity: not has (entity)

While less vital than the ‘require’ contract it is important for the client class to know, contractually, the entity has been removed from the game as this can cause cascading problems for the game.

5.2. Query Data Structure Contracts

5.2.1. SECTOR.next_available_quad: INTEGER

require not_is_full: not is_full

Function next_available_quad returns the index of the first empty space in the SECTOR from right to left. The contract requires the client to check beforehand, otherwise an invalid index could be returned and potentially accessed.

ensure valid_index: Result > 0 and Result <= shared_info.Max_capacity

Contacts guarantees to the client that they can directly access the index value returned without issue.
Note: Max_capacity is defined in the SHARED_INFORMATION class (default is 4).

5.2.2. SECTOR.item_at (index: INTEGER): detachable ENTITY_ALPHABET

require valid_index: index > 0 and index <= shared_info.Max_capacity

As with any accessor function, this contract guarantees an invalid index does not cause a segmentation and system crash by attempting to access memory outside of the bounds of the array. Note: the return is potentially Void due to the detachable nature of the implementation so the function does not guarantee a returned entity.

6. Testing Results

6.1. Test Descriptions and Results

Test File	Description	Results
at001.txt	play and status command	Passed
at002.txt	play and lots of EXPLORER moves followed by pass many times	Passed
at003.txt	successive play and aborts followed by pass many times	Passed
at004.txt	various test command inputs and aborts	Passed
at005.txt	EXPLORER death due to BLACKHOLE	Passed
at006.txt	Lots of ASTEROIDS and passes, test death messages	Passed
at007.txt	Test quick death due to ASTEROID	Passed
at008.txt	Test quick death due to ASTEROID followed by many test, abort and pass	Passed
at009.txt	Winning condition in play mode with many passes	Passed
at010.txt	Land and liftoff from planet, valid and invalid	Passed
at011.txt	Take EXPLORER through a WORMHOLE, with and without one present	Passed
at012.txt	Test and abort with nearly all ASTEROIDS and many passes	Passed
at013.txt	test_mode with many entities and passes	Passed
at014.txt	Land on multiple PLANETs in the same sector	Passed
at015.txt	Even spread of entities with test_mode and large number of passes	Passed
at016.txt	Few entities, move EXPLORER wrap corner to corner until out of fuel	Passed
at017.txt	Lots of BENIGN and MALEVOLENT interacting with each other	Passed
at018.txt	Lots of ASTEROIDS and JANITAURS interacting with each other	Passed
at019.txt	Lots of planets, moving, orbiting and supporting life correctly	Passed
at020.txt	EXPLORER die by out of fuel as entering BLACKHOLE	Passed
at021.txt	Winning condition, abort, restart very large number of passes	Passed
at022.txt	Several successive wins, aborts and passes	Passed
at023.txt	Large number of entities, pass test added contracts	Passed

6.2. Output from Regression Testing

```
Success: log/student/at001.actual.txt and log/student/at001.expected.txt are identical.
Success: log/student/at002.actual.txt and log/student/at002.expected.txt are identical.
Success: log/student/at003.actual.txt and log/student/at003.expected.txt are identical.
Success: log/student/at004.actual.txt and log/student/at004.expected.txt are identical.
Success: log/student/at005.actual.txt and log/student/at005.expected.txt are identical.
Success: log/student/at006.actual.txt and log/student/at006.expected.txt are identical.
Success: log/student/at007.actual.txt and log/student/at007.expected.txt are identical.
Success: log/student/at008.actual.txt and log/student/at008.expected.txt are identical.
Success: log/student/at009.actual.txt and log/student/at009.expected.txt are identical.
Success: log/student/at010.actual.txt and log/student/at010.expected.txt are identical.
Success: log/student/at011.actual.txt and log/student/at011.expected.txt are identical.
Success: log/student/at012.actual.txt and log/student/at012.expected.txt are identical.
Success: log/student/at013.actual.txt and log/student/at013.expected.txt are identical.
Success: log/student/at014.actual.txt and log/student/at014.expected.txt are identical.
Success: log/student/at015.actual.txt and log/student/at015.expected.txt are identical.
Success: log/student/at016.actual.txt and log/student/at016.expected.txt are identical.
Success: log/student/at017.actual.txt and log/student/at017.expected.txt are identical.
Success: log/student/at018.actual.txt and log/student/at018.expected.txt are identical.
Success: log/student/at019.actual.txt and log/student/at019.expected.txt are identical.
Success: log/student/at020.actual.txt and log/student/at020.expected.txt are identical.
Success: log/student/at021.actual.txt and log/student/at021.expected.txt are identical.
Success: log/student/at022.actual.txt and log/student/at022.expected.txt are identical.
Success: log/student/at023.actual.txt and log/student/at023.expected.txt are identical.
Success: log/student/at024.actual.txt and log/student/at024.expected.txt are identical.
Success: log/student/at025.actual.txt and log/student/at025.expected.txt are identical.
Success: log/student/at026.actual.txt and log/student/at026.expected.txt are identical.
Success: log/student/at027.actual.txt and log/student/at027.expected.txt are identical.
Success: log/instructor/at001.actual.txt and log/instructor/at001.expected.txt are identical.
Success: log/instructor/at001.actual.txt and log/instructor/at001.expected.txt are identical.
Success: log/instructor/at002.actual.txt and log/instructor/at002.expected.txt are identical.
Success: log/instructor/at002.actual.txt and log/instructor/at002.expected.txt are identical.
Success: log/instructor/at003.actual.txt and log/instructor/at003.expected.txt are identical.
Success: log/instructor/at003.actual.txt and log/instructor/at003.expected.txt are identical.
============================
Test Results: 33/33 passed.
============================
```

7. Appendix

7.1 Contract View of all Classes

7.1.1. ETF_MODEL_ACCESS

```
note
	description: "Singleton access to the default business model."
	author: "Jackie Wang"
	date: "$Date$"
	revision: "$Revision$"

expanded class interface
	ETF_MODEL_ACCESS

create
	default_create

feature

	M: ETF_MODEL

invariant
		M = M

end -- class ETF_MODEL_ACCESS
```

7.1.2. ETF_MODEL

```
note
	description: "Primary game logic of SimOdyssey2 Boardgame"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	ETF_MODEL

create {ETF_MODEL_ACCESS}
	make

feature -- model attributes

	game_board: GALAXY
			--access to the board that contains all sectors and entities in the game

	play_mode: BOOLEAN
			--did the user start the game in play mode
```

```
	shared_info: SHARED_INFORMATION
			--singleton access to shared information

	shared_info_access: SHARED_INFORMATION_ACCESS
			--access the base game state info like size of the board

	test_mode: BOOLEAN
			--did the user start the game in test mode, provides additional
			--information
			--about what happens each turn

feature -- model operations

	abort
			--quit the current game and remove all pieces from the board
			--increment the error state
		require
			in_game: play_mode or test_mode
		ensure
			error_state_updated: error_state = old error_state + 1
			game_ended: not (play_mode or test_mode)

	initialize_game (a_thresh: INTEGER_32; j_thresh: INTEGER_32; m_thresh: INTEGER_32;
				b_thresh: INTEGER_32; p_thresh: INTEGER_32)
		require
			valid_threshold: 0 < a_thresh and a_thresh <= j_thresh and
			j_thresh <= m_thresh and m_thresh <= b_thresh and
			b_thresh <= p_thresh and p_thresh <= 101

	land_explorer
			--land the explorer at the first unvisited plane in the
			--sector and check to see if life was found
			--increment game state and reset error state
			--complete a turn for all movable entities if no life found
		require
			not_landed: not game_board.explorer.landed
			in_game: play_mode or test_mode

	liftoff_explorer
			--lift the explorer back into space in the current sector
			--increment game state and reset error state
			--complete a turn for all movable entities
		require
			landed: game_board.explorer.landed
			in_game: play_mode or test_mode
		ensure
			game_state_updated: game_state = old game_state + 1
```

```
move_explorer (dest_row: INTEGER_32; dest_col: INTEGER_32)
            --move the explorer to the first available quadrant in
            --[dest_row, dest_col]
            --increment game state and reset error state
            --complete a turn of all movable entities in the game
      require
            in_game: play_mode or test_mode
            not_landed: not game_board.explorer.landed
      ensure
            game_state_updated: game_state = old game_state + 1

pass
            --the explorer does not move or act this turn
            --increments the game state and resets the error state
            --causes a turn for all movable entities to occur
      require
            in_game: play_mode or test_mode
      ensure
            game_state_updated: game_state = old game_state + 1

set_error (msg: STRING_8)
            --User command triggered an error, increment error state
      ensure
            error_state_updated: error_state = old error_state + 1

set_play
            --Turn on play mode, increment the current game state
            --and reset the error state
      require
            not_in_game: not (test_mode or play_mode)
      ensure
            game_state_updated: game_state = old game_state + 1
            set_play_mode: play_mode

set_test
            --Turn on test mode which provides the user with additional
            --information as to what each entity is doing every turn.
            --Increment the game state and reset the error state
      require
            not_in_game: not (test_mode or play_mode)
      ensure
            game_state_updated: game_state = old game_state + 1
            set_test_mode: test_mode

status
            --update the status of the explorer, including
            --if explorer is flying, landed and their current life, fuel and
            -- location
            --counts as an error state as it does not cause a turn of all the
            --movable entities to occur
      require
```

```
                in_game: play_mode or test_mode
            ensure
                error_state_updated: error_state = old error_state + 1

        wormhole_explorer
                --send the explorer through the wormhole to a random location
                --increment game state and reset error state
                --complete a turn of all movable entities in the game
            require
                in_game: play_mode or test_mode
            ensure
                game_state_updated: game_state = old game_state + 1

feature -- queries

        get_death_msgs: STRING_8
                --Returns a list of all of the entities that died during
                --the turn

        out: STRING_8
                --Output the state of the game to the user
feature --operations support

        get_dest_coord (start: PAIR [INTEGER_32, INTEGER_32];
                increment: PAIR [INTEGER_32, INTEGER_32]): PAIR [INTEGER_32, INTEGER_32]
                --Given a start coordinate as [x,y] and the direction to move as
                -- [x_inc,y_inc],
                --determine the destination of [x + x_inc, y + y_inc]. Wraps around the
                -- edges of the board in all directions
            require
                valid_start: (start.first <= shared_info.Number_rows) and
                            (start.second <= shared_info.Number_columns)
            ensure
                valid_dest: (Result.first <= shared_info.Number_rows) and
                            (Result.second <= shared_info.Number_columns)

        map_direction (direction: INTEGER_32): PAIR [INTEGER_32, INTEGER_32]
                --Map a given single integer value [1,8] to a increment
                --representing a direction change from [0,0].
                --Start with 1 = N, 2 = NE until 8 = NW
            require
                valid_direction: direction >= 1 and direction <= 8
            ensure
                valid_coord: (Result.first <= shared_info.Number_rows) and
                             (Result.second <= shared_info.Number_columns)

end -- class ETF_MODEL
```

7.1.3. GALAXY

```
note
	description: "Galaxy represents a game board in simodyssey."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	GALAXY

create
	make

feature -- attributes

	explorer: EXPLORER
			--the galaxies intrepid explorer, the players regent

	gen: RANDOM_GENERATOR_ACCESS

	movable_entities: SORTED_TWO_WAY_LIST [ENTITY_MOVABLE]
			--all of the movable entities in the galaxy, sorted by increasing id

	movable_id: INTEGER_32
			--value of the next id to assign to a new MOVABLE_ENTITY

	shared_info: SHARED_INFORMATION

	shared_info_access: SHARED_INFORMATION_ACCESS

	stationary_entities: SORTED_TWO_WAY_LIST [ENTITY_STATIONARY]
			--all of the stationary entities in the galaxy, sorted by increasing id

feature -- query

	get_entity_desc: STRING_8
			--Build a string representing all stationary and
			--movable entities currently on the board and the entities
			--vital stats

	get_sector (row: INTEGER_32; col: INTEGER_32): SECTOR
		require
			in_bounds: row <= shared_info.Number_rows and
					col <= shared_info.Number_columns

	get_sector_desc: STRING_8
			--Builds a string representation of all of the entity id's and
```

```
                --symbols at each quadrant of each sector
                --Used for the 'Section' output in test mode

        out: STRING_8
                --Returns grid in string form

feature --commands

        create_stationary_item (id: INTEGER_32; location: SECTOR): ENTITY_STATIONARY
                -- this feature randomly creates one of the possible types of stationary
                -- actors

        inc_movable_id
                --increments the movable id, used by model to
                --set id of entities created in the reproduce phase of a turn

        set_movable_items (a_thresh: INTEGER_32; j_thresh: INTEGER_32; m_thresh: INTEGER_32;
                           b_thresh: INTEGER_32; p_thresh: INTEGER_32)
                --Set a random number of movable entities at each sector.
                --Likelihood of each entity type is defined as:
                --Asteroid (0,a_thresh], Janitaur [a_thresh,j_thresh],
                --Malevolent [m_thresh,b_thresh], Benign[b_thresh,p_thresh] and
                --Planet[p_thresh,101]
              require
                valid_threshold: 0 < a_thresh and a_thresh <= j_thresh and
                                 j_thresh <= m_thresh and m_thresh <= b_thresh and
                                  b_thresh <= p_thresh and p_thresh <= 101

        set_stationary_items
                -- distribute stationary items amongst the sectors in the grid.
                -- There can be only one stationary item in a sector

feature --constructor

        make
                -- creates a dummy of galaxy grid
                -- Places the explorer at [1,] and a blackhole at [3,3]

end -- class GALAXY
```

7.1.4. SECTOR

```
note
	description: "Represents a sector which holds a set of entities in the galaxy."
	author: ""
	date: "$Date$"
	revision: "$Revision$"

class interface
	SECTOR

create
	make,
	make_dummy

feature -- Queries

	contents_out: STRING_8
			--build a string of
[row,col]->[id,symbol],[id,symbol],[id,symbol],[id,symbol]
			--or a '-' in quadrants with no entity

	get_movables: SORTED_TWO_WAY_LIST [ENTITY_MOVABLE]
			--Return a list of the movable entities in this sector sorted by id.
			--Can be an empty list

	get_stationary: ENTITY_STATIONARY
			--Return a reference to the ENTITY_STATIONARY in this sector,
			--must contain an ENTITY_STATIONARY
		require
				Current.has_stationary
		ensure
			found_stationary: Result.id /= -999

	has (entity: ENTITY_ALPHABET): BOOLEAN

	has_benign: BOOLEAN
			--Returns TRUE if this sector contains a Benign entity

	has_blackhole: BOOLEAN
			--Returns TRUE if this sector contains a blackhole

	has_planet: BOOLEAN
			--Returns TRUE if this sector contains a Planet

	has_star: BOOLEAN
			--Returns TRUE if this sector contains a star

	has_stationary: BOOLEAN
			-- returns whether the location contains any stationary item
```

```
	has_wormhole: BOOLEAN
			--Returns TRUE if this sector contains a wormhole

	has_yellow_dwarf: BOOLEAN
			--Returns TRUE if this sector contains a Yellow Dwarf star

	index_of (entity: ENTITY_ALPHABET): INTEGER_32
		require
			valid_entity: has (entity)
		ensure
			entity_present: Result > 0 and Result <= shared_info.Max_capacity

	is_full: BOOLEAN
			-- Is the location currently full?

	item_at (index: INTEGER_32): detachable ENTITY_ALPHABET
		require
			valid_index: index > 0 and index <= shared_info.Max_capacity

	next_available_quad: INTEGER_32
			--find the next empty quadrant of the sector
		require
				not is_full
		ensure
			valid_index: Result > 0 and Result <= shared_info.Max_capacity

	number_entities: INTEGER_32
		ensure
			valid_result: Result = contents.count

	out: STRING_8
			--build a string with [row,col] for the sector

feature -- attributes

	column: INTEGER_32

	gen: RANDOM_GENERATOR_ACCESS

	row: INTEGER_32

	shared_info: SHARED_INFORMATION

	shared_info_access: SHARED_INFORMATION_ACCESS

feature -- commands

	make_dummy
			--initialization without creating entities in quadrants
```

```
	put (entity: ENTITY_ALPHABET)
			-- put `entity` in contents array at the first available quadrant
		ensure
			entity_put: not is_full implies contents.has (entity)

	remove (entity: ENTITY_ALPHABET)
			--remove 'entity' from the quadrant by setting the index of entity to
			-- void
		require
			contains_entity: has (entity)
		ensure
			removed_entity: not has (entity)

feature -- constructor

	make (row_input: INTEGER_32; column_input: INTEGER_32)
			--Create an empty sector
		require
			valid_row: (row_input >= 1) and (row_input <= shared_info.Number_rows)
			valid_column: (column_input >= 1) and
					(column_input <= shared_info.Number_columns)

end -- class SECTOR
```

7.1.5. ENTITY_ALPHABET

```
note
	description: "[
		Alphabet allowed to appear on the galaxy board.
	]"
	author: ""
	date: "April 30, 2019"
	revision: "1"

deferred class interface
	ENTITY_ALPHABET

feature --Comparison

	is_equal (other: like Current): BOOLEAN
			--id and item define equality

	is_less alias "<" (other: like Current): BOOLEAN
			--for sorting by ascending id

feature -- Attributes common to all entities

	id: INTEGER_32
			--unique id of the entity
```

```
	item: CHARACTER_8
			--character that represents the entity

	location: SECTOR
			--location of the entity

feature -- Query

	id_out: STRING_8
			--Returns a string of form: [id,symbol]

	is_asteroid: BOOLEAN
			-- Return if current item is an asteroid

	is_benign: BOOLEAN
			-- Return if current item is benign

	is_blackhole: BOOLEAN
			-- Return if current item is a blackhole

	is_blue_giant: BOOLEAN
			-- Return if current item is a blue giant star

	is_explorer: BOOLEAN
			-- Return if current item is the explorer

	is_janitaur: BOOLEAN
			-- Return if current item is janitaur

	is_malevolent: BOOLEAN
			-- Return if current item is malevolent

	is_movable: BOOLEAN
			-- Return if current item is movable

	is_planet: BOOLEAN
			-- Return if current item is a planet

	is_star: BOOLEAN
			-- Return if current item is a star

	is_stationary: BOOLEAN
			-- Return if current item is stationary.

	is_wormhole: BOOLEAN
			-- Return if current item is a wormhole

	is_yellow_dwarf: BOOLEAN
			-- Return if current item is a yellow dwarf star

	loc_out: STRING_8
```

```
                --Returns the full address of the entity [row,column,quadrant]

      out: STRING_8
                -- Return string representation of the entity

invariant
      allowable_symbols: item = 'E' or item = 'P' or item = 'A' or item = 'M' or
                         item = 'J' or item = 'O' or item = 'W' or item = 'Y' or
                         item = '*' or item = 'B'

end -- class ENTITY_ALPHABET
```

7.1.6. ENTITY_STATIONARY

```
note
      description: "Base class for stationary entities"
      author: "Chris Boyd : 216 869 356 : chris360"
      date: "$Date$"
      revision: "$Revision$"

deferred class interface
      ENTITY_STATIONARY

feature --Queries

      get_status: STRING_8
                --[id,item] representation of the entity

invariant
      allowable_symbols: item = 'O' or item = 'W' or item = 'Y' or item = '*'

end -- class ENTITY_STATIONARY
```

7.1.7. STAR

```
note
      description: "Deferred class for Star Entities, which have a luminosity value that
                    provides fuel"
      author: "Chris Boyd : 216 869 356 : chris360"
      date: "$Date$"
      revision: "$Revision$"

deferred class interface
      STAR

feature --Queries

      get_status: STRING_8
                --returns [id,item]->Luminosity:X

feature --attributes
```

```
	luminosity: INTEGER_32
			--how much fuel does the star provide to passing entities

invariant
	allowable_symbols: item = '*' or item = 'Y'

end -- class STAR
```

7.1.8. BLUE_GIANT

```
note
	description: "Vibrant star that provides up to 5 units of fuel"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	BLUE_GIANT

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)

invariant
	allowable_symbols: item = '*'

end -- class BLUE_GIANT
```

7.1.9. YELLOW_DWARF

```
note
	description: "Star that provides 2 fuel and can support life"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	YELLOW_DWARF

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)

invariant
	allowable_symbols: item = 'Y'
```

```
end -- class YELLOW_DWARF
```

7.1.10. BLACKHOLE

```
note
	description: "Stationary entity that kills everything that enters the sector"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	BLACKHOLE

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)

invariant
	allowable_symbols: item = 'O'

end -- class BLACKHOLE
```

7.1.11. WORMHOLE

```
note
	description: "Wormhole allows instant travel between sectors without using fuel"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	WORMHOLE

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)

invariant
	allowable_symbols: item = 'W'

end -- class WORMHOLE
```

7.1.12. ENTITY_MOVABLE

```
note
	description: "Provides base interface for all movable entities"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

deferred class interface
	ENTITY_MOVABLE

feature --Queries

	get_entity_msg: STRING_8
			--Return the entity's status message, likely set in the event
			--of the entity being killed by something.

	get_move_info: STRING_8
			--return the string of move entity performed during current turn

	is_dead: BOOLEAN
			--Is the entity out of life?

feature --attributes

	entity_msg: STRING_8
			--set by other entities that interact with this entity
			--through their behave action

	fuel: INTEGER_32
			--how much fuel this entity has until it will die

	gen: RANDOM_GENERATOR_ACCESS
			--deterministic random number generator used for
			--travelling through wormholes and resetting number of turns

	life: INTEGER_32
			--how many lives does this entity have left until death

	move_info: STRING_8
			--represents where this entity moved from and to during
			--a turn

	turns_left: INTEGER_32
			--how many turns left until this entity acts

feature --commands

	kill
			--kill the entity
```

```
	move (dest: SECTOR)
			--Move the entity to the destination sector. If destination sector
			--is full does nothing

	set_entity_msg (msg: STRING_8)
			--Set the entity's status message, used by other entities
			--when interacting with this entity

	set_turns (i: INTEGER_32)
			--Set number of turns for the entity

	take_life
			--Take a life from the current entity
		require
				has_life: life > 0

	wormhole (board: GALAXY)
			--Sends the entity through the wormhole which randomly selections
			--a destination sector on the board until a non-full sector is found.
			--Can send the entity back to the start location.

feature --deferred command

	behave: LINKED_LIST [ENTITY_MOVABLE]
			--perform inherited class behaviour, returns list
			--of entities killed by Current during the turn (may be empty)

	check_post_move
			--perform inherited class post move actions

	get_name: STRING_8
			--get string representation of inherited class name

	get_status: STRING_8
			--return status string of inherited class

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--if reproduce timer is up, create a new entity of same type in
			--current location (if not full)

invariant
	allowable_symbols: item = 'E' or item = 'P' or item = 'A' or item = 'M' or
				item = 'J' or item = 'B'

end -- class ENTITY_MOVABLE
```

7.1.13. ASTEROID

```
note
	description: "Asteroids randomly move about the galaxy destroying"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	ASTEROID

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)

feature --Commands

	behave: LINKED_LIST [ENTITY_MOVABLE]
			--seeks out movable_entities other than asteroids
			--and planets and destroys in ascending id order
			--entity got destroyed by asteroid (id: Z) at Sector:X:Y

	check_post_move
			--Checks if asteroids new location contains a blackhole, which will
			--kill the asteroid

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--Asteroids do not reproduce

feature --Queries

	get_name: STRING_8
			--Return string representation of this class

	get_status: STRING_8
			--Returns current status of the asteroid for
			--outputting in test mode [id,A]->turns_left:X

invariant
	allowable_symbols: item = 'A'

end -- class ASTEROID
```

7.1.14. BENIGN

```
note
	description: "Entity that protects Explorer from Malevolent and kill Malevolents"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	BENIGN

create
	make

feature -- Initialization

	make (i: INTEGER_32; loc: SECTOR)
			-- Initialization for `Current`.

feature --Commands

	behave: LINKED_LIST [ENTITY_MOVABLE]
			--destroy all malevolent in sector from low to high id
			--Malevolent got destroyed by benign (id: Z) at Sector:X:Y

	check_post_move
			--check if the sector has a star to refuel from
			--check if there are any blackholes that will devour malevolent in the
			--sector
			--if runs out of fuel entering a sector with a blackhole, dies due to
			--out of fuel

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--if reproduce timer is up, create a new benign in
			--current location (if not full)

feature --Queries

	get_name: STRING_8
			--Return string representation of this class

	get_status: STRING_8
			--Returns current status of the benign for
			--outputting in test mode
			--[id,B]->fuel:fF/3, actions_left_until_reproduction:A/1, turns_left:X

feature --attributes

	repro_interval: INTEGER_32
```

```
invariant
	allowable_symbols: item = 'B'

end -- class BENIGN
```

7.1.15. EXPLORER

```
note
	description: "Intrepid explorer that travels the galaxy seeking new life"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	EXPLORER

create
	make

feature -- Initialization

	make (i: INTEGER_32; loc: SECTOR)
			-- Initialization for `Current`.

feature --Attributes

	found_life: BOOLEAN
			--did the explorer find life on a planet after landing

	landed: BOOLEAN
			--is the explorer landed on a planet

feature --Commands

	behave: LINKED_LIST [ENTITY_MOVABLE]
			--Explorer does not behave without user input

	check_post_move
			--Checks if explorers new location contains a blackhole, which will
			--kill the explorer, checks for a star to refuel, if life or fuel
			--reach 0, dies

	land: STRING_8
			--Attempt to land on an unvisited planet in the sector to seek out life
			--must be a yellow star in the sector
			--Can attempt to land during successive turns on each planet in current
			--sector
			--until all are visited
			--Cannot move again until the explorer lifts off
		require
			not_landed: not landed
```

```
	liftoff
			--Explorer leaves the planet for new adventures
		require
			landed: landed

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--Explorer does not reproduce, sad explorer

feature --Queries

	get_name: STRING_8
			--Return string representation of this class

	get_status: STRING_8
			--Returns current status of the benign for
			--outputting in test mode
			--[id,E]->fuel:F/3, life:L/3, landed?:t|f

invariant
	allowable_symbols: item = 'E'

end -- class EXPLORER
```

7.1.16. JANITAUR

```
note
	description: "Entity that seeks out and destroys asteroids"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	JANITAUR

create
	make

feature -- Initialization

	make (i: INTEGER_32; loc: SECTOR)
			-- Initialization for `Janitaur`.

feature --Commands

	behave: LINKED_LIST [ENTITY_MOVABLE]
			--Janitaur grabs up to two asteroids in the current sector
			--taking them off the board until they can be dumped in a sector
			--with a wormhole, from which they do not return
			--Asteroid got imploded by janitaur (id: Z) at Sector:X:Y
```

```
	check_post_move
			--Check for a star to refuel at, if run out of fuel or enter
			--sector with a blackhole, dies

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--if reproduce timer is up, create a new janitaur in
			--current location (if not full)

feature --Queries

	get_name: STRING_8
			--Return string representation of this class

	get_status: STRING_8
			--Returns current status of the benign for
			--outputting in test mode
			--[id,J]->fuel:fF/3, actions_left_until_reproduction:A/2, turns_left:X

feature --attributes

	load_level: INTEGER_32
			--number of asteroids janitaur can transport, max 2

	repro_interval: INTEGER_32
			--number of turns before Janitaur reproduces (default = 2)

invariant
	allowable_symbols: item = 'J'

end -- class JANITAUR
```

7.1.17. MALEVOLENT

```
note
	description: "Evil entity that seeks out and attacks"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	MALEVOLENT

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)
			-- Initialization for `malevolent`.

feature --Commands
```

```
	behave: LINKED_LIST [ENTITY_MOVABLE]
			--attack explorer, if present and not landed and no benign in sector

	check_post_move
			--Refuel if there is a star in the sector, check if Malevolent ran out
			--of fuel or if there is a blackhole, both of which kill the Malevolent.

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--If the Malevolent has a repro_interval of 0, create a new Malevolent
			-- object in the current sector if there is space

feature --Queries

	get_name: STRING_8
			--Return string representation of this class

	get_status: STRING_8
			--Returns current status of the benign for
			--outputting in test mode
			--[id,M]->fuel:fF/3, actions_left_until_reproduction:A/1, turns_left:X

feature --attributes

	repro_interval: INTEGER_32
			----number of turns before Malevolent reproduces (default = 1)

invariant
	allowable_symbols: item = 'M'

end -- class MALEVOLENT
```

7.1.18. PLANET

```
note
	description: "Planets travel the galaxy seeking out stars to orbit"
	author: "Chris Boyd : 216 869 356 : chris360"
	date: "$Date$"
	revision: "$Revision$"

class interface
	PLANET

create
	make

feature --Initialization

	make (i: INTEGER_32; loc: SECTOR)
			-- Initialization for `planet`.
```

```
feature --Attributes

	has_set: BOOLEAN
			--has this planet already randomly determined if it
			--supports life, necessary to not re-determine during
			--every behave operation

	orbiting: BOOLEAN
			--is this planet orbiting a star

	supports_life: BOOLEAN
			--does this planet support life

	visited: BOOLEAN
			--has the explorer already visited this planet

feature --Commands

	behave: LINKED_LIST [ENTITY_MOVABLE]
			--If sector has a star, orbit it, if the star is a yellow dwarf
			--randomly assign if planet supports life (50% chance)
			--Once this value is set, it will not change

	check_post_move
			--if planet moves into a sector with blackhole, dies

	reproduce (next_id: INTEGER_32): detachable ENTITY_MOVABLE
			--planets don't reproduce

	set_life
			--planet supports life

	set_orbit
			--Planet is now orbiting a star

	set_visited
			--planet has been visited by explorer

feature --Queries

	get_name: STRING_8
			--Return string representation of this class

	get_status: STRING_8
			--Returns current status of the benign for
			--outputting in test mode
			--[id,P]->attached?:t|f, support_life?:t|f, visited?:t|f, turns_left:X
invariant
	allowable_symbols: item = 'P'

end -- class PLANET
```

7.2. Implemented Entities Descriptions

7.2.1. ASTEROID

CHECK
• If in sector (3,3), it dies by blackhole.
BEHAVE
• Seeks any other movable entities in its sector except planets and other asteroids and destroys all of them in ascending id order. (Note that the explorer cannot be hit if it is landed).
• Reset turns_left 0-2.

7.2.2. BENIGN

CHECK
• Moving successfully uses 1 fuel, while using a wormhole does not cost any fuel.
• Fuel is gained based on star luminosity in the current sector.
• Fuel of 0 means death.
• If in sector (3,3), it dies by blackhole.
REPRODUCTION
• In the new sector, it tries to clone itself (create a new copy) if its reproduce is 0 and the current sector is not already full, resetting reproduce to reproduction_interval if done successfully. Otherwise reproduce = reproduce – 1 until reproduce is 0.
BEHAVE
• It will destroy all malevolent entities in the current sector in the order from lowest to highest id.
• Reset turns_left 0-2.

7.2.3. EXPLORER

• It is a movable entity.
• The explorer gets to choose which action to perform when activation occurs. Please refer to section 7.3.
• Has maximum fuel of 3 which decreases by 1 each time it moves. It can also take a wormhole, pass, land or liftoff which does not consume any fuel. Fuel is recharged by moving into a sector with a star where it gains fuel equivalent to the star’s luminosity value.
• Has a life value of three, which is reduced by one each time when attacked by a malevolent and reduced to zero when running out of fuel, entering a region with a black hole and being hit by an asteroid).
• A life or fuel value of zero ends the game.
• Represented by the character ‘E’.

7.2.4. JANITAUR

CHECK
• Moving successfully uses 1 fuel.
• Fuel is gained based on star luminosity in the current sector.
• Fuel of 0 means death.
• If in sector (3,3), it dies by blackhole.
REPRODUCTION

• In the new sector, it tries to clone itself (create a new copy) if its reproduce is 0 and the current sector is not already full, resetting reproduce to reproduction_interval if done successfully. Otherwise reproduce = reproduce – 1 until reproduce is 0.
BEHAVE
• Unless its maximum_load_level has been reached, look for asteroids to implode and haul away (where it destroys all the asteroids in that sector and increment the load level by the number of asteroids destroyed). If there are multiple asteroids and not enough room in the janitaur, lower id asteroids are targeted first.
• If a wormhole is in the current sector, it will then throw all the asteroids into it, clearing the load level. Note the asteroids thrown into the wormhole do not appear anywhere.
• Reset turns_left 0-2.

7.2.5. MALEVOLENT

CHECK
• Moving successfully uses 1 fuel, while using a wormhole does not cost any fuel.
• Fuel is gained based on star luminosity in the current sector.
• Fuel of 0 means death.
• If in sector (3,3,), it dies by blackhole.
REPRODUCTION
• In the new sector, it tries to clone itself (create a new copy) if its reproduce is 0 and the current sector is not already full, resetting reproduce to reproduction_interval if done successfully. Otherwise reproduce = reproduce – 1 until reproduce is 0.
BEHAVE
• Then in the absence of a benign the given sector, looks for the explorer to attack. (Explorer cannot be attacked if it is landed).
• Reset turns_left 0-2.

7.2.6. PLANET

CHECK
• If in sector (3,3), it dies
BEHAVE
• If the sector contains a star, the planet remains in that sector (and becomes attached).
• If a planet shares a sector with a yellow dwarf, the planet has a 50% chance to support life
• If the planet is not attached, then reset turns_left 0-2.

7.3. SimOdyssey2 Interface Definition

```
-- SimOdyssey System Types
-----------------------------------------------------------------

type DIRECTION = {N, NE, E, SE, S, SW, W, NW}
        --movement directions

type THRESHOLD     = 1..101
        --values for specifying thresholds in test mode

-----------------------------------------------------------------
-- SimOdyssey User Commands
-----------------------------------------------------------------

test(a_threshold:THRESHOLD ;
            j_threshold:THRESHOLD; m_threshold:THRESHOLD;
            b_threshold:THRESHOLD ; p_threshold:THRESHOLD)
  -- Starts a new game in test mode provided game
  -- has not been started yet or is over.
  -- Test mode uses a deterministic random generator and displays
  -- the abstract state of the game.
  -- Allows the setting of threshold values to populate the board initially
  -- (between 1 and 101 non-decreasing), e.g
      -- a_threshold: 20, i.e. generate Asteroids for 1..19
      -- j_threshold: 40, i.e. generate Janitaurs for 20..39
      -- m_threshold: 50, i.e. generate Malevolents for 40..49
      -- b_threshold: 60, i.e. generate Benigns for 50..59
      -- p_threshold: 70, i.e. generate Planets for 60..69
      -- a random number of 70 to 100 generates no moveable entities
    -- If the random number generated a number from 1 to 100,
      -- if the number is in the interval from 1 (inclusive)
    -- to the first number (exclusive), an asteroid
    -- is created, first number (inclusive) to second
    -- number (exclusive) is janitaur, second number (inclusive)
    -- to third number (exclusive) is malevolent, third number
      -- (inclusive) to fourth number (exclusive) is
      -- benign, fourth number (inclusive) to the
      -- fifth number (exclusive) is planet and fifth number
    -- (inclusive) to 101 (exclusive) is nothing.
  -- Note that this command will not cause a turn to pass/occur.

play
  -- Starts a new game using test(3,5,7,15,30)
  -- provided a game has not been started yet or is over.
```

```
  -- Play mode displays only the board and key messages as outputs
  -- and not the complete abstract state.

abort
  -- Ends the game prematurely. Only valid when game is
  -- in progress.

move (dir: DIRECTION)
   -- Moves the explorer in a given direction.
   -- A game has to be in progress and the sector
   -- to travel to is not full.
   -- Note that this command will cause a turn to pass/occur.
   -- After the explorer moves, other moveable entities whose clock
   -- time (rest) is zero also act in id order, i.e. 1, 2, ..
   -- In test mode: displays entity actions, abstract state,
   -- then board.

land
   -- Lands the explorer on a planet to check for life on planet.
   -- A game has to be in progress, the explorer is not already
   -- landed and there must be a planet with a yellow dwarf in the
   -- current sector where that planet has not been landed on yet.
   -- If there are multiple planets in this sector, land on the one
   -- that has not been landed on yet with the lowest id.
   -- Note that this command will cause a turn to pass/occur.
   -- Asteroid and Malevolent cannot affect the explorer
   -- when it is landed.
   -- After the explorer land, other moveable entities whose clock
   -- time (rest) is zero also act in id order, i.e. 1, 2, ..
   -- In test mode: displays entity actions, abstract state,
   -- then board.

liftoff
   -- Lifts the explorer off a planet.
   -- A game has to be in progress and the explorer is landed
   -- on a planet which also has a yellow dwarf in the same
   -- sector that cannot support life.
   -- The explorer remains in its quadrant, but can now move.
   -- Note that this command will cause a turn to pass/occur.
   -- After the explorer liftoff, other moveable entities whose clock
   -- time (rest) is zero also act in id order, i.e. 1, 2, ..
   -- In test mode: displays entity actions, abstract state,
   -- then board.
```

```
pass
   -- Lets the explorer pass a turn.
   -- Note that this command will cause a turn to pass/occur
   -- and other entities can affect the explorer.
   -- After the explorer pass, other moveable entities whose clock
   -- time (rest) is zero also act in id order, i.e. 1, 2, ..
   -- In test mode: displays entity actions, abstract state,
   -- then board.

wormhole
   -- Tunnels the explorer to a random sector (first open quadrant).
   -- A game has to be in progress and there must be
   -- a wormhole in the current sector.
   -- Note that this command will cause a turn to pass/occur
   -- and other entities can affect the explorer.
   -- After the explorer wormholes, other moveable entities whose
   -- clock time (rest) is zero also acts in id order, i.e. 1, 2, ..
   -- In test mode: displays entity actions, abstract state,
   -- then board.
-------------------------------------------------------------------
-- SimOdyssey Queries
-------------------------------------------------------------------

status
    -- Displays explorer's energy, life and sector.
    -- Note that this command does not cause a turn to pass/occur
```