EECS3311-W20 — Project Report

Submitted electronically by:

Team members	Name	Prism Login	Signature
Member 1:	Jinho Hwang	howden2	[illegible]
Member 2:	Ato Koomson	flashato	[illegible]
*Submitted under Prism account:		howden2	[illegible]

* Submit under **one** Prism account only

Also submit a printed version with signatures in the course Drop Box

Contents

1. Requirements for Project

SimOdyssey is “a galaxy exploration simulator to prepare a new generation for deep space exploration.” [1] The user of the program, the explorer, gets to play in this game to find a life-existing and habitable planet to land on starting from the top left of the galaxy. The galaxy is represented by 25 sectors, with 5 rows and 5 columns, having a blackhole that kills any moveable entities within the same sector in the middle of the galaxy. Each sector is made out of four quadrants, and it works like a space reserved for any entity to reside in; full occupation in quadrants in a sector prevents another moveable entity to get in. The explorer is not alone in the galaxy. There are other moveable entities like benigns, malevolents, janituars, asteroids and planets, and stationary entities like a blackhole, wormholes, yellow dwarves, and blue giants.

Moveable entities do interact with other entities within the same sector when they behave in their own way. The explorer gets to move around the galaxy using the move command with a finite fuel that can run out to death and has a humanity-saving-duty of finding a life supporting planet to land. Benign moves around the galaxy to kill malevolent at once while malevolent tries to kill explorer with three hits to the explorer. Janituar cleans asteroids and dumps into the wormhole. All benigns, malevolents, and janituars have limited fuel capacity and reproductive capability. Any fuel storable entities can recharge their fuel by going into a sector of either a blue giant or a yellow dwarf. Asteroids, on the other hand, do not reproduce nor have fuel capacity but fly around the galaxy until it finds another moveable entity including other asteroids and excluding planets and kills it. Planets move through the galaxy and are able to attach themselves into a sector with a blue giant or a yellow dwarf. When the planet attaches to the blue giant or yellow dwarf, the planet gets a 50% chance that they support life. This life-supporting planet is what the explorer tries to land on to finish the simulation successfully. All moveable entities can be consumed by the blackhole in the middle of the galaxy.

Stationary entities stay in the same sector where they were created for the first time, and allow interactions with moveable entities entering the sector. A yellow dwarf and a blue giant allows any moveable entities with fuel to recharge their fuel if they enter the sector with the yellow dwarf or the blue giant. Wormhole allows the explorer, benigns, and melovanants to teleport any sector in the galaxy including where it started and the sector with the blackhole if they do move into the same sector with the wormhole. benigns and melovanants will favouritize taking wormhole, while the explorer has a choice of either to wormhole or not. The blackhole consumes any moveable entity entering the sector with the blackhole.

See the rest of the contents for a detailed description of the software design used in the project.

[1] SimOdyssey2-Spec.pdf page 4.

2. BON class diagram overview (architecture of the design)

2.1 Top Level Architectural BON Class Diagram

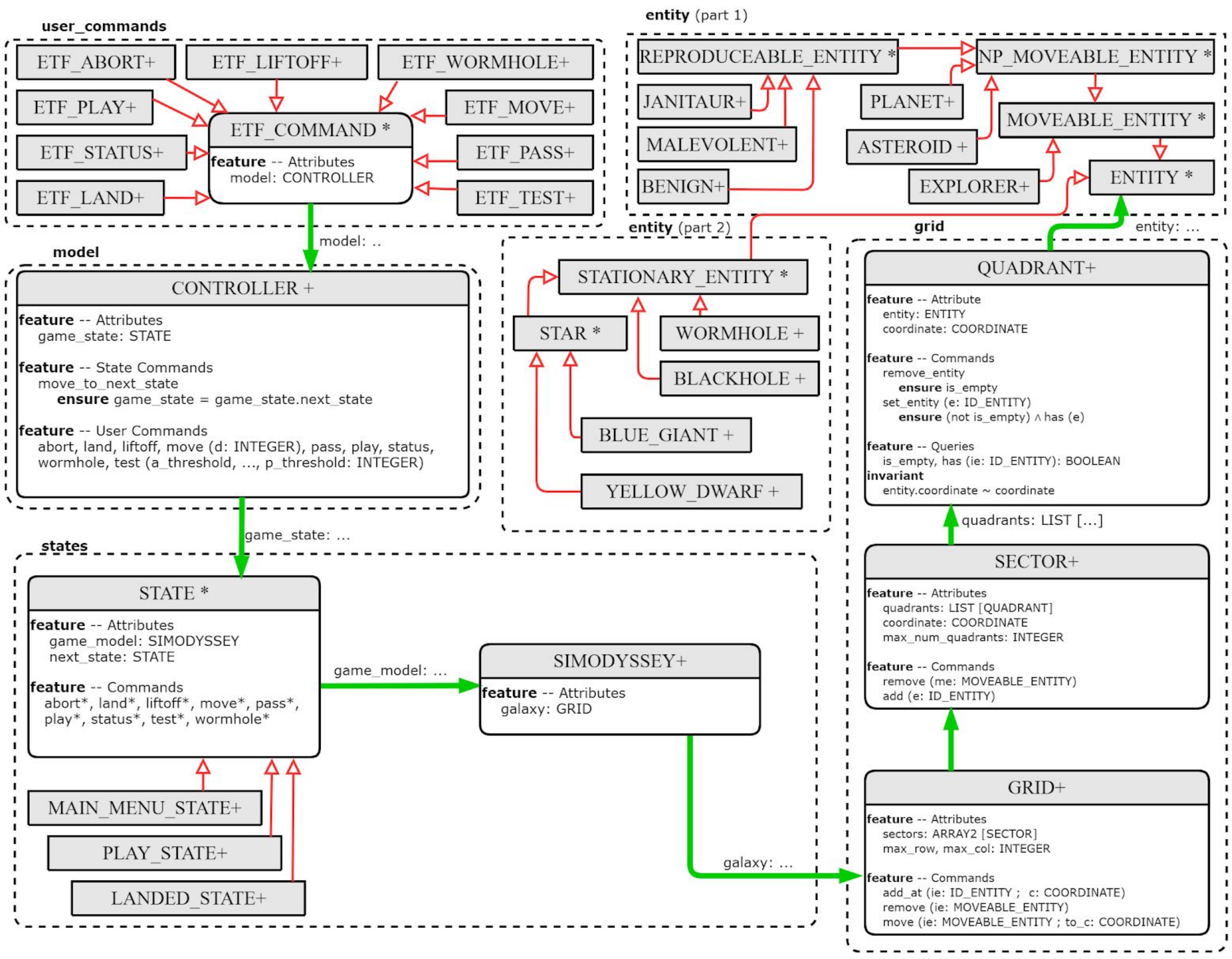
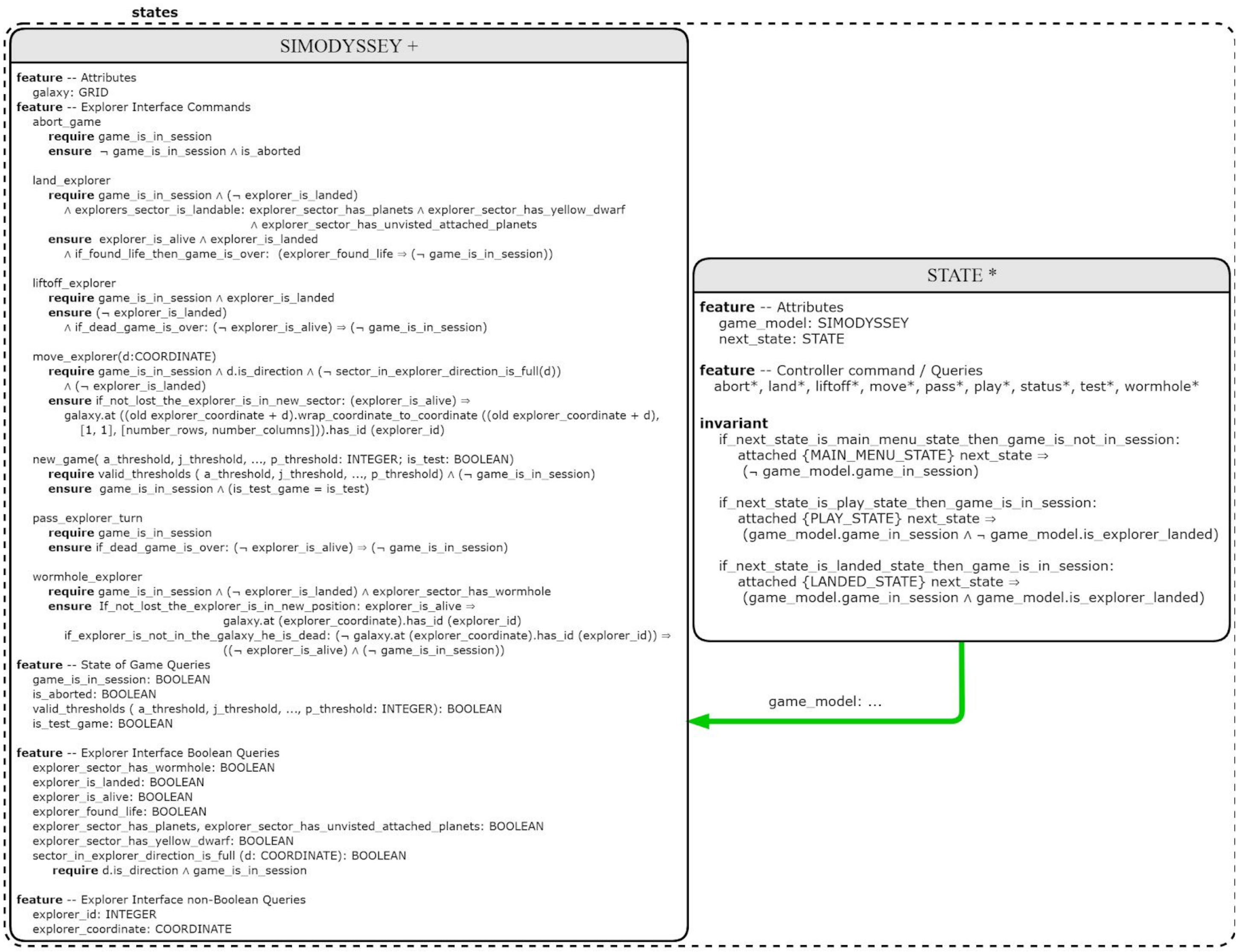


Figure 1: Top level architectural BON class diagram

2.2 Important Class SIMODYSSEY and STATE with their Contracts



```
SIMODYSSEY +

feature -- Attributes
   galaxy: GRID
feature -- Explorer Interface Commands
   abort_game
      require game_is_in_session
      ensure  ¬ game_is_in_session ∧ is_aborted

   land_explorer
      require game_is_in_session ∧ (¬ explorer_is_landed)
         ∧ explorers_sector_is_landable: explorer_sector_has_planets ∧ explorer_sector_has_yellow_dwarf
                                         ∧ explorer_sector_has_unvisted_attached_planets
      ensure  explorer_is_alive ∧ explorer_is_landed
         ∧ if_found_life_then_game_is_over:  (explorer_found_life ⇒ (¬ game_is_in_session))

   liftoff_explorer
      require game_is_in_session ∧ explorer_is_landed
      ensure (¬ explorer_is_landed)
         ∧ if_dead_game_is_over: (¬ explorer_is_alive) ⇒ (¬ game_is_in_session)

   move_explorer(d:COORDINATE)
      require game_is_in_session ∧ d.is_direction ∧ (¬ sector_in_explorer_direction_is_full(d))
         ∧ (¬ explorer_is_landed)
      ensure if_not_lost_the_explorer_is_in_new_sector: (explorer_is_alive) ⇒
         galaxy.at ((old explorer_coordinate + d).wrap_coordinate_to_coordinate ((old explorer_coordinate + d),
            [1, 1], [number_rows, number_columns])).has_id (explorer_id)

   new_game( a_threshold, j_threshold, ..., p_threshold: INTEGER; is_test: BOOLEAN)
      require valid_thresholds ( a_threshold, j_threshold, ..., p_threshold) ∧ (¬ game_is_in_session)
      ensure  game_is_in_session ∧ (is_test_game = is_test)

   pass_explorer_turn
      require game_is_in_session
      ensure if_dead_game_is_over: (¬ explorer_is_alive) ⇒ (¬ game_is_in_session)

   wormhole_explorer
      require game_is_in_session ∧ (¬ explorer_is_landed) ∧ explorer_sector_has_wormhole
      ensure  If_not_lost_the_explorer_is_in_new_position: explorer_is_alive ⇒
                                  galaxy.at (explorer_coordinate).has_id (explorer_id)
         if_explorer_is_not_in_the_galaxy_he_is_dead: (¬ galaxy.at (explorer_coordinate).has_id (explorer_id)) ⇒
                                  ((¬ explorer_is_alive) ∧ (¬ game_is_in_session))
feature -- State of Game Queries
   game_is_in_session: BOOLEAN
   is_aborted: BOOLEAN
   valid_thresholds ( a_threshold, j_threshold, ..., p_threshold: INTEGER): BOOLEAN
   is_test_game: BOOLEAN

feature -- Explorer Interface Boolean Queries
   explorer_sector_has_wormhole: BOOLEAN
   explorer_is_landed: BOOLEAN
   explorer_is_alive: BOOLEAN
   explorer_found_life: BOOLEAN
   explorer_sector_has_planets, explorer_sector_has_unvisted_attached_planets: BOOLEAN
   explorer_sector_has_yellow_dwarf: BOOLEAN
   sector_in_explorer_direction_is_full (d: COORDINATE): BOOLEAN
       require d.is_direction ∧ game_is_in_session

feature -- Explorer Interface non-Boolean Queries
   explorer_id: INTEGER
   explorer_coordinate: COORDINATE
```

```
STATE *

feature -- Attributes
   game_model: SIMODYSSEY
   next_state: STATE

feature -- Controller command / Queries
  abort*, land*, liftoff*, move*, pass*, play*, status*, test*, wormhole*

invariant
   if_next_state_is_main_menu_state_then_game_is_not_in_session:
      attached {MAIN_MENU_STATE} next_state ⇒
       (¬ game_model.game_in_session)

   if_next_state_is_play_state_then_game_is_in_session:
      attached {PLAY_STATE} next_state ⇒
       (game_model.game_in_session ∧ ¬ game_model.is_explorer_landed)

   if_next_state_is_landed_state_then_game_is_in_session:
      attached {LANDED_STATE} next_state ⇒
       (game_model.game_in_session ∧ game_model.is_explorer_landed)
```

Figure 2: The two most important classes in the design: SIMODYSSEY and STATE

2.3 Overall Design and the Main Design Decisions

2.3.1 grid Cluster

The grid cluster contains classes used to represent the five by five size galaxy and how to locate, add, delete, and move entities in the galaxy. It is organized in GRID, SECTOR, and QUADRANT classes to represent the galaxy in a hierarchical fashion; GRID contains 25 sectors, each SECTOR contains 4 QUADRANT, and a QUADRANT contains an entity. Each class takes care of only what they own. They each manage addition, removal, checking various status of what they own; for example, GRID allows the movement of an entity by obtaining the entity among the SECTOR the GRID owns, removing it, and adding it to a destination SECTOR.

2.3.2 entity Cluster

The entity cluster contains classes to represent each type of entity in the game. Rather than build each entity type from scratch, classes in the entity cluster are organized in hierarchical fashion, where entity types are built via inheritance relationships (i.e. using the foundation of less capable entity types to build more capable entity types). In the entity cluster, the most basic entity type is ENTITY, and it encapsulates an entity's character and coordinate. Through inheritance of ENTITY, more complex entity types such as ID_ENTITY are also built to encapsulate an entity's character, coordinate and id. Because complex entity types need not be built from scratch and can be built by inheritance of basic entity types, the design decision to organize the entity cluster via inheritance relationships, provides reusability, and avoids code duplication.

2.3.3 model Cluster

The model cluster has a class named CONTROLLER. This is formerly ETF_MODEL and is the entry point of ETF_COMMAND's execution. CONTROLLER uses STATE and its concrete children's implementation to execute commands.

2.3.4 states Cluster

The states cluster contains classes to represent the current state of the execution. There are STATE and its children MAIN_MENU_STATE, PLAY_STATE, LANDED_STATE intuitively representing what state they describe, and SIMODYSSEY has a galaxy object typed GRID, which deals with the status of entities.

2.3.5 utility Cluster

The utility cluster contains classes COORDINATE, DIRECTION_UTILITY, MESSAGE, ID_DISPATCHER. COORDINATE is used to encapsulate the coordinate attributes of ENTITY, QUADRANT, and SECTOR. DIRECTION_UTILITY is used to encapsulate direction related queries. MESSAGE is used for the generation of Abstract State messages. ID_DISPATCHER is used for generating unique entity ids in SIMODYSSEY.

2.3.6 starter Cluster

The starter Cluster contains the RANDOM_GENERATOR and its access given in initial code.

3. Table of modules — responsibilities and information hiding

3.1 grid Cluster

1	GRID	**Responsibility:** A collection of SECTOR objects arranged in a 2-D grid.	**Alternative:** ID_ENTITY inherits from COMPARABLE and "is_less" is implemented by comparing the id of other and current. Implement features "all_moveable_entities" query and "all_stationary_entities" query using SORTED_LIST. The efficiency of this design depends on the efficiency of SORTED_LIST[G].
	Concrete	**Secret:** The collection of STATIONARY_ENTITY in the GRID is stored in a HASH_TABLE, to allow efficient implementation of "all_stationary_entities" query. A similar approach is used to implement the "all_moveable_entities" query.	
1.1	SECTOR	**Responsibility:** A collection of QUADRANT objects arranged in a LIST.	**Alternative:** Use FIXED_LIST instead of ARRAYED_LIST to store the collection of QUADRANT. The invariant "min_max_count" in SECTOR is similar to "extendible" query in FIXED_LIST[G]. Therefore, the invariant "min_max_count" in SECTOR would be redundant using this design.
	Concrete	**Secret:** The collection of QUADRANT is stored in an ARRAYED_LIST.	
1.1.1	QUADRANT	**Responsibility:** A container for storing an ENTITY in a SECTOR.	**Alternative:** "entity" attribute is now of type detachable ENTITY. Now QUADRANT. "is_empty" = true, implies "entity" attribute refers to void.
	Concrete	**Secret:** QUADRANT. "is_empty" = true, implies "entity" attribute refers to a NULL_ENTITY.	

3.2 entity Cluster

1	DEATHABLE	**Responsibility:** A class that encapsulates common queries, attributes, and commands for entities capable of death. (e.g. MOVEABLE_ENTITY)	**Alternative:** Directly implement LIFE in DEATHABLE such that LIFE and DEATHABLE combine into a single class. This single class is now responsible for handling the internal bugs and behavior associated with both LIFE and DEATHABLE classes. Cohesion and encapsulation principles are broken as a result. & Use a HASH_TABLE to store all valid death causes. The efficiency of "is_valid_death_cause" query is made faster due to the look-up efficiency of hashable items (e.g STRING).
	Abstract	**Secret:** Private attribute "life" is of type LIFE which means DEATHABLE is a client of LIFE. & The collection of all valid death causes is stored in an ARRAY.	
1.1	LIFE	**Responsibility:** A class that encapsulates DEATHABLE's life.	**Alternative:** (see DEATHABLE alternative)
	Concrete	**Secret:** none	
2	FUELABLE	**Responsibility:** A class that encapsulates common queries, attributes, and commands for entities with fuel.	**Alternative:** Remove the inheritance relationship between FUELABLE and subclasses of FUELABLE and replace it with a client-supplier relationship (similar to the relationship between DEATHABLE and LIFE). Now, previous subclasses of FUELABLE require their own implementation of the FUELABLE interface. Across all previous subclasses of FUELABLE, the signature and implementation of these interfaces will be identical, resulting in code duplication.

	Abstract	**Secret:** none	
3	ENTITY	**Responsibility:** A class to represent an entity in a QUADRANT.	**Alternative:** ENTITY not only has attributes “character” and “coordinate” but also “id”. This removes the necessity for NULL_ENTITY and ID_ENTITY. Implementing this design also requires implementation of QUADRANT’s alternative. (see QUADRANT alternative)
	Abstract	**Secret:** none	
3.1	NULL_ENTITY	**Responsibility:** A class to represent the absence of an ENTITY.	**Alternative:** (see ENTITY alternative.)
	Concrete	**Secret:** (see QUADRANT secret).	
3.2	ID_ENTITY	**Responsibility:** A class to represent an ENTITY and its identification number.	**Alternative:** (see ENTITY alternative.)
	Abstract	**Secret:** none	
3.2.1	MOVEABLE_ENTITY	**Responsibility:** A class to represent an ID_ENTITY that can change its coordinate and is capable of death.	**Alternative:** Remove MOVEABLE_ENTITY from the design and implement the interface once part of MOVEABLE_ENTITY in all previous subclasses of MOVEABLE_ENTITY. This results in code duplication.
	Abstract	**Secret:** none	

3.2.1.1	EXPLORER	**Responsibility:** A class to represent the explorer entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.1.2	NP_MOVEABLE_ENTITY	**Responsibility:** A class to represent a MOVEABLE_ENTITY whose actions occur in defined intervals and whose actions cannot be explicitly controlled via user commands. Note: NP stands for NON_PLAYABLE	**Alternative:** Attempt MOVEABLE_ENTITY alternative (see MOVEABLE_ENTITY alternative) with NP_MOVEABLE_ENTITY. This results in code duplication.
	Abstract	**Secret:** none	
3.2.1.2.1	PLANET	**Responsibility:** A class to represent a planet entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.1.2.2	ASTEROID	**Responsibility:** A class to represent an asteroid entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.1.2.3	REPRODUCEABLE_ENTITY	**Responsibility:** A class to represent an NP_MOVEABLE_ENTITY that can reproduce.	**Alternative:** none
	Abstract	**Secret:** none	
3.2.1.2.3.1	BENIGN	**Responsibility:** A class to represent a benign entity.	**Alternative:** none
	Concrete	**Secret:** none	

3.2.1.2.3.2	JANITAUR	**Responsibility:** A class to represent a janitaur entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.1.2.3.3	MALEVOLENT	**Responsibility:** A class to represent a malevolent entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.2	STATIONARY_ENTITY	**Responsibility:** A class to represent an ID_ENTITY that is not also MOVEABLE_ENTITY.	**Alternative:** Remove STATIONARY_ENTITY from the design so that previous subclasses of STATIONARY_ENTITY now inherit directly from ID_ENTITY. Now referring to an ID_ENTITY that is not also a MOVEABLE_ENTITY requires code that is verbose. For example, referring to a STATIONARY_ENTITY requires code like "not attached {MOVEABLE_ENTITY}" in contrast to "attached {STATIONARY_ENTITY}".
	Abstract	**Secret:** none	
3.2.2.1	BLACKHOLE	**Responsibility:** A class to represent a blackhole entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.2.2	WORMHOLE	**Responsibility:** A class to represent a wormhole entity.	**Alternative:** none
	Concrete	**Secret:** none	

3.2.2.3	STAR	**Responsibility:** A class to represent a STATIONARY_ENTITY and its luminosity value.	**Alternative:** Remove STAR from the design completely. This results in code duplication between all current descendants of STAR. Similar in effect to (see 3.2.1 alternative)
	Abstract	**Secret:** none	
3.2.2.3.1	YELLOW_DWARF	**Responsibility:** A class to represent a yellow_dwarf entity.	**Alternative:** none
	Concrete	**Secret:** none	
3.2.2.3.2	BLUE_GIANT	**Responsibility:** A class to represent a blue_giant entity.	**Alternative:** none
	Concrete	**Secret:** none	

3.3 model Cluster

1	CONTROLLER	**Responsibility:** A class that provides an interface for executing all nine user commands and updates the user output when commands are executed.	**Alternative:** none
	Concrete	**Secret:** Attribute "game_state" is of type STATE which means CONTROLLER is a client of STATE. Note: "game_state" is polymorphic. Post executing a command in CONTROLLER, "game_state" transitions (changes its reference) to a subclass of STATE that is appropriate for the game.	

3.4 states Cluster

1	STATE	**Responsibility:** A class that defines valid, invalid user commands, and generates the user's output when commands are executed.	**Alternative:** Implement in STATE, features once encapsulated by ABSTRACT_STATE_NUMBERS. Principle of cohesion/single responsibility is broken as a result.
	Abstract	**Secret:** private attribute "abstract_state_numbers" is of type ABSTRACT_STATE_NUMBERS which means STATE is a client of ABSTRACT_STATE_NUMBERS. "abstract_state_numbers" is updated accordingly after the execution of a command.	
1.1	ABSTRACT_STATE_NUMBERS	**Responsibility:** A class to manage the first and second number as seen in the user output "e.g state: 3.1".	**Alternative:** (see STATE alternative)
	Concrete	**Secret:** none	
1.2	SIMODYSSEY	**Responsibility:** A class that controls the addition, removal,and movement of entities in the galaxy and provides an interface for controlling the explorer's actions in the galaxy. .	**Alternative:** none
	Concrete	**Secret:** Post execution of a command, non-user-controlled entities move, reproduce, behave, and check according to the "turn" command algorithm on page 33 of the project specifications.	
3.1	MAIN_MENU_STATE	**Responsibility:** A class that defines valid, and invalid user commands for when the user is not in a game.	**Alternative:** none
	Concrete	**Secret:** none	

3.2	PLAY_STATE	**Responsibility:** A class that defines valid, and invalid user commands for when the user is in a game, and the explorer is not landed.	**Alternative:** none
	Concrete	**Secret:** none	
3.3	LANDED_STATE	**Responsibility:** A class that defines valid, and invalid user commands for when the user is in a game and the explorer is landed.	**Alternative:** none
	Concrete	**Secret:** none	

3.5 utility Cluster

1	DIRECTION_UTILITY	**Responsibility:** A class that contains common direction COORDINATEs (e.g. N -> [-1,0], E -> [0,1] …)	**Alternative:** When a direction vector is needed, create the direction manually using COORDINATE's make routine. Higher probability for error (i.e. creating a direction that was not intended)
	Concrete	**Secret:** none	
1.1	COORDINATE	**Responsibility:** A class to represent comparable coordinates.	**Alternative:** none
	Concrete	**Secret:** none	
2	ID_DISPATCHER	**Responsibility:** A class for generating unique entity ids.	**Alternative:** none
	Concrete	**Secret:** none	
3	MESSAGE	**Responsibility:** A class for generating Abstract State messages.	**Alternative:** Generate Abstract State messages manually in SIMODYSSEY or STATE. The responsibility of

	Concrete	**Secret:** none	SIMODYSSEY/STATE becomes unfocused and cohesion principle is broken as a result.

3.6 starter Cluster

1	RANDOM_GENERATOR_ACCESS	**Responsibility:** Singleton for accessing RANDOM_GENERATOR.	**Alternative:** none
	Concrete	**Secret:** none	
1.1	RANDOM_GENERATOR	Responsibility: A class used to generate random numbers using the same seed (deterministically).	**Alternative:** none
	Concrete	**Secret:** none	

4. Expanded description of design decisions

The module to expand and describe is the states cluster. The states cluster has the following structure:

4.1 BON Class Diagram for states Cluster

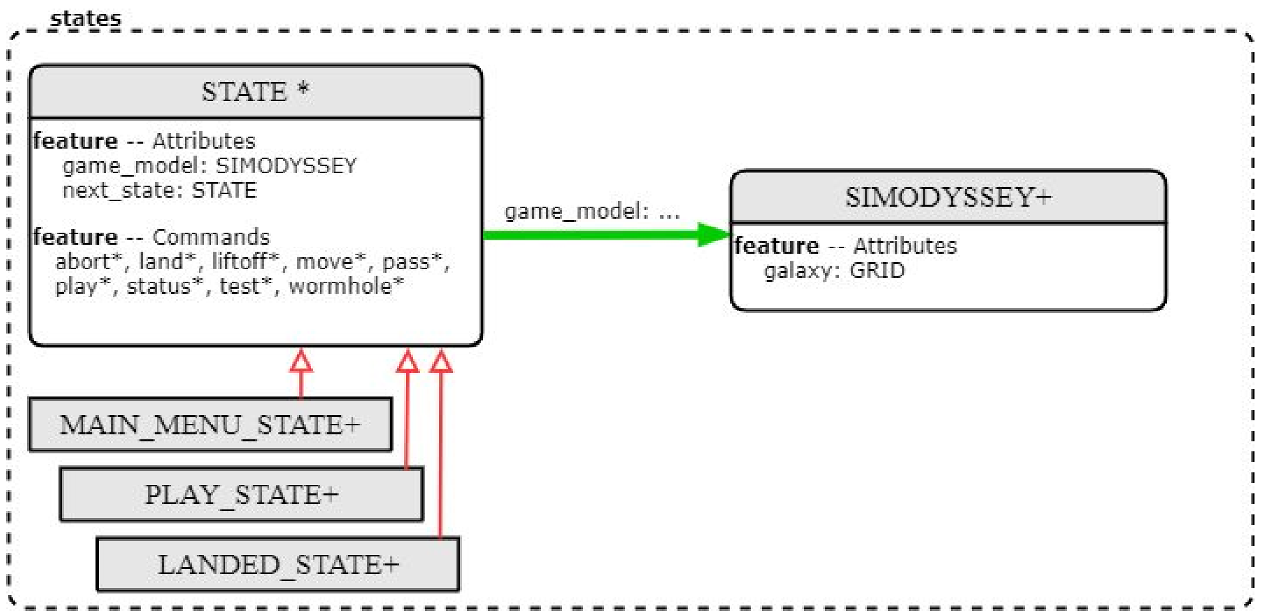


Figure 3: BON Class Diagram for states Cluster

4.2 Class Description and its Responsibilities

The design decision made for this cluster was to use state-panel design pattern to accommodate different STATE's out query which represents the console text output on the console depending on what states it currently is in.

Using the state-panel design allows a

1. More coherent code, since every implementation of the commands which depends on which state they are in, are encapsulated by the concrete children classes.
2. Easy maintainability of semantic validity of commands in each situation because it is very intuitive to see that the commands on each of the children classes' represents the commands run in those concrete states, therefore allowing easier fixes to be found rather than going into many different classes to figure one out.

How the state-panel design is implemented will be described below.

4.2.1 STATE

The deferred STATE class represents the current state of the SimOdyssey program. It is the supplier to the CONTROLLER class in the model cluster (formally ETF_MODEL), and the STATE class variable is denoted as game_state in CONTROLLER class. The CONTROLLER class uses the commands provided in STATE, where the behaviour of each command depends on the dynamic binding of either of the concrete children classes. For example, when the MAIN_MENU_STATE object is bounded to game_state in CONTROLLER class, the MAIN_MENU_STATE's commands will be called (for instance, **start**), changing appropriate text query **out** to represent the output of the current state's abstract message specified in a detailed simodyssey2 specification. The concrete children deals with semantic correctness of command usage; for example, when a user tries to play while they are in PLAY_STATE, an appropriate abstract message will be set so it can be used in CONTROLLER class so it can display the message on the ETF console output.

A state's command may change what the next state is; for example, if a user types **play**, then the game_state must change to the instance of PLAY_STATE, and if a user types **land** in a landable planet on PLAY_STATE, the game_state must change to LANDED_STATE. This is dealt by letting the STATE have next_state, initialized as Current, later potentially changed by some concrete commands, and next_state becomes the game_state every after execution of STATE commands by CONTROLLER. This ensures that whenever next_state is changed by some condition in commands or not, the CONTROLLER correctly changes the game_state to next_state.

4.2.2 SIMODYSSEY

The concrete SIMODYSSEY represents the "model" of the state; it contains all the information about the galaxy. Whenever the STATE calls its abstract commands dynamically bounded by the concrete children's commands, the command checks whether changes about to be made by command is valid one or not; If the changes are valid or invalid, the STATE will output the expected output and set next_state appropriately.

For instance, **move(d: COORDINATE)** command in PLAY_STATE will first check in SIMODYSSEY whether moving in the direction given by the argument **d** is a valid one or not. If invalid, report the invalidity, if valid, apply the actual move to the explorer in SIMODYSSEY to the direction, and check what happened to the explorer. It could have died, attacked, and/or charged fuel, so report these changes from STATE to the abstract message out, **out** query in STATE. SIMODYSSEY also shows the galaxy as its **out** query, and is used by STATE.

4.3 The trade-offs

Implementing state-panel pattern at first is not very intuitive and hard to structure therefore harder to start and build from scratch, compared with brute force implementation that is not to use state-panel pattern. However, using this pattern is more scalable and easier to fix when there is one. In fact, several abstract message errors were found and fixed easily because of the design choice of implementing a state-panel design pattern.

5. Significant Contracts (Correctness)

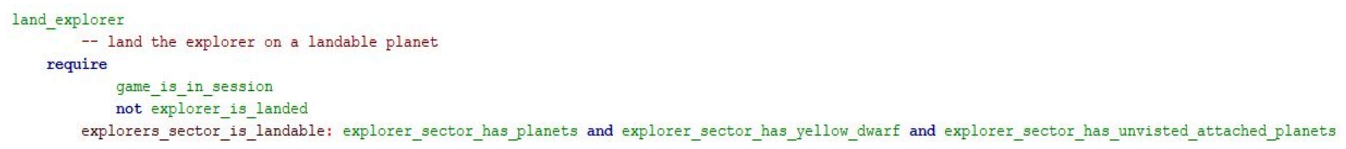
The states cluster module consists of classes SIMODYSSEY, STATE and subclasses of STATE (i.e PLAY_STATE, MAIN_MENU_STATE, and LANDED_STATE). Contracts in SIMODYSSEY are significant to the design, because they define when the execution of a user command will be successful , and when the execution of a user command will be unsuccessful. Additionally, contracts in STATE subclasses are also significant to the design because they specify the set of user commands that will be valid after the successful execution of a user command. Overall, contracts in the states module are the most significant in the design of SimOdyssey2 because they define when user commands will be successfully executed, when the execution of a user commands will result in an error message, and dictate the set of valid user commands after the successful execution of a user command.

Firstly, preconditions for commands in SIMODYSSEY are significant to the design because they directly define the required conditions needed for the successful execution of a user command. For example, the precondition for “land_explorer” command in SIMODYSSEY dictates all conditions that must be satisfied in order for the user to successfully execute “land” command in the user interface. The preconditions for “land_explorer” (see figure 4) specify that a user can successfully execute “land” via the user interface, only when the user is currently in a game, the explorer entity is not already landed, the sector where the explorer is located occupies planet(s) entities, the sector where the explorer is located occupies a yellow dwarf entity, and the sector where the explorer is located occupies uninvited , attached planet entities. Essentially, preconditions for all commands in SIMODYSSEY such as “land_explorer” define when the execution of a user command (e.g “land”) will be successful.

In addition to specifying when user commands can be executed successfully, the preconditions for commands in SIMODYSSEY also indirectly specify all conditions where the attempt to execute a user command will result in an error message (specifically an Abstract State: error message) in the user output. Error conditions for a command are defined by the negation of the command’s precondition. Therefore through demorgan’s law, the error conditions for “land_explorer” command in SIMODYSSEY are “not game_is_in_session”, or “explorer_is_landed”, or “not explorer_sector_has_planets”, or “not explorer_sector_has_yellow_dwarfs”, or “not explorer_sector_has_unvisited_attached_planets”. Furthermore, these specific error conditions regarding “land_explorer” are used in STATE subclasses to systematically determine, when Abstract State: error messages should be displayed in the user output.

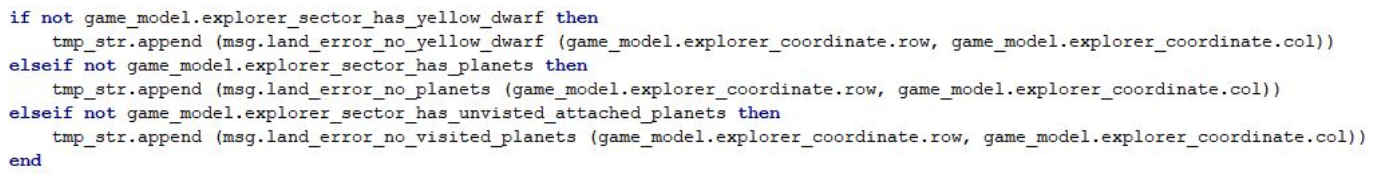
By examining the implementation of “land” command in PLAY_STATE (see Figure 5), one can see that executing the “land” command in PLAY_STATE may result in the production of an error message in the output if it is the case that at least one of the last three error conditions of "land_explorer" (e.g "not explorer_sector_has_unvisited_attached_planets") is true.

Overall, the approach used above, to demonstrate the significance of "land_explorer"'s preconditions in the design of SIMODYSSEY can be used to prove the same for all interface commands in SIMODYSSEY. Therefore the preconditions for commands in SIMODYSSEY are the most significant to the design of SimOdyssey2 because they directly define when user commands can be executed successfully and indirectly define when the execution of a user commands will result in an error message.



```
land_explorer
        -- land the explorer on a landable planet
    require
            game_is_in_session
            not explorer_is_landed
        explorers_sector_is_landable: explorer_sector_has_planets and explorer_sector_has_yellow_dwarf and explorer_sector_has_unvisted_attached_planets
```

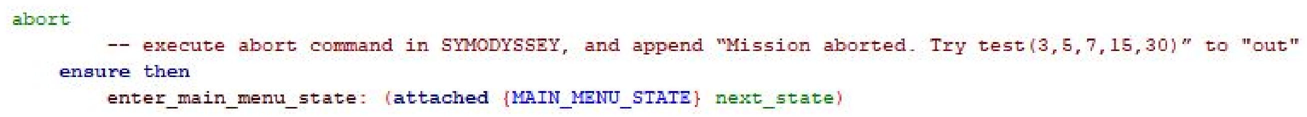
Figure 4: Preconditions of land_explorer command in SIMODYSSEY



```
if not game_model.explorer_sector_has_yellow_dwarf then
    tmp_str.append (msg.land_error_no_yellow_dwarf (game_model.explorer_coordinate.row, game_model.explorer_coordinate.col))
elseif not game_model.explorer_sector_has_planets then
    tmp_str.append (msg.land_error_no_planets (game_model.explorer_coordinate.row, game_model.explorer_coordinate.col))
elseif not game_model.explorer_sector_has_unvisted_attached_planets then
    tmp_str.append (msg.land_error_no_visited_planets (game_model.explorer_coordinate.row, game_model.explorer_coordinate.col))
end
```

Figure 5:Snippet of the implementation of "land" command in PLAY_STATE

In addition to the significance of SIMODYSSEY contracts to the design of SimOdyssey2, post conditions for commands in STATE subclasses (i.e PLAY_STATE, MAIN_MENU_STATE and LANDED_STATE) are also significant to the design because they dictate the set of all valid user commands after the successful execution of a user command.According to its abstraction the responsibility of STATE is to define the valid, invalid user commands, and generate the user's output when commands are executed. Being said, the abort command in LANDED_STATE (see Figure 6) means that MAIN_MENU_STATE is the next state that "game_state" attribute in CONTROLLER will be transitioned (change reference) to after the execution of abort. Therefore the post condition of abort in LANDED_STATE clearly defines the next set of valid commands that the user can execute after executing abort in the LANDED_STATE. Essentially, all post conditions in PLAY_STATE, MAIN_MENU_STATE and LANDED_STATE dictate the next set of valid user commands in a similar manner. Therefore all post conditions in STATE subclasses are important to the design of SimOdyssey2 because they dictate the set of valid user commands after the successful execution of a user command.



```
abort
        -- execute abort command in SYMODYSSEY, and append "Mission aborted. Try test(3,5,7,15,30)" to "out"
    ensure then
        enter_main_menu_state: (attached {MAIN_MENU_STATE} next_state)
```

Figure 6: Postcondition of abort command in LANDED_STATE

6. Summary of Testing Procedures

6.1 Table of all the Acceptance Tests

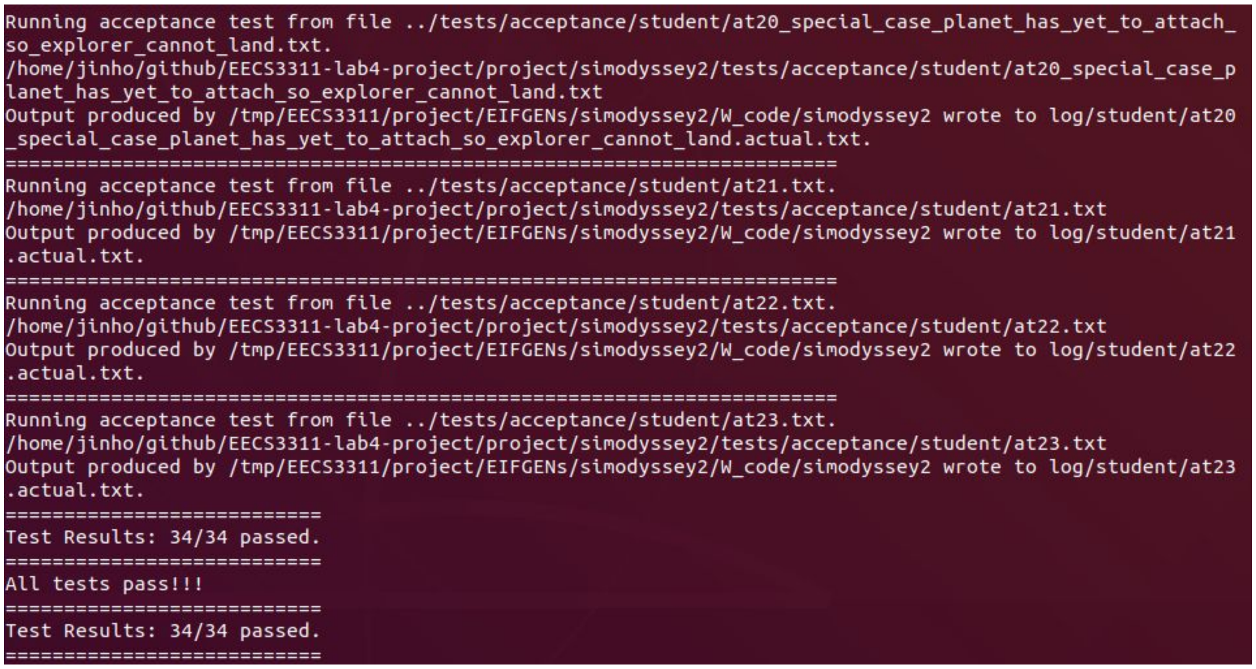
Test file	**Description**	**Passed**
at00_abstract_state_initial_message.txt	Display **Abstract State: Command-Specific Messages: Initial Message (1)** in the user's output.	Yes
at01_abstract_state_status.txt	Display Abstract State: Command-Specific Messages: **status (1 and 2)** in the user's output.	Yes
at02_abstract_state_land_and_liftoff.txt	Display Abstract State: Command-Specific Messages: **land (1 and 2)** in the user's output. Display Abstract State: Command-Specific Messages: **liftoff (1)** in the user's output.	Yes
at03_abstract_state_abort.txt	Display Abstract State: Command-Specific Messages: **abort (1)** in the user's output.	Yes
at04_aborting_while_landed.txt	Display Abstract State: Command-Specific Messages: **abort (1)** in the user's output. **(specifically, while explorer is landed)**	Yes
at05_abstract_state_game_is_over_and_abstract_death_messages_explorer_1_to_4.txt	Display Abstract State: **Command-Specific Messages: game is over (1)** in the user's output. Displays Abstract State: **Death Messages: EXPLORER (1,2,3 and 4)** in the user output.	Yes
at05_abstract_death_messages_explorer_part2.txt	Displays Abstract State: Death Messages: **EXPLORER (2)** in the user output. (**specifically, for the case** when explorer dies after wormhole command is executed by user)	Yes
at06_abstract_death_messages_benign.txt	Displays Abstract State: Death Messages: **BENIGN (1 and 2)** in the user output.	Yes
at06_abstract_death_messages_benign_part_2.txt	Displays Abstract State: Death Messages: **BENIGN (3)** in the user output.	Yes

at07_abstract_death_messages_malevolent.txt	Displays Abstract State: Death Messages: **MALEVOLENT (1,2 and 3)** in the user output.	Yes
at07_abstract_death_messages_malevolent_part2.txt	Displays Abstract State: Death Messages: **MALEVOLENT (4)** in the user output.	Yes
at08_abstract_death_messages_janitaur.txt	Displays Abstract State: Death Messages: **JANITAUR (1 and 2)** in the user output.	Yes
at08_abstract_death_messages_janitaur_part2_and_asteroid.txt	Displays Abstract State: Death Messages: **JANITAUR (3) and ASTEROID (2)** in the user output.	Yes
at08_abstract_death_messages_asteroid_part2.txt	Displays Abstract State: Death Messages: **ASTEROID (1)** in the user output.	Yes
at09_abstract_death_messages_planet.txt	Displays Abstract State: Death Messages: **PLANET (1)** in the user output.	Yes
at10_abstract_state_error_messages_no_mission_in_progress.txt	Displays **Abstract State: Error Messages: ABORT (1), LAND (1), LIFTOFF (1), MOVE (1), PASS (1), STATUS (1) and WORMHOLE (1)** in the user output.	Yes
at11_abstract_state_error_messages_land_and_liftoff_and_wormhole_and_move.txt	Displays Abstract State: Error Messages: **LAND (2,3,4,5), LIFTOFF (2), MOVE (2) and WORMHOLE (2 and 3)** in the user output.	Yes
at11_abstract_state_error_messages_move.txt	Displays Abstract State: Error Messages: **MOVE (3)** in the user output.	Yes
at12_abstract_error_messages_play_and_test.txt	Displays Abstract State: Error Messages: **PLAY (1) and TEST (1 and 2)** in the user output.	Yes
at13_testing_commands_by_state_landed.txt	**Confirming commands** which should be invalid (land, move, test, wormhole, play) /valid (status, pass, liftoff, abort) while the explorer is landed, respond appropriately.	Yes

at14_testing_commands_by_state_play_and_test.txt	Confirming commands which should be invalid (liftoff, test, play) /valid (pass, status, abort) while the explorer is not landed, respond appropriately.	Yes
at15_actions_and_consequences_move_wormhole_and_fuel.txt	Confirming commands (move, pass, wormhole) which may affect (increase or decrease) the explorer's fuel respond appropriately.	Yes
at15_actions_and_consequences_move_land_liftoff_and_fuel.txt	Confirming commands (land, move, liftoff) which may affect (increase or decrease) the explorer's fuel respond appropriately.	Yes
at16_npc_reproduction.txt	**Confirming ENTITYs** Janitaur, Benign and Malevolent produce accordingly.	Yes
at17_npc_wormholeing_malevolent_and_benign.txt	Confirming ENTITYs Malevolent and Benign entities prefer wormhole-ing if there exists a wormhole in their sector.	Yes
at18_npc_load_increment_and_load_clearing.txt	**Confirming** a Janitaur's load increments appropriately, and a Janitaur's load resets to zero if there exists a wormhole in its sector.	Yes
at19_interesting_case_explorer_is_killed_and_his_sector_becomes_full.txt	**Interesting case** where the pass command is executed, the explorer dies, and its sector is filled with non explorer entities.	Yes
at19_interesting_case_wormholing_into_same_sector.txt	Interesting case where executing wormhole command will wormhole the explorer, back into its current sector.	Yes
at19_interesting_case_multiple_planets_death_case.txt	Interesting case where multiple planets die after a turn command.	Yes
at19_interesting_case_wormhole_and_blackhole_death.txt	Interesting case where explorer wormholes into the blackhole.	Yes
at20_special_case_planet_has_yet_to_attach_so_explorer_cannot_land.txt	Confirming **a special case** that when a planet has yet to attach to a yellow star in its sector, the explorer cannot land on the planet.	Yes
at21.txt	**Test** a winning condition in test mode.	Yes

at22.txt	Test a winning condition in play mode.	Yes
at23.txt	Test losing condition in test mode (lose by out of life).	Yes

6.2 A Screenshot of the Result from Running the Above Set of Regression Tests



```
Running acceptance test from file ../tests/acceptance/student/at20_special_case_planet_has_yet_to_attach_
so_explorer_cannot_land.txt.
/home/jinho/github/EECS3311-lab4-project/project/simodyssey2/tests/acceptance/student/at20_special_case_p
lanet_has_yet_to_attach_so_explorer_cannot_land.txt
Output produced by /tmp/EECS3311/project/EIFGENs/simodyssey2/W_code/simodyssey2 wrote to log/student/at20
_special_case_planet_has_yet_to_attach_so_explorer_cannot_land.actual.txt.
=================================================================
Running acceptance test from file ../tests/acceptance/student/at21.txt.
/home/jinho/github/EECS3311-lab4-project/project/simodyssey2/tests/acceptance/student/at21.txt
Output produced by /tmp/EECS3311/project/EIFGENs/simodyssey2/W_code/simodyssey2 wrote to log/student/at21
.actual.txt.
=================================================================
Running acceptance test from file ../tests/acceptance/student/at22.txt.
/home/jinho/github/EECS3311-lab4-project/project/simodyssey2/tests/acceptance/student/at22.txt
Output produced by /tmp/EECS3311/project/EIFGENs/simodyssey2/W_code/simodyssey2 wrote to log/student/at22
.actual.txt.
=================================================================
Running acceptance test from file ../tests/acceptance/student/at23.txt.
/home/jinho/github/EECS3311-lab4-project/project/simodyssey2/tests/acceptance/student/at23.txt
Output produced by /tmp/EECS3311/project/EIFGENs/simodyssey2/W_code/simodyssey2 wrote to log/student/at23
.actual.txt.
=========================
Test Results: 34/34 passed.
=========================
All tests pass!!!
=========================
Test Results: 34/34 passed.
=========================
```

Figure 7: The regression test result

7. Appendix A (Contract view of all classes)

7.1 grid Cluster

7.1.1 GRID class

```
note
        description: "[
                A collection of SECTOR objects arranged in a 2-D grid.

                Secret:
                The collection of STATIONARY_ENTITY in the GRID
                is stored in a HASH_TABLE, to allow efficient
                implementation of “all_stationary_entities” query.
                A similar approach is used to implement
                “all_moveable_entities” query.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"


class interface
        GRID


create
        make


feature -- Attributes


        sectors: ARRAY2 [SECTOR]


        max_row: INTEGER_32
                        -- maximum number of rows


        max_col: INTEGER_32
                        -- maximum number of columns

feature -- Commands


        add_at (ie: ID_ENTITY; c: COORDINATE)
                        -- add ie to a SECTOR with coordinate ~ c.
                require
                                valid_coordinate (c)
                                not has (ie)
                                not at (c).is_full
                ensure
                        me_is_added_at_correct_sector: at (c).has (ie)
                        count_is_incremented_by_one: entity_count ~ (old entity_count + 1)


        remove (me: MOVEABLE_ENTITY)
                        -- remove me.
                require
                                has (me)
                ensure
                        me_is_removed: not has (me)
                        count_is_decremented_by_one: entity_count ~ (old entity_count - 1)


        move (ie: MOVEABLE_ENTITY; to_c: COORDINATE)
```

```
                    -- move ie away from its SECTOR to a SECTOR in GRID with (coordinate ~ to_c)
                require
                            has (ie)
                            valid_coordinate (to_c)
                    new_sector_is_not_full_or_already_contains_ie: (not at (to_c).is_full or at (to_c).has (ie))
                ensure
                    ie_is_at_new_coordinate: at (to_c).has (ie)
                    count_is_incremented_by_one: entity_count ~ entity_count

feature -- Queries

        entity_count: INTEGER_32
                    -- the culmulative sum of all STATIONARY_ENTITY and MOVEABLE_ENTITY contained

        has_sector (s: SECTOR): BOOLEAN
                    -- result is true if GRID contains SECTOR s.

        all_moveable_entities: ARRAY [MOVEABLE_ENTITY]
                    -- The collection of all MOVEABLE_ENTITY contained; arranged in increasing order ids.

        all_stationary_entities: ARRAY [STATIONARY_ENTITY]
                    -- The collection of all STATIONARY_ENTITY contained; arranged in increasing order ids.

        at (c: COORDINATE): SECTOR
                    -- the SECTOR in grid with coordinate ~ c
                require
                            valid_coordinate (c)
                ensure
                    matching_sector_coordinate: Result.coordinate ~ c
                    result_is_contained_in_grid: has_sector (Result)

        sector_with (ie: ID_ENTITY): SECTOR
                    -- the SECTOR in GRID that contains ie.
                require
                            has (ie)
                ensure
                    result_has_ie: Result.has (ie)
                    result_is_contained_in_grid: has_sector (Result)

        valid_coordinate (c: COORDINATE): BOOLEAN
                    -- true if c lies between [0,0] and [max_row,max_col]

        has (ie: ID_ENTITY): BOOLEAN
                    -- result equals true if "ie" is contained in any SECTOR in GRID

feature -- Traversal

        new_cursor: INDEXABLE_ITERATION_CURSOR [SECTOR]
                    -- facilitate traversal of GRID using across notation

feature -- Out

        out_abstract_sectors: STRING_8
                    -- result -> (below)
                    -- Sectors:
                    --          [1,1]->[0,E],[36,P],[40,P],-
                    --          [1,2]->[3,P],-,[4,P],-
                    --          ..
                    --          [5,5]->[48,P],[32,P],[47,P],[15,P]

        out_abstract_description: STRING_8
```

```
                    -- result -> (below)
                    -- Descriptions:
                    -- [-11,*]->Luminosity:5
                    --                      ..
                    -- [-1,O]->
                    -- [0,E]->fuel:3/3, life:3/3, landed?:F
                    -- [1,P]->attached?:F, support_life?:F, visited?:F, turns_left:0
                    --                      ..

        out: STRING_8
                    -- result -> (below)
                    -- (1:1) (1:2) (1:3) (1:4) (1:5)
                    -- M--- *--- B--- ---- ----
                    -- (2:1) (2:2) (2:3) (2:4) (2:5)
                    -- ---- ---- W--- W--- ----
                    -- (3:1) (3:2) (3:3) (3:4)
                    -- PY-- ---- O---

end -- class GRID
```

7.1.2 SECTOR class

```
note
        description: "[
                A collection of QUADRANT objects arranged in a LIST.

                Secret:
                The collection of QUADRANT is stored in an
                ARRAYED_LIST.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        SECTOR

create
        make_empty

feature -- Attributes

        coordinate: COORDINATE
                        -- coordinate of the SECTOR

        max_num_quadrants: INTEGER_32
                        -- maximum number of quadrants the SECTOR can occupy

feature -- Commands

        remove (me: MOVEABLE_ENTITY)
                        --remove me from the SECTOR. note: only MOVEABLE_ENTITY can be removed once added.
                require
                                has (me)
                ensure
                                not has (me)
                        entities_in_current_sector_are_contained_in_the_old_sector: across
                                        (quadrants).deep_twin is i_q
                                all
                                        attached {ID_ENTITY} i_q.entity as i_q_e implies ((old Current).has (i_q_e))
                                end
                        count_has_decreased_by_one: count = (old count - 1)

        add (e: ID_ENTITY)
                        -- add e to the SECTOR
                require
                        not_full: not is_full
                        not_has_already: not has (e)
                ensure
                                has (e)
                        entities_in_old_sector_remain_in_current_sector: across
                                        (old quadrants).deep_twin is i_q
                                all
                                        attached {ID_ENTITY} i_q.entity as i_q_e implies (Current.has (i_q_e))
                                end
                        count_is_incremented_by_one: count ~ (old count) + 1

feature -- Queries

        find_landable_planet: PLANET
                        -- the PLANET contained in the SECTOR with the lowest id and is_landable.
                require
```

```
                        is_landable
            ensure
                    result_is_landable: Result.attached_to_star and (not Result.visited)
                    result_is_contained_in_sector: has (Result)

        is_equal (other: like Current): BOOLEAN
                    -- Is other attached to an object considered
                    -- equal to current object?

        is_sorted (a: ARRAY [MOVEABLE_ENTITY]): BOOLEAN
                    -- is the collection of MOVEABLE_ENTITY contained in a, arranged in increasing order id?

        quadrants: LIST [QUADRANT]
                    -- the collection of QUADRANTS contained in the SECTOR

        is_landable: BOOLEAN
                    -- does the SECTOR contain landable PLANET(s)?

        has_planet: BOOLEAN
                    -- does the SECTOR contain PLANET(s)?

        new_cursor: INDEXABLE_ITERATION_CURSOR [QUADRANT]
                    -- facilitate the traversal over the SECTOR using "across" notation

        is_full: BOOLEAN
                    -- are all QUADRANTs in the SECTOR occupied with ID_ENTITYs?

        count: INTEGER_32
                    -- the number of ID_ENTITY contained in the SECTOR

        get_stationary_entity: STATIONARY_ENTITY
                    -- the STATIONARY_ENTITY contained in the SECTOR
            require
                        has_stationary_entity

        has (me: ID_ENTITY): BOOLEAN
                    -- does the SECTOR contain me?

        has_id (a_id: INTEGER_32): BOOLEAN
                    -- does the SECTOR contain an ID_ENTITY with id = a_id?

        quadrant_at (me: ID_ENTITY): INTEGER_32
                    -- result -> the numerical position of me's QUADRANT when looking from left to right in the SECTOR
            require
                        has (me)
            ensure
                    valid_position_in_sector: attached {ID_ENTITY} quadrants [Result].entity as id_e and then id_e ~
    me

        has_stationary_entity: BOOLEAN
                    -- does the SECTOR contain a STATIONARY_ENTITY?

        has_star: BOOLEAN
                    -- does the SECTOR contain a STAR?

        has_wormhole: BOOLEAN
                    -- does the SECTOR contain a WORMHOLE?

        has_blackhole: BOOLEAN
```

```
                        -- does the SECTOR contain a BLACKHOLE?

        moveable_entity_count: INTEGER_32
                        -- the number of all MOVEABLE_ENTITY contained in the SECTOR.

        moveable_entities_in_increasing_order: ARRAY [MOVEABLE_ENTITY]
                        -- the collection of all MOVEABLE_ENTITY contained in the SECTOR; arranged in increasing order
id.
                ensure
                        all_moveable_entities_in_result_are_in_current_sector: across
                                        Result is me
                                all
                                        has (me)
                                end
                        result_has_correct_count: Result.count ~ moveable_entity_count
                        result_is_sorted: is_sorted (Result)

feature -- Output

        out_abstract_full_coordinate (me: MOVEABLE_ENTITY): STRING_8
                        -- result -> "[x,y,q]" ie "[2,2,4]" where x and y are coordinate.row/coordinate.col respectively, and q is
                        -- quadrant_at(me).
                require
                                has (me)

        out_abstract_sector: STRING_8
                        -- result -> "[x,y]->[0,E],-,-,[2,P]" where x and y are are coordinate.row/coordinate.col respectively,
                        -- and the remaining text is the out_abstract of each QUADRANT in the SECTOR

        out_coordinate: STRING_8
                        -- result -> "(row:col)" where row and col are coordinate.row and coordinate.col respectively

        out_quadrants: STRING_8
                        -- result -> "E--*" or "----" where each character represents an ENTITY in the SECTOR.

invariant
        min_max_count: 0 <= count and count <= max_num_quadrants
        entity_coordinates_are_same_as_coordinate: across
                        quadrants is i_q
                all
                        i_q.entity.coordinate ~ coordinate
                end

end -- class SECTOR
```

7.1.3 QUADRANT class

```
note
	description: "[
		A container for storing an ENTITY in a SECTOR

		Secret:
		QUADRANT. “is_empty” = true, implies “entity”
		attribute refers to a NULL_ENTITY.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

class interface
	QUADRANT

create
	make_empty

feature -- Attribute

	entity: ENTITY
			-- entity contained in the QUADRANT

	coordinate: COORDINATE
			-- coordinate of the QUADRANT

feature -- Commands

	remove_entity
			-- remove entity's current refference
		require
			cannot_remove_a_stationary_entity: not (attached {STATIONARY_ENTITY} entity)
		ensure
				is_empty

	set_entity (e: ID_ENTITY)
			-- replace entity's current refference with e.
		require
			stationary_entities_cannot_change_coordinate:
				attached {STATIONARY_ENTITY} e implies e.coordinate ~ coordinate
		ensure
				not is_empty
				e.coordinate ~ coordinate
				has (e)

feature -- Queries

	is_empty: BOOLEAN
			-- does QUADRANT contain an ENTITY?

	has (ie: ID_ENTITY): BOOLEAN
			-- does QUADRANT contain ie?

	is_equal (other: like Current): BOOLEAN
			-- Is other attached to an object considered
			-- equal to current object?

feature -- Out
```

```
        out_abstract: STRING_8
                        -- if is_empty, then result -> "-"
                        -- if not is_empty, then result -> "[id, character]"

        out_character: STRING_8
                        -- Result -> "entity.out"

invariant
        entity_coordinate_is_equivelant_to_quadrant_coordinate: entity.coordinate ~ coordinate

end -- class QUADRANT
```

7.2 entity Cluster

7.2.1 ASTEROID class

```
note
        description: "A class to represent an asteroid entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        ASTEROID

create
        make

feature -- Queries

        is_dead_by_janitaur: BOOLEAN
                -- was killed by JANITAUR?

feature -- Commands

        behave (sector: SECTOR)
                -- perform behavior algorithm that pertains to ASTEROID as seen on pg 36 of Project Specification

        kill_by_janitaur (killer_id: INTEGER_32)
        ensure
                is_dead_by_janitaur

feature -- out

        out_death_message: STRING_8
                -- result -> {Abstract State: Death Messages ASTEROID on pg 26-27}

        out_description: STRING_8
                -- result -> "[id, character]->turns_left: N/A or turns_left"

end -- class ASTEROID
```

7.2.2 BENIGN class

```
note
        description: "A class to represent a benign entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        BENIGN

create
        make

feature -- Command

        check_health (sector: SECTOR)
                -- if sector.has_star ~ true recharge the benign's fuel cells
                -- execute kill_by_blackhole if sector.has_blachole ~ true.
                -- execute kill_by_out_of_fuel if "is_out_of_fuel" ~ true.

        behave (sector: SECTOR)
                -- perform behavior algorithm that pertains to BENIGN as seen on pg 36

feature -- Queries

        is_dead_by_out_of_fuel: BOOLEAN
                -- was killed by out of fuel?.

        is_dead_by_asteroid: BOOLEAN
                -- was killed by ASTEROID?

feature -- Commands

        kill_by_out_of_fuel
        require
                fuel = 0
        ensure
                is_dead_by_out_of_fuel

        kill_by_asteroid (killer_id: INTEGER_32)
        ensure
                is_dead_by_asteroid

        reproduce (moveable_id: INTEGER_32): like Current
                -- create another ENTITY of type {like current} with the same coordinate as current.
feature -- Output

        out_death_message: STRING_8
                -- result -> {Abstract State: Death Messages BENIGN on pg 26-27}

        out_description: STRING_8
                -- result -> "[id, character]->fuel:cur_fuel/max_fuel,
        -- actions_left_until_reproduction: c_value / reproduction_interval, turns_left: N/A or turns_left"

end -- class BENIGN
```

7.2.3 BLACKHOLE class

```
note
	description: "A class to represent a blackhole entity."
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

class interface
	BLACKHOLE

create
	make

end -- class BLACKHOLE
```

7.2.4 BLUE_GIANT class

```
note
	description: "A class to represent a blue_giant entity."
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

class interface
	BLUE_GIANT

create
	make

end -- class BLUE_GIANT
```

7.2.5 DEATHABLE class

```
note
        description: "[
        A class that encapsulates common queries, attributes,
        and commands for entities capable of death.
        (e.g. MOVEABLE_ENTITY)

        Secret:
        Private attribute “life” is of type LIFE which means
        DEATHABLE is a client of LIFE.
        The collection of all valid death causes is stored
        in an ARRAY.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

deferred class interface
        DEATHABLE

feature -- Constructor

        make (a_max_life: INTEGER_32)

feature -- Attribute

        current_life_point: INTEGER_32

        max_life_point: INTEGER_32
                -- maximum value of current_life_point

feature -- Queries

        is_dead: BOOLEAN

        is_alive: BOOLEAN
        ensure
                Result = (not is_dead)

        is_valid_death_cause (a_death_cause: STRING_8): BOOLEAN
                -- is a_death_cause a valid death cause to use to execute kill_by

        get_death_cause: STRING_8
                -- a string that describes the cause for death
        require
                is_dead

feature -- Command

        kill_by (a_cause: STRING_8)
        require
                is_valid_death_cause (a_cause)
        ensure
                is_dead
                get_death_cause ~ a_cause

end -- class DEATHABLE
```

7.2.6 ENTITY class

```
note
	description: "[
	A class to represent an entity in a QUADRANT.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

deferred class interface
	ENTITY

feature -- Attributes

	character: CHARACTER_8
			-- result -> ie 'E'

	coordinate: COORDINATE
			-- coordinate in GRID

feature -- Queries

	is_equal (other: like Current): BOOLEAN
			-- Is other attached to an object considered equal to current object?

feature -- out

	out: STRING_8
			-- result -> ie "E"

end -- class ENTITY
```

7.2.7 FUELABLE class

```
note
	description: "[
	A class that encapsulates common queries, attributes,
	and commands for entities with fuel.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

deferred class interface
	FUELABLE

feature -- Attributes

	fuel: INTEGER_32

	max_fuel: INTEGER_32
			-- maximum value of fuel

feature -- Commands

	spend_fuel_unit
			-- decrement fuel by one
	require
			fuel > 0
	ensure
			fuel ~ (old fuel - 1)

	charge_fuel (s: STAR)
			--increment fuel by {STAR}.luminosity, up to max_fuel
	require
			s.luminosity >= 0
	ensure
			max_fuel_does_not_change: max_fuel ~ old max_fuel
			never_charge_above_max_fuel: (((old fuel + s.luminosity) >= max_fuel) implies (fuel ~ max_fuel))
					and (((old fuel + s.luminosity) < max_fuel) implies (fuel ~ (old fuel + s.luminosity)))

feature -- Queries

	is_out_of_fuel: BOOLEAN
	ensure
			Result = (fuel ~ 0)

invariant
	0 <= fuel and fuel <= max_fuel

end -- class FUELABLE
```

7.2.8 ID_ENTITY class

```
note
	description: "[
	A class to represent an ENTITY and its identification number.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

deferred class interface
	ID_ENTITY

feature -- Attribute

	id: INTEGER_32

feature -- Queries

	is_equal (other: like Current): BOOLEAN
			-- Is other attached to an object considered equal to current object?

feature -- out

	out_sqr_bracket: STRING_8
			-- result -> "[id:character]"

	out_description: STRING_8
			-- result -> "out_sqr_bracket->"

end -- class ID_ENTITY
```

7.2.9 JANITAUR class

```
note
        description: "A class to represent a janitaur entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        JANITAUR

create
        make

feature -- Attributes

        max_load: INTEGER_32
                -- maximum value of load

        load: INTEGER_32

feature -- Queries

        is_dead_by_out_of_fuel: BOOLEAN
                -- was killed by out of fuel?

        is_dead_by_asteroid: BOOLEAN
                -- was killed by JANITAUR?

feature -- Commands

        increment_load_by_one
                -- increment load by one.
        require
                load /~ max_load
        ensure
                load ~ (old load + 1)

        clear_load (w: WORMHOLE)
                -- initialize load to 0.
        require
                wormhole_in_sector: w.coordinate ~ coordinate
        ensure
                load = 0

        check_health (sector: SECTOR)
                -- if sector.has_star ~ true recharge the janitaur's fuel cells
                -- execute kill_by_blackhole if sector.has_blachole ~ true.
                -- execute kill_by_out_of_fuel if "is_out_of_fuel" ~ true.

        behave (sector: SECTOR)
                -- perform behavior algorithm that pertains to JANITAUR as seen on pg 36

        kill_by_out_of_fuel
        require
                fuel = 0
        ensure
```

```
			is_dead_by_out_of_fuel

		kill_by_asteroid (killer_id: INTEGER_32)
		ensure
			is_dead_by_asteroid

		reproduce (moveable_id: INTEGER_32): like Current
			-- create another ENTITY of type {like current} with the same coordinate
				-- as current.
feature -- out

		out_death_message: STRING_8
			-- result -> {Abstract State: Death Messages JANITAUR on pg 26-27}

		out_description: STRING_8
			-- result -> "[id, character]->fuel:cur_fuel/max_fuel actions_left_until_reproduction: c_value /
reproduction_interval,
		-- turns_left: N/A or turns_left"
invariant
		0 <= load and load <= max_load

end -- class JANITAUR
```

7.2.10 LIFE class

```
note
        description: "A class that encapsulates an DEATHABLE's life."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        LIFE

create {DEATHABLE}
        make

feature -- Attribute

        point: INTEGER_32
                -- "life-point" as an INTEGER

        max: INTEGER_32
                -- maximum value of point

        is_dead: BOOLEAN
        ensure
                Result = (point ~ 0)

feature -- Commands

        set_life (a_value: INTEGER_32)
                -- initialize point to a_value
        require
                valid_value (a_value)
                not is_dead
        ensure
                point = a_value

        add_life (a_value: INTEGER_32)
                -- increment point by a_value up to max
        require
                a_value >= 0
                not is_dead

        subtract_life (a_value: INTEGER_32)
                -- decrement point by a_value down to 0.
        require
                a_value >= 0
                not is_dead

feature -- Queries

        valid_value (a_value: INTEGER_32): BOOLEAN
        ensure
                Result = (a_value >= 0 and a_value <= max)

        out: STRING_8
                -- result -> "life:point/max"

invariant
        min_0: point >= 0 and max >= 0
        value_max: point <= max
        no_revive: (is_dead) = (point = 0)
```

```
end -- class LIFE
```

7.2.11 MALEVOLENT class

```
note
        description: "A class to represent a malevolent entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        MALEVOLENT

create
        make

feature -- Queries

        is_dead_by_out_of_fuel: BOOLEAN
                -- was killed by out of fuel?

        is_dead_by_asteroid: BOOLEAN
                -- was killed by ASTEROID?

        is_dead_by_benign: BOOLEAN
                -- was killed by out of BENIGN?

feature -- Commands

        check_health (sector: SECTOR)
                -- if sector.has_star ~ true recharge the malevolent's fuel cells
                -- execute kill_by_blackhole if sector.has_blachole ~ true.
                -- execute kill_by_out_of_fuel if "is_out_of_fuel" ~ true.

        behave (sector: SECTOR)
                -- perform behavior algorithm that pertains to MALEVOLENT as seen on pg 36

        kill_by_out_of_fuel
        require
                fuel = 0
        ensure
                is_dead_by_out_of_fuel

        kill_by_asteroid (killer_id: INTEGER_32)
        ensure
                is_dead_by_asteroid

        kill_by_benign (killer_id: INTEGER_32)
        ensure
                is_dead_by_benign

        reproduce (moveable_id: INTEGER_32): like Current
                -- create another ENTITY of type {like current} with the same coordinate as current.
feature -- out

        out_death_message: STRING_8
                -- result -> {Abstract State: Death Messages MALEVOLENT on pg 26-27}

        out_description: STRING_8
                -- result -> "[id, character]->fuel:cur_fuel/max_fuel, actions_left_until_reproduction: c_value /
reproduction_interval,
                -- turns_left: N/A or turns_left"
```

```
end -- class MALEVOLENT
```

7.2.12 MOVEABLE_ENTITY class

```
note
	description: "[
	A class to represent an ID_ENTITY that can change
	its coordinate and is capable of death.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

deferred class interface
	MOVEABLE_ENTITY

feature -- Commands

	kill_by_blackhole (killer_id: INTEGER_32)
	ensure
		is_dead_by_blackhole

	check_health (sector: SECTOR)
		-- execute kill_by_blackhole if sector.has_blachole ~ true.
	require
		sector.coordinate ~ coordinate
		is_alive
	ensure
		alive_or_dead_current_remains_in_sector: sector.coordinate ~ coordinate

	set_coordinate (a_coordinate: COORDINATE)
		-- initialize coordinate to a_coordinate
	ensure
		coordinate ~ a_coordinate

feature -- Queries

	is_dead_by_blackhole: BOOLEAN
		-- was killed by BLACKHOLE?

feature -- out

	out_death_description: STRING_8
		-- result -> "out_description,%N		out_death_message".
	require
		is_dead

	out_death_message: STRING_8
		-- result -> {Abstract State: Death Message from pg 26-27 relevant to this entity}
	require
		is_dead

end -- class MOVEABLE_ENTITY
```

7.2.13 NP_MOVEABLE_ENTITY class

```
note
        description: "[
        A class to represent a MOVEABLE_ENTITY whose actions
        occur in defined intervals and whose actions cannot
        be explicitly controlled via user commands.
        Note: NP stands for NON_PLAYABLE
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

deferred class interface
        NP_MOVEABLE_ENTITY

feature -- Attributes

        behavior_messages: ARRAY [STRING_8]
                -- messages produced after behave executes.

feature -- Attributes

        turns_left: INTEGER_32
                -- turns left to behave

        set_turns_left (value: INTEGER_32)
                -- initialize turns_left to value
        require
                valid_value: 0 <= value and value <= 2
        ensure
                        turns_left ~ value

feature -- Commands

        behave (sector: SECTOR)
        require
                sector.coordinate ~ coordinate
                turns_left ~ 0
        ensure
                is_alive
                sector.coordinate ~ coordinate

invariant
        0 <= turns_left and turns_left <= 2

end -- class NP_MOVEABLE_ENTITY
```

7.2.14 NULL_ENTITY class

```
note
	description: "[
		A class to represent the absence of an ENTITY.

		Secret:
		(see QUADRANT secret).
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

class interface
	NULL_ENTITY

create {QUADRANT}
	make

end -- class NULL_ENTITY
```

7.2.15 PLANET class

```
note
        description: "A class to represent a planet entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        PLANET

create
        make

feature -- Attributes

        visited: BOOLEAN
                        -- was visited by EXPLORER?

        attached_to_star: BOOLEAN
                        -- is attached to a STAR?

        support_life: BOOLEAN
                        -- supports life?

        is_landable: BOOLEAN
                        -- is landable?

feature -- Command

        set_attached_to_star (s: STAR)
                        -- attach to STAR.
                require
                        star_is_in_same_sector: s.coordinate ~ coordinate
                                turns_left ~ 0
                                is_alive
                ensure
                                attached_to_star = True
                                turns_left ~ 0
                                is_alive
                        if_attached_to_yellow_star_then_current_is_landable: (attached {YELLOW_DWARF} s) implies
is_landable
                        if_not_attached_to_yellow_star_then_is_landable_false: (not (attached {YELLOW_DWARF} s))
implies
                                                                                (not is_landable)

        set_support_life (b: BOOLEAN)
                        -- initialize support_life to b
                require
                                attached_to_star
                                turns_left ~ 0
                                is_alive
                ensure
                                support_life = b
                                is_alive
                                turns_left ~ 0

        set_visited
                        -- initialize visited to true
                require
                                attached_to_star
```

```
                                        is_alive
                                        is_landable
                        ensure
                                        visited
                                        is_alive
                                        attached_to_star
                                        not is_landable

        behave (sector: SECTOR)
                                -- perform behavior algorithm that pertains to PLANET as seen on pg 36

feature -- Out

        out_death_message: STRING_8
                                -- result -> {Abstract State: Death Messages PLANET on pg 26-27}

        out_description: STRING_8
                                --result -> "[id, character]->attached?:T or F, support_life?:T or F, visited:T or F,
                                -- turns_left: N/A or turns_left"

end -- class PLANET
```

7.2.16 REPRODUCEABLE_ENTITY class

```
note
        description: "[
                A class to represent an NP_MOVEABLE_ENTITY that can reproduce.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

deferred class interface
        REPRODUCEABLE_ENTITY

feature -- Attributes

        actions_left_until_reproduction: INTEGER_32

        reproduction_interval: INTEGER_32
                        -- maximum value of actions_left_until_reproduction

feature -- Queries

        ready_to_reproduce: BOOLEAN
                ensure
                                Result = (actions_left_until_reproduction ~ 0)

feature -- Commands

        decrement_actions_left_by_one
                        -- decrement actions_left_until_reproduction by 1
                require
                                actions_left_until_reproduction > 0
                ensure
                                actions_left_until_reproduction = (old actions_left_until_reproduction - 1)

feature -- Commands

        reset_actions_left_until_reproduction
                        -- initialize actions_left_until_reproduction to reproduction_interval
                ensure
                                reproduction_interval ~ actions_left_until_reproduction

        reproduce (moveable_id: INTEGER_32): like Current
                        -- create another ENTITY of type {like current} with the same coordinate as current.
                require
                                ready_to_reproduce
                ensure
                                is_alive
                        reproduction_interval_is_reset: actions_left_until_reproduction ~ reproduction_interval
                        clone_and_current_are_different_entities: (Result /~ Current)
                        clone_and_current_have_same_coordinates: Result.coordinate ~ coordinate

invariant
                0 <= actions_left_until_reproduction and actions_left_until_reproduction <= reproduction_interval

end -- class REPRODUCEABLE_ENTITY
```

7.2.17 STAR class

```
note
        description: "[
                A class to represent a STATIONARY_ENTITY and its
                luminosity value.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

deferred class interface
        STAR

feature -- Attribute

        luminosity: INTEGER_32
                        -- luminosity value

feature -- Out

        out_description: STRING_8
                        -- result -> "[id, character]->Luminosity: luminosity".

end -- class STAR
```

7.2.18 STATIONARY_ENTITY class

```
note
        description: "[
        A class to represent an ID_ENTITY that is not also
        a MOVEABLE_ENTITY.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

deferred class interface
        STATIONARY_ENTITY

end -- class STATIONARY_ENTITY
```

7.2.19 WORMHOLE class

```
note
        description: "A class to represent a wormhole entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        WORMHOLE

create
        make

end -- class WORMHOLE
```

7.2.20 YELLOW_DWARF class

```
note
        description: "A class to represent a yellow_dwarf entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        YELLOW_DWARF

create
        make

end -- class YELLOW_DWARF
```

7.2.21 EXPLORER class

```
note
        description: "A class to represent the explorer entity."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        EXPLORER

create
        make

feature -- Attributes

        landed: BOOLEAN
                        -- is landed on a PLANET?

        found_life: BOOLEAN
                        -- has found life on a PLANET?

feature -- Queries

        is_dead_by_out_of_fuel: BOOLEAN
                        -- was killed by out of fuel.

        is_dead_by_malevolent: BOOLEAN
                        -- was killed by MALEVOLENT?

        is_dead_by_asteroid: BOOLEAN
                        -- was killed by ASTEROID?

feature -- Commands

        kill_by_malevolent (killer_id: INTEGER_32)
                ensure
                        is_dead_by_malevolent

        kill_by_asteroid (killer_id: INTEGER_32)
                ensure
                        is_dead_by_asteroid

        kill_by_out_of_fuel
                require
                        is_out_of_fuel
                ensure
                        is_dead_by_out_of_fuel

        check_health (sector: SECTOR)
                        -- if sector.has_star ~ true recharge the explorer's fuel cells
                        -- if "is_out_of_fuel" execute "kill_by_out_of_fuel".

        land_on (a_p: PLANET)
                        -- land and visit a_p.
                require
                        a_p.is_landable and not a_p.visited
                ensure
                        a_p.visited and landed and (a_p.support_life implies found_life)
```

```
	liftoff
			-- liftoff the planet explorer is currently landed on.
		require
				landed
		ensure
				not landed

feature -- Out

	out_status (quadrant: INTEGER_32): STRING_8
			-- result -> {Abstract State: Command-Specific Messages STATUS on pg 26}

	out_death_message: STRING_8
			-- result -> {Abstract State: Death Messages EXPLORER on pg 26-27 }

	out_description: STRING_8
			-- result -> "[id, character]->fuel:cur_fuel/max_fuel, life:cur_life/max_life, landed?:boolean".
			--  ie. "[0,E]->fuel:2/3, life:3/3, landed?:F"

end -- class EXPLORER
```

7.3 model Cluster

7.3.1 CONTROLLER class

```
note
        description: "[
                A class that provides an interface for executing
                all nine user commands and updates the user
                output when commands are executed.

                Secret:
                Attribute “game_state” is of type STATE which
                means CONTROLLER is a client of STATE.
                Note: “game_state” is polymorphic.
                Post executing a command in CONTROLLER,
                “game_state” transitions (changes its reference)
                to a subclass of STATE that is appropriate
                for the game.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        CONTROLLER

create {CONTROLLER_ACCESS}
        make

feature -- Attributes

        game_state: STATE
                        -- "game_state" is polymorphic
                        -- provides the user's STATE (see description in STATE) in the program. eg PLAY_STATE,
                        -- MAIN_MENU_STATE...

feature -- State Commands

        move_to_next_state
                        -- transition "game_sate" to the STATE referenced by "game_state.next_state" such that
"game_state" =
                        -- "game_state.next_state".
                        -- Note reference equality.
                ensure
                                game_state = game_state.next_state

feature -- User Commands

        abort
                        --execute "abort" command in "game_state" followed by "move_to_next_state"

        land
                        --execute "land" command in "game_state" followed by "move_to_next_state"

        liftoff
                        --execute "liftoff" command in "game_state" followed by "move_to_next_state"

        move (d: INTEGER_32)
                        --execute "move" command in "game_state" followed by "move_to_next_state"

        pass
```

```
                    --execute "pass" command in "game_state" followed by "move_to_next_state"

        play
                    --execute "play" command in "game_state" followed by "move_to_next_state"

        status
                    --execute "status" command in "game_state" followed by "move_to_next_state"

        test (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32)
                    --execute "test" command in the current "game_state" followed by "move_to_next_state"

        wormhole
                    --execute "wormhole" command in "game_state" followed by "move_to_next_state"

        reset
                    -- Reset model state.

feature -- Queries

        out: STRING_8
                    -- output "game_state"
            ensure then
                        Result ~ game_state.out

end -- class CONTROLLER
```

7.4 states Cluster

7.4.1 ABSTRACT_STATE_NUMBERS class

```
note
	description: "[
		A class to manage the first and second number
		as seen in the user output “e.g state: 3.1”.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

class interface
	ABSTRACT_STATE_NUMBERS

create
	make

feature -- Attibute

	first_number: INTEGER_32
			-- first number as seen in ie. state: 3.0
			-- signifies the number of valid turn/play/test commands executed during the course of the program.
			-- incremented by one after a valid turn command is executed in a subclass of STATE. ie state: 3.0 ->
			-- state: 4.0

	second_number: INTEGER_32
			-- second number as seen in ie. state: 3.2
			-- signifies the number of valid non-turn commands executed and invalid commands attempted after
			-- the last
			-- successful turn/play/test command was executed. Incremented by one after valid non-turn
			-- commands are
			-- executed or invalid commands are attempted by a subclass of STATE
			-- ie initialized to zero after a valid turn command is exectured in a subclass of STATE. ie state: 3.4 ->
			-- state: 4.0

feature -- Queries

	out: STRING_8
			-- output first_number and second_number into a form like "state:12.3"

end -- class ABSTRACT_STATE_NUMBERS
```

7.4.2 LANDED_STATE class

```
note
        description: "[
                A class that defines valid, and invalid user commands
                for when the user is in a game and the explorer
                is landed.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        LANDED_STATE

create
        make

feature -- Controller command / queries

        abort
                        -- execute abort command in SYMODYSSEY, and append “Mission aborted. Try test(3,5,7,15,30)” to
                        -- "out"
                ensure then
                        enter_main_menu_state: (attached {MAIN_MENU_STATE} next_state)

        land
                        -- attempting to execute "land_explorer" command of SIMODYSSEY while in LANDED_STATE and
not
                        -- PLAY_STATE, implies that preconditions of SIMODYSSEY. land_explorer are not met,
                        -- therefore append "Negative on that request:you are currently landed at Sector:X:Y" to "out"
                ensure then
                        invalid_command_implies_remain_in_landed_state: (attached {LANDED_STATE} next_state)

        liftoff
                        -- execute abort command in SYMODYSSEY
                        -- if not {SIMOSYSSEY}.explorer_alive, append one of Abstract State:
                        -- Death Messages EXPLORER [3 to 4] to "out"
                ensure then
                        if_explorer_is_dead_enter_main_menu_state: (not game_model.explorer_is_alive)
                                                        implies (attached {MAIN_MENU_STATE} next_state)
                        if_explorer_is_alive_enter_play_state: (game_model.explorer_is_alive)
                                                        implies (attached {PLAY_STATE} next_state)

        move (d: COORDINATE)
                        -- attempting to execute "move_explorer" command of SIMODYSSEY while
                        -- in LANDED_STATE and not PLAY_STATE,
                        -- implies that preconditions of {SIMODYSSEY}.move_explorer are not met,
                        -- therefore append "Negative on that request:you are currently landed at Sector:X:Y" to "out"
                ensure then
                        invalid_command_implies_remain_in_landed_state: (attached {LANDED_STATE} next_state)

        pass
                        -- execute "pass" command in SIMODYSSEY
                ensure then
                        remain_in_landed_state: (attached {LANDED_STATE} next_state)

        play
                        -- attempting to execute "new_game" command of SIMODYSSEY while
                        -- in LANDED_STATE and not MAIN_MENU_STATE,
                        -- implies that preconditions of {SIMODYSSEY}.new_game are not met,
                        -- therefore append "To start a new mission, please abort the current one first." to "out"
                ensure then
                        invalid_command_implies_remain_in_landed_state: (attached {LANDED_STATE} next_state)

        status
                        -- append "Negative on that request:already landed on a planet at Sector:X:Y" to "out"
                ensure then
                        remain_in_landed_state: (attached {LANDED_STATE} next_state)

        test (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32)
                        -- attempting to execute "new_game" command of SIMODYSSEY while
                        -- in LANDED_STATE and not MAIN_MENU_STATE,
                        -- implies that preconditions of {SIMODYSSEY}.new_game are not met,
                        -- therefore append "To start a new mission, please abort the current one first." to "out"
                ensure then
                        invalid_command_implies_remain_in_landed_state: (attached {LANDED_STATE} next_state)

        wormhole
                        -- attempting to execute "wormhole_explorer" command of SIMODYSSEY while
                        -- in LANDED_STATE and not PLAY_STATE,
                        -- implies that preconditions of {SIMODYSSEY}.wormhole_explorer are not met,
                        -- therefore append "Negative on that request:you are currently landed at Sector:X:Y" to "out"
                ensure then
                        invalid_command_implies_remain_in_landed_state: (attached {LANDED_STATE} next_state)

end -- class LANDED_STATE
```

7.4.3 MAIN_MENU_STATE class

```
note
	description: "[
		A class that defines valid, and invalid user commands
		for when the user is not in a game.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"


class interface
	MAIN_MENU_STATE


create
	make


feature -- Controller command / queries


	abort
				-- attempt to execute "abort, land, liftoff, pass, status, wormhole"
				-- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
				-- implies that preconditions of such commands in SIMODYSSEY are not met,
				-- therefore append "Negative on that request:no mission in progress." to "out"
				-- Was declared in MAIN_MENU_STATE as synonym of land, liftoff, pass, status and wormhole.
		ensure then
				invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

	land
				-- attempt to execute "abort, land, liftoff, pass, status, wormhole"
				 -- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
				-- implies that preconditions of such commands in SIMODYSSEY are not met,
				-- therefore append "Negative on that request:no mission in progress." to "out"
				-- Was declared in MAIN_MENU_STATE as synonym of abort, liftoff, pass, status and wormhole.
		ensure then
				invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

	liftoff
				-- attempt to execute "abort, land, liftoff, pass, status, wormhole"
				-- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
				-- implies that preconditions of such commands in SIMODYSSEY are not met,
				-- therefore append "Negative on that request:no mission in progress." to "out"
				-- Was declared in MAIN_MENU_STATE as synonym of abort, land, pass, status and wormhole.
		ensure then
				invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

	pass
				-- attempt to execute "abort, land, liftoff, pass, status, wormhole"
				-- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
				-- implies that preconditions of such commands in SIMODYSSEY are not met,
				-- therefore append "Negative on that request:no mission in progress." to "out"
				-- Was declared in MAIN_MENU_STATE as synonym of abort, land, liftoff, status and wormhole.
		ensure then
				invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

	status
				-- attempt to execute "abort, land, liftoff, pass, status, wormhole"
				-- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
                    -- implies that preconditions of such commands in SIMODYSSEY are not met,
                    -- therefore append "Negative on that request:no mission in progress." to "out"
                    -- Was declared in MAIN_MENU_STATE as synonym of abort, land, liftoff, pass and wormhole.
            ensure then
                    invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

        wormhole
                    -- attempt to execute "abort, land, liftoff, pass, status, wormhole"
                    -- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
                    -- implies that preconditions of such commands in SIMODYSSEY are not met,
                    -- therefore append "Negative on that request:no mission in progress." to "out"
                    -- Was declared in MAIN_MENU_STATE as synonym of abort, land, liftoff, pass and status.
            ensure then
                    invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

        move (d: COORDINATE)
                    -- attempting to execute "move_explorer"
                    -- commands of SIMODYSSEY while in MAIN_MENU_STATE and not PLAY_STATE,
                    -- implies that preconditions of {SIMODYSSEY}.move_explorer are not met,
                    -- therefore append "Negative on that request:no mission in progress." to "out"
            ensure then
                    invalid_command_implies_remain_in_main_menu_state: (attached {MAIN_MENU_STATE}
next_state)

        play
                    -- execute "new_game" command in SIMODYSSEY
            ensure then
                    enter_play_state: (attached {PLAY_STATE} next_state)


        test (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32)
                    -- if precondition for command "new_game" in SIMODYSSEY is not met,
                    -- append "Thresholds should be non-decreasing order." to "out"
                    -- if preconditions are met, execute "new_game" command in SIMODYSSEY
            ensure then
                    valid_thresholds_implies_enter_play_state: game_model.valid_thresholds (a_threshold, j_threshold,
                        m_threshold, b_threshold, p_threshold) implies (attached {PLAY_STATE} next_state)
                    invalid_thresholds_implies_remain_in_main_menu_state: (not game_model.valid_thresholds
(a_threshold,
                        j_threshold, m_threshold, b_threshold, p_threshold))
                                        implies (attached {MAIN_MENU_STATE} next_state)

end -- class MAIN_MENU_STATE
```

7.4.4 PLAY_STATE class

```
note
        description: "[
                A class that defines valid, and invalid user commands
                for when the user is in a game, and the explorer is not landed.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        PLAY_STATE

create
        make

feature -- Commands

        abort
                        -- execute abort command in SYMODYSSEY, and append “Mission aborted. Try test(3,5,7,15,30)” to
                        -- "out"
                ensure then
                        enter_main_menu_state: attached {MAIN_MENU_STATE} next_state

        land
                        -- if precondition for command "land_explorer" in SIMODYSSEY is not met,
                        -- append one of Abstract State: Error Messages LAND [3 to 5] to "out"
                        -- if land precondition in SIMODYSSEY is met, execute land in SIMODYSSEY.
                        -- after succesfully executing "land_explorer", if "{SIMODYSSEY}.explorer_found_life",
                        -- append “Tranquility base here - we've got a life!” to "out"
                        -- after succesfully executing "land_explorer", if "not {SIMODYSSEY}.explorer_found_life",
                        -- append “Explorer found no life as we know it at Sector:X:Y” to "out"
                ensure then
                        if_explorer_is_not_landed_remain_in_play_state:
                                ((not game_model.explorer_is_landed)
                                implies (attached {PLAY_STATE} next_state))
                        if_explorer_is_landed_and_explorer_did_not_find_life_enter_landed_state:
                                (((game_model.explorer_is_landed) and (not game_model.explorer_found_life))
                                implies (attached {LANDED_STATE} next_state))
                        if_explorer_is_landed_and_explorer_did_found_life_enter_main_menu_state:
                                (((game_model.explorer_is_landed) and (game_model.explorer_found_life))
                                implies (attached {MAIN_MENU_STATE} next_state))

        liftoff
                        -- attempting to execute "liftoff_explorer" command of SIMODYSSEY
                        -- while in PLAY_STATE and not LANDED_STATE,
                        -- implies that preconditions of {SIMODYSSEY}.liftoff are not met,
                        -- therefore append "Negative on that request:you are not on a planet at Sector:X:Y" to "out"
                ensure then
                        invalid_command_implies_remain_in_play_state: (attached {PLAY_STATE} next_state)

        move (d: COORDINATE)
                        -- if precondition for command "move_explorer" in SIMODYSSEY is not met,
                        -- append "Cannot transfer to new location as it is full." to "out"
                        -- if preconditions are met, execute "move_explorer" command in SIMODYSSEY.
                        -- if explorer dies after successfully moving, append one of Abstract State:
                        -- Death Messages EXPLORER [1 to 4] to "out"
                ensure then
                        if_explorer_is_alive_after_succesfully_moving_remain_in_play_state:
                                 (game_model.explorer_is_alive) implies (attached {PLAY_STATE} next_state)
                        if_explorer_is_dead_after_succesfully_moving_enter_in_main_menu_state:
```

```
                                        (not game_model.explorer_is_alive) implies (attached {MAIN_MENU_STATE} next_state)

        pass
                                -- execute "pass" command in SIMODYSSEY
                                -- if explorer dies after succesffuly passing, append one of Abstract State:
                                -- Death Messages EXPLORER [3 to 4] to "out"
                ensure then
                                if_not_dead_remain_in_play_state:
                                        (game_model.explorer_is_alive) implies (attached {PLAY_STATE} next_state)
                                if_dead_enter_main_menu_state:
                                         (not game_model.explorer_is_alive) implies (attached {MAIN_MENU_STATE}
                                next_state)


        play
                                -- attempting to execute "new_game" command of SIMODYSSEY
                                -- while in PLAY_STATE and not MAIN_MENU_STATE,
                                -- implies that preconditions of {SIMODYSSEY}.new_game are not met,
                                -- therefore append "To start a new mission, please abort the current one first." to "out"
                ensure then
                                invalid_command_implies_remain_in_play_state: (attached {PLAY_STATE} next_state)


        status
                                -- append “Explorer status report:Travelling at cruise speed at [X,Y,Z]
                                -- Life units left:V, Fuel units left:W” to "out"
                ensure then
                                status_should_not_change_the_state: (attached {PLAY_STATE} next_state)


        test (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32)
                                -- attempting to execute "new_game" command of SIMODYSSEY
                                -- while in PLAY_STATE and not MAIN_MENU_STATE,
                                -- implies that preconditions of {SIMODYSSEY}.new_game are not met,
                                -- therefore append "To start a new mission, please abort the current one first." to "out"
                ensure then
                                invalid_command_implies_remain_in_play_state: (attached {PLAY_STATE} next_state)


        wormhole
                                -- if precondition for command "wormhole_explorer" in SIMODYSSEY is not met,
                                -- append "Explorer couldn't find wormhole at Sector:X:Y" to "out"
                                -- if preconditions are met, execute "wormhole_explorer" command in SIMODYSSEY.
                                -- if explorer dies after succesffuly wormhole-ing, append one of Abstract State: Death Messages
                                -- EXPLORER [1 to 4] to "out"
                ensure then
                                explorer_is_alive_after_successful_wormhole_implies_remain_in_play_state:
                                        (game_model.explorer_is_alive) implies (attached {PLAY_STATE} next_state)
                                explorer_is_dead_after_successful_wormhole_implies_enter_main_menu_state:
                                        ((not game_model.explorer_is_alive)) implies (attached {MAIN_MENU_STATE}
                                next_state)

end -- class PLAY_STATE
```

7.4.5 STATE class

```
note
	description: "[
				A class that defines the set of valid, invalid user
				commands, and generates the user's output when
				commands are executed.

				Secret:
				private attribute “abstract_state_numbers” is of type
				ABSTRACT_STATE_NUMBERS which means STATE is a client
				of ABSTRACT_STATE_NUMBERS.
				“abstract_state_numbers” is updated accordingly after
				the execution of a command.
	]"
	author: "Jinho Hwang, Ato Koomson"
	date: "April 13, 2020"
	revision: "1"

deferred class interface
	STATE

feature -- Attributes

	game_model: SIMODYSSEY
				-- the game whose output is being defined in STATE

	next_state: STATE
				-- after creation next_state = current. Note refference equality
				-- hence forth, the refference of next_state is modified by executuring commands (eg. abort, land...) in
				-- current.
				-- AFTER executing a command (eg. abort, land...) in current,
				-- next_state stores a refference to the "resultant STATE" of the system.
				-- This "resultant STATE" can be "transitioned to" by any client of STATE.
				-- "See {CONTROLLER}.move_to_next_state for an example"
				-- next_state is polymorphic and can occupy an instance of PLAY_STATE, MAIN_MENU_STATE and
				-- LANDED_STATE

feature -- Commands

	abort

	land

	liftoff

	move (d: COORDINATE)

	pass

	play

	status

	test (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32)

	wormhole

feature -- Out
```

```
	out: STRING_8
			-- output generated after executing a commnad.

invariant
	if_next_state_is_main_menu_state_then_game_is_not_in_session: attached {MAIN_MENU_STATE} next_state
								implies not game_model.game_is_in_session
	if_next_state_is_play_state_then_game_is_in_session: attached {PLAY_STATE} next_state
							implies (game_model.game_is_in_session and not
							game_model.explorer_is_landed)
	if_next_state_is_landed_state_then_game_is_in_session: attached {LANDED_STATE} next_state
							implies (game_model.game_is_in_session and
						game_model.explorer_is_landed)

end -- class STATE
```

7.4.6 SIMODYSSEY class

```
note
        description: "[
                A class that controls the addition, removal,and movement
                of entities in the galaxy and provides an interface for
                controlling the explorer’s actions in the galaxy.

                Secret:
                Post execution of a command, non-user-controlled entities
                move, reproduce, behave, and check according to the
                “turn" command algorithm on page 33 of the project
                specifications.
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        SIMODYSSEY

create
        make

feature -- Attributes

        galaxy: GRID
                        -- a GRID containing all live ENTITYs in the game.

feature -- State of Game Queries

        is_test_game: BOOLEAN
                        -- result is true if query "game_is_in_session" is true AND command "new_game"
                        -- was previously called with argument TRUE in-place for parameter "is_test". result is false
                otherwise.

        is_aborted: BOOLEAN
                        -- result is true if query "game_is_in_session"is true AND THEN command "abort" is called.
                        -- false otherwise.

        game_is_in_session: BOOLEAN
                        -- result equals true means that a game_is_in_session.
                ensure
                        valid_game_session:
                         Result = (explorer_is_alive
                        and (not explorer_found_life)
                        and (not is_aborted)
                        and galaxy.at (explorer_coordinate).has_id (explorer_id))

        valid_thresholds (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32): BOOLEAN
                        -- result equals true if threshold values from left to right are passed (as arguments) in increasing order

feature -- Explorer Interface Commands

        abort_game
                        -- abort the game
                require
                                game_is_in_session
                ensure
                                is_aborted
                                not game_is_in_session
```

```
new_game (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold: INTEGER_32; is_test: BOOLEAN)
                    -- start a "new_game"
        require
                            valid_thresholds (a_threshold, j_threshold, m_threshold, b_threshold, p_threshold)
                            not game_is_in_session
        ensure
                            game_is_in_session and (is_test_game = is_test)

move_explorer (d: COORDINATE)
                    -- move the explorer away from its current sector and towards a sector in direction "d".
        require
                            game_is_in_session
                            d.is_direction
                            not sector_in_explorer_direction_is_full (d)
                            not explorer_is_landed
        ensure
                    if_not_lost_the_explorer_is_in_new_sector:
                            (explorer_is_alive)
                             implies galaxy.at ((old explorer_coordinate + d).wrap_coordinate_to_coordinate ((old
                            explorer_coordinate + d), create {COORDINATE}.make ([1, 1]), create
                            {COORDINATE}.make ([number_rows, number_columns]))).has_id (explorer_id)

                    if_explorer_is_not_at_new_sector_then_explorer_is_dead:
                            (not galaxy.at ((old explorer_coordinate + d).wrap_coordinate_to_coordinate ((old
                            explorer_coordinate + d), create {COORDINATE}.make ([1, 1]), create
                            {COORDINATE}.make ([number_rows, number_columns]))).has_id (explorer_id)) implies
                            (not explorer_is_alive)

wormhole_explorer
                    -- wormhole the explorer into a sector.
        require
                            game_is_in_session
                            not explorer_is_landed
                            explorer_sector_has_wormhole
        ensure
                    if_not_lost_the_explorer_is_in_new_position:
                             explorer_is_alive implies galaxy.at (explorer_coordinate).has_id (explorer_id)
                    if_explorer_is_not_in_the_galaxy_is_dead:
                            (not galaxy.at (explorer_coordinate).has_id (explorer_id))
                            implies ((not explorer_is_alive) and (not game_is_in_session))

land_explorer
                    -- land the explorer on a landable planet
        require
                            game_is_in_session
                            not explorer_is_landed
                    explorers_sector_is_landable:
                            explorer_sector_has_planets
                            and explorer_sector_has_yellow_dwarf
                            and explorer_sector_has_unvisted_attached_planets
        ensure
                            explorer_is_alive
                            explorer_is_landed
                    if_found_life_then_game_is_over: (explorer_found_life implies (not game_is_in_session))

liftoff_explorer
                    -- liftoff explorer.
        require
                            game_is_in_session
                            explorer_is_landed
        ensure
                            not explorer_is_landed
                    if_dead_then_game_is_over: ((not explorer_is_alive) implies (not game_is_in_session))
        pass_explorer_turn
                        -- pass the explorer's turn in the game.
                require
                                game_is_in_session
                ensure
                        if_dead_game_is_over: (not explorer_is_alive) implies (not game_is_in_session)

        status_of_explorer
                require
                                game_is_in_session
                ensure
                                game_is_in_session

feature -- Explorer Interface Boolean Queries

        sector_in_explorer_direction_is_full (d: COORDINATE): BOOLEAN
                        -- result equals true if a sector in "galaxy" in direction d is full. result equals false otherwise.
                require
                                d.is_direction
                                game_is_in_session

        explorer_is_landed: BOOLEAN
                        -- result equals true if explorer is landed on a planet in "galaxy". false otherwise.

        explorer_sector_has_wormhole: BOOLEAN
                        -- result equals true if explorer is contained in a SECTOR that also contains a wormhole. false
otherwise.
                require
                                game_is_in_session

        explorer_found_life: BOOLEAN
                        -- result equals true if the explorer found life while landed on a planet. false otherwise
                ensure
                                (Result = True) implies (explorer_is_landed and explorer_is_alive)

        planet_in_explorer_sector_supports_life: BOOLEAN
                        -- result equals true if there exists a planet in the explorer's sector that supports life. false otherwise.
                require
                                game_is_in_session

        explorer_is_alive: BOOLEAN
                        -- result equals true if explorer is alive. false otherwise.

        explorer_sector_is_landable: BOOLEAN
                        -- result equals true if there exists (attached and unvisited) planet(s) in the explorer's sector
                        --  and the explorer's sector contains a YELLOW_DWARF . false otherwise.
                require
                                game_is_in_session
                ensure
                        valid_properties_for_life: Result = (explorer_sector_has_planets and
explorer_sector_has_yellow_dwarf
                                                                        and
                                                                explorer_sector_has_unvisted_attached_planets)

        explorer_dead_by_out_of_fuel: BOOLEAN
                        -- result equals true if the explorer died by out of fuel. false otherwise.

        explorer_dead_by_blackhole: BOOLEAN
                        -- result equals true if the explorer died by blackhole. false otherwise.
```

```
		explorer_dead_by_asteroid: BOOLEAN
				-- result equals true if the explorer died by asteroid. false otherwise.


		explorer_dead_by_malevolent: BOOLEAN
				-- result equals true if the explorer died by malevolent. false otherwise.


		explorer_sector_has_yellow_dwarf: BOOLEAN
				-- result equals true if the explorer's SECTOR contains a YELLOW_DWARF. false otherwise.
			require
					game_is_in_session


		explorer_sector_has_planets: BOOLEAN
				-- result equals true if the explorer's SECTOR contains PLANETs. false otherwise.
			require
					game_is_in_session


		explorer_sector_has_unvisted_attached_planets: BOOLEAN
				-- result equals true if the explorer's SECTOR contains attached, yet unvisited PLANET's. false
otherwise.
			require
					game_is_in_session

feature -- Explorer Interface non-Boolean Queries


		explorer_coordinate: COORDINATE
				-- result equals the explorer's coordinate in "galaxy"


		explorer_id: INTEGER_32
				-- result equals explorer's id

feature -- Out


		out: STRING_8
				-- result equals output messages (Movement: ... Sectors: ... Description: ... Deaths This Turn: ... and
				-- galaxy.out) when "is_test_game" is true


		out_status_explorer: STRING_8
				-- explorer status message
			require
					explorer_is_alive


		explorer_death_message: STRING_8
				-- output message generated when the explorer dies
			require
					not explorer_is_alive

end -- class SIMODYSSEY
```

7.5 utility Cluster

7.5.1 COORDINATE class

```
note
        description: "A class to represent comparable coordinates."
        author: "CD, Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        COORDINATE

create
        make

convert
        make ({TUPLE [INTEGER_32, INTEGER_32]})

feature -- Attributes

        row: INTEGER_32
                        -- the row of a COORDINATE. ie [row, ]

        col: INTEGER_32
                        -- the column of a COORDINATE. ie [ ,column]

feature -- Queries

        is_less alias "<" (other: like Current): BOOLEAN
                        -- Is current object less than other?

        is_equal (other: like Current): BOOLEAN
                        -- Is other attached to an object of the same type
                        -- as current object and identical to it?

        add alias "+" (other: like Current): COORDINATE
                        -- result -> [row + other.row, col + other.col]

        subtract alias "-" (other: like Current): COORDINATE
                        -- result -> [row - other.row, col - other.col]

        is_direction: BOOLEAN
                        -- is current object equal to an attribute in DIRECTION_UTILITY?

        wrap_coordinate_to_coordinate (c, lower_bound, upper_bound: COORDINATE): COORDINATE
                        --result equals a COORDINATE that lies between lower_bound and upper_bound

feature -- out

        out: STRING_8
                        -- result -> "(row:col)"

        out_sqr_bracket: STRING_8
                        -- result -> "[row:col]"
```

```
		out_colon: STRING_8
						-- result -> "row:col"

		out_sqr_bracket_comma: STRING_8
						-- result -> "[row,col]"

end -- class COORDINATE
```

7.5.2 DIRECTION_UTILITY class

```
note
        description: "[
                                A class that contains common direction COORDINATEs
                                (e.g. N -> [-1,0], E -> [0,1] …)
        ]"
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

expanded class interface
        DIRECTION_UTILITY

create
        default_create

feature -- Attributes

        n: COORDINATE
                        -- result -> [-1,0]

        e: COORDINATE
                        -- result -> [0,1]

        s: COORDINATE
                        -- result -> [1,0]

        w: COORDINATE
                        -- result -> [0,-1]

        ne: COORDINATE
                        -- result -> [-1,1]

        se: COORDINATE
                        -- result -> [1,1]

        sw: COORDINATE
                        -- result -> [1,-1]

        nw: COORDINATE
                        -- result -> [-1,-1]

feature -- Queries

        number_for_direction (d: INTEGER_32): COORDINATE
                        -- 1 implies result -> N, 2 implies result -> NE, 3 implies result -> E,
                        -- 4 implies result -> SE, ... 8 implies result -> NW
                require
                        d_is_in_range: d <= 8 and d >= 1
                ensure
                                Result.is_direction

end -- class DIRECTION_UTILITY
```

7.5.3 ID_DISPATCHER class

```
note
        description: "A class for generating unique entity ids."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

class interface
        ID_DISPATCHER

create
        make

feature -- Commands

        reset
                        -- initialize current_id to initial_id.
                ensure
                                current_id ~ initial_id

        update_id
                        -- if id_up is true, increment current_id by 1.
                        -- decrement current_id by 1 otherwise.
                ensure
                        case_where_get_id_is_incremented: id_up implies (current_id ~ (old current_id + 1))
                        case_where_get_id_is_decremented: (not id_up) implies (current_id ~ (old current_id - 1))

feature -- Attributes

        current_id: INTEGER_32
                        -- the current id

        id_up: BOOLEAN
                        --see update_id.

        initial_id: INTEGER_32
                        -- first id returned by current_id.

end -- class ID_DISPATCHER
```

7.5.4 MESSAGE class

```
note
        description: "A class for generating Abstract State messages."
        author: "Jinho Hwang, Ato Koomson"
        date: "April 13, 2020"
        revision: "1"

expanded class interface
        MESSAGE

create
        default_create

feature -- Format related messages

        Empty_string: STRING_8 = ""

        Left_margin: STRING_8 = " "

        Left_big_margin: STRING_8 = " "

feature -- First line: Validity

        Ok: STRING_8 = "ok"

        Error: STRING_8 = "error"

feature -- First line: Mode

        Play: STRING_8 = "play"

        Test: STRING_8 = "test"

feature -- Abstract State: Command-Specific Messages INITIAL MESSAGE

        initial_message: STRING_8
                        -- result -> " Welcome! Try test(3,5,7,15,30)"

feature -- Abstract State: Command-Specific Messages STATUS

        status_not_landed (row, col, quad, life, fuel: INTEGER_32): STRING_8

        status_landed (row, col, quad, life, fuel: INTEGER_32): STRING_8

feature -- Abstract State: Error Messages STATUS

        status_error_no_mission: STRING_8

feature -- Abstract State: Command-Specific Messages LAND

        land_life_found: STRING_8

        land_life_not_found (row, col: INTEGER_32): STRING_8

feature -- Abstract State: Error Messages LAND

        land_error_no_mission: STRING_8
```

```
		land_error_landed_already (row, col: INTEGER_32): STRING_8

		land_error_no_yellow_dwarf (row, col: INTEGER_32): STRING_8

		land_error_no_planets (row, col: INTEGER_32): STRING_8

		land_error_no_visited_planets (row, col: INTEGER_32): STRING_8

feature -- Abstract State: Command-Specific Messages LIFTOFF

		liftoff (row, col: INTEGER_32): STRING_8

feature -- Abstract State: Error Messages LIFTOFF

		liftoff_error_no_mission: STRING_8

		liftoff_error_not_on_planet (row, col: INTEGER_32): STRING_8

feature -- Abstract State: Command-Specific Messages ABORT

		abort: STRING_8

feature -- Abstract State: Error Messages ABORT

		abort_error_no_mission: STRING_8

feature -- Abstract State: Command-Specific Messages GAME IS OVER

		game_is_over: STRING_8

feature -- Abstract State: Death Messages (Death due to blackhole.)

		moveable_entity_death_by_blackhole (np: MOVEABLE_ENTITY; sector_row, sector_col, blackhole_id: INTEGER_32):
STRING_8
					-- result -> " MOVEABLE_ENTITY got devoured by blackhole (id: -1) at Sector:row:col"
			require
						np.is_dead_by_blackhole
						blackhole_id ~ -1
					valid_sector_of_death: sector_row ~ 3 and sector_col ~ 3

feature -- Abstract State: Death Messages (Death due to janitaur.)

		asteroid_death_by_janitaur (a: ASTEROID; sector_row, sector_col, janitaur_id: INTEGER_32): STRING_8
					-- result -> " Asteroid got imploded by janitaur (id: id) at Sector:row:col"
			require
						a.is_dead_by_janitaur
					valid_sector_of_death: a.coordinate.row ~ sector_row and a.coordinate.col ~ sector_col

feature -- Abstract State: Death Messages (Death due to asteroid.)

		moveable_entity_death_by_asteroid (me: MOVEABLE_ENTITY; sector_row, sector_col, asteroid_id: INTEGER_32):
STRING_8
					-- result -> " MOVEABLE_ENTITY got destroyed by asteroid (id: id) at Sector:row:col"
			require
					me_is_not_a_planet: not attached {PLANET} me
					me_is_not_an_asteroid: not attached {ASTEROID} me
						me.is_dead
					valid_sector_of_death: me.coordinate.row ~ sector_row and me.coordinate.col ~ sector_col
```

```
feature -- Abstract State: Death Messages (Death due to benign.)

        malevolent_death_by_benign (m: MALEVOLENT; sector_row, sector_col, benign_id: INTEGER_32): STRING_8
                        -- result -> " Malevolent got destroyed by benign (id: id) at Sector:row:col"
                require
                        m.is_dead_by_benign
                        valid_sector_of_death: m.coordinate.row ~ sector_row and m.coordinate.col ~ sector_col

feature -- Abstract State: Death Messages (Out of fuel.)

        fuelable_moveable_entity_death_by_out_of_fuel (f: MOVEABLE_ENTITY; sector_row, sector_col: INTEGER_32):
STRING_8
                        -- result -> " MOVEABLE_ENTITY got lost in space - out of fuel at Sector:row:col"
                require
                        f_is_fuelable: attached {FUELABLE} f
                        f_is_out_of_fuel: (attached {FUELABLE} f as f_e) implies f_e.is_out_of_fuel
                        f.is_dead
                        valid_sector_of_death: f.coordinate.row ~ sector_row and f.coordinate.col ~ sector_col

feature -- Abstract State: Death Messages (Death due to malevolent.)

        explorer_death_by_malevolent (e: EXPLORER; sector_row, sector_col: INTEGER_32): STRING_8
                        -- result -> " Explorer got lost in space - out of life support at Sector:row:col"
                require
                        e.is_dead_by_malevolent
                        valid_sector_of_death: e.coordinate.row ~ sector_row and e.coordinate.col ~ sector_col

feature -- Abstract State: Error Messages MOVE

        move_error_no_mission: STRING_8

        move_error_landed (row, col: INTEGER_32): STRING_8

        move_error_sector_full: STRING_8

feature -- Abstract State: Error Messages PASS

        pass_error_no_mission: STRING_8

feature -- Abstract State: Error Messages PLAY

        play_error_no_mission: STRING_8

feature -- Abstract State: Error Messages TEST

        test_error_no_mission: STRING_8

        test_error_threshold: STRING_8

feature -- Abstract State: Error Messages WORMHOLE

        wormhole_error_no_mission: STRING_8

        wormhole_error_landed (row, col: INTEGER_32): STRING_8

        wormhole_error_explorer_not_found_wormhole (row, col: INTEGER_32): STRING_8

end -- class MESSAGE
```