NR#2 and Project, Requirements, Functionality

Team Name: 4-Six Stars

Supervisor:
Dr. Balla Katalin

Members:

[Ashiq Muhammad]	**[EDOXAM]**	**[muhammadashiqse10@gmail.com]**
[Ibrar Shakoor]	**[FS4J6L]**	**[ibrarraja11222@gmai.com]**
[Wail Ullah]	**[WTW2SV]**	**[Wali6985@gmail.com]**
[Asad Idrees Razak]	**[XJJUK4]**	**[Idreesshaikh6300@gmail.com]**
[Igbineweka Vincent]	**[TOM9S6]**	**[ifyvinz@gmail.com]**
[Orgil Luis Macha]	**[VH0JMU]**	**[orgilluismacha@gmail.com]**

2-Requirements, project, functionality

2.1 Introduction

Asteroid Mining

Humanity has decided to exploit the resources contained in asteroids. In order to do this, the settlers have to build a space station in the asteroid belt. The resources to build a space station would be too expensive to transport there, so the settlers have to mine them from the asteroids. Players of the game control the settlers. Settlers wander in the asteroid belt with single person spaceships looking for resources.

Asteroids are covered in rocks. The depth of this rock mantle can vary from asteroid to asteroid. The important resources (water ice, iron, carbon, uranium, etc.) can be found in the core of the asteroids. Some resources (like uranium) are highly radioactive. There can be hollow asteroids, whose core is empty. The core of an asteroid is always homogeneous; it is always made of a single kind of a material.

A settler can do one operation in a single move. There are several operations, like travelling, drilling, mining, building robots, building teleportation-gates, etc. When a settler travels, they go to a neighboring asteroid (an asteroid may have multiple hundreds of neighbors). When a settler drills, they deepen the hole in the mantle with one unit. When a settler mines, they extract the resource from the core of the asteroid. This is only possible if the mantle has been completely drilled through. A single settler can only carry 10 units of resources, that's the spaceships' capacity. A hollow asteroid, however, can be filled with a unit of resource, counting as an operation.

There are some dangers lurking in the asteroid belt. When a fully drilled asteroid with a radioactive core is at perihelion, the asteroid explodes and kills any settler on it. Radioactive material, therefore, can only be mined when the asteroid is at aphelion. Sometimes a sun storm reaches the asteroid belt, and it is also dangerous to the settlers. A settler can only survive a sun storm if they hide in the core of a hollow asteroid. This, of course, is only possible if the mantle has been drilled through.

Using up a unit of iron, a unit of carbon and a unit of uranium, the settlers can build autonomous robots controlled by artificial intelligence. These robots can only travel between asteroids and drill holes. Robots cannot mine because they are unable to transport things. Robots, however, can survive radioactive explosions, and in this case they land on a neighboring asteroid. Sun storms do damage robots unless they hide in a hollow asteroid.

Using up two units of iron, a single unit of water ice and a single unit of uranium, the settlers can build a pair of teleportation-gates. The gates can be later deployed in the vicinity of the asteroid the settler is on. The two gates of a pair remain in contact for good, and entering either the traveller (settler, robot, etc) will be immediately transported to the other. The settler can bring the freshly built gates with themselves, but at the same time a single settler can only bring two gates.

The game can end in two ways. If all the settlers die, the players lose. If, however, they can mine at least three units of each resource and they collect those materials on a single asteroid, they can build the space station and the players win the game.

2.1.1 Goal

The goal of the document is to define the requirements, project characteristics, and required functionalities of the Asteroid Mining application.

2.1.2 Application Domain

This is an intuitive game where the Players play the role of the Settlers whose task is to successfully build a space station by travelling to, drilling in, and mining resources from the current/neighbouring asteroids. Throughout the game the Players have to collect the resources (water ice, iron, carbon, uranium) and add them to inventory. There are two possible ways for the game to end: either the settlers collect enough resources to build the space station or die from a sun storm or asteroid explosion. Apart from the settlers there are autonomous robots which are made by the settlers to assist in everything except mining, and there are teleportation gates for the travel.

2.1.3 References

Software Used:

- StarUML
- MS Word
- Google Docs

Problem Definition: https://www.iit.bme.hu/targyak/BMEVIIIAB02/problem-definition

2.1.4 Document content

- **Overview**: Describing the whole problem statement in a much more detailed manner with the high level architecture description of the system, subsystems and interfaces.
- **Requirements**: The functional and non-functional requirements as well as the software and hardware resources to design, implement and test the system.
- **Essential Use-Cases**: It is the interaction of actors and the system, and their corresponding behavioral responses as well as the behavior inside them.
- **Glossary**: Terminology definition and their explanation. These are the essential substance of the document in alphabetical order for better accessibility.
- **Project Plan**: It is a workflow routine that we will follow as a Team in order to deliver a Multiple Viable Product (MVP) by the end of each week to meet the requirements.
- **Protocol**: It is the general division of time spent on tasks and exactly what is achieved by whom.

2.2 Overview

2.2.1 General overview

The essential components of the game are settlers, robots, and asteroids. Settlers are controlled by the player to collect resources to build robots and space stations. These resources are present in the core of the asteroids, and the settlers/robots need to drill to the core to access these resources.

Settlers and Robots can also travel between different asteroids in their vicinity by either normal transportation, or by using teleportation gates. Settlers need to collect a fixed amount of resources to win the game, while also avoiding sun storms and radioactive asteroids.

2.2.2 Functions

ASTEROID MINING

Asteroid Mining is a software application game. The main task of the game is building of a space station in the Asteroid. The players control the settlers. A settler function is to be moving around move around the Asteroid belt alone in search of resource of materials that will be used in building the space station.

The important resources are water ice, iron, carbon, uranium etc. These resources are buried in the core of the Asteroids. The surface of each Asteroids is covered in rocks. The depth of this rock mantle can vary from asteroid to asteroid. There some highly radioactive resources like uranium. There are also hallow asteroids, these type of asteroids cores are empty. Each Asteroid core is always homogeneous; it is always made of a single kind of a material all over that asteroid belt.

A settler controlled by a player can only do one operation in a single move. There are several operations, like travelling, drilling, mining, building robots, building teleportation-gates, etc. When a settler travels, they go to a neighboring or adjacent asteroid. An asteroid may have multiple hundreds of neighbors. When a settler drills, they deepen the hole in the mantle with one unit. When a settler mines, they extract the resource from the core of the asteroid. This is only possible if the mantle has been completely drilled through. A single settler can only carry 10 units of resources, that's the spaceships' capacity. A hollow asteroid, however, can be filled with a unit of resource, counting as an operation.

There are some dangers lurking in the asteroid belt. When a fully drilled asteroid with a radioactive core is at perihelion, the asteroid explodes and kills any settler on it. Radioactive material, therefore, can only be mined when the asteroid is at aphelion. Sometimes a sun storm reaches the asteroid belt, and it is also dangerous to the settlers. A settler can only survive a sun storm if they hide in the core of a hollow asteroid. This, of course, is only possible if the mantle has been drilled through.

The settlers can build autonomous robots with a unit of iron, a unit of carbon and a unit of uranium, that is controlled by artificial intelligence. These robots can only travel between asteroids and drill holes. Robots cannot mine because they are unable to transport things. Robots, however, can survive radioactive explosions, and in this case, they land on a neighboring asteroid. Sun storms do damage robots unless they hide in a hollow asteroid.

Two units of iron, a single unit of water ice and a single unit of uranium, the settlers can build a pair of teleportation-gates. The gates can be later deployed in the vicinity of the asteroid the settler is on. The two gates of a pair remain in contact for good and entering either the traveler which can be the settler, robot etc., will be immediately transported to the other. The settler can bring the freshly built gates with themselves, but at the same time a single settler can only bring two gates.

The game can end in two ways. If all the settlers die, the players lose. If, however, they can mine at least three units of each resource and they collect those materials on a single asteroid, they can build the space station and the players win the game.

2.2.3 Users

Average asteroid mining simulation enjoyers in this case the Players.

2.3 Requirements

2.3.1 Functional requirements

ID	Description	Check	Priority	Source	Use-case	Content
R01	The settlers must build a space station in the asteroid belt.	demonstration	should have	problem description	Build Space Station	

R02	The settlers have to mine resources from the asteroids.	demonstration	should have	problem description	Control Settler, View Asteroid Belt, Control Operations	
R03	Players of the game control the settlers.	demonstration	must have	problem description	View Asteroid Belt Control Settler	
R04	Asteroids are covered in rocks.	demonstration	should have	problem description	View Asteroid Belt Control Asteroid	
R05	The depth of the rock mantle can vary from asteroid to asteroid.	demonstration	should have	problem description	View Asteroid Belt Control Asteroid	
R06	There are dangerous asteroids, whose core is highly radioactive.	demonstration	should have	problem description	Control Asteroid	
R07	There can be hollow asteroids, whose core is empty.	demonstration	could have	problem description	Control Asteroid	
R08	The core of an asteroid is always homogeneous; it is always made of a single kind of a material.	testing	should have	problem description	Control Dangers Control Resources	
R09	A settler can do one operation in a single move.	demonstration	must have	problem description	Control Settler	

R10	A settler can travel.	demonstration	should have	problem description	Control Settler	
R11	A settler can drill.	demonstration	should have	problem description	Control Settler	
R12	A settler can mine.	demonstration	should have	problem description	Control Settler	
R13	A settler can build robots.	demonstration	should have	problem description	Control Settler	
R14	A settler can build teleportation gates.	demonstration	should have	problem description	Control Settler	
R13	When a settler travels, they go to a neighboring asteroid.	demonstration	should have	problem description	Control Settler	
R14	An asteroid may have multiple hundreds of neighbors.	demonstration	should have	problem description	View Asteroid Belt	
R15	When a settler drills, they deepen the hole in the mantle with one unit.	demonstration	must have	problem description	Control Settler	
R16	When a settler mines, they extract the resource from the core of the asteroid.	testing	must have	problem description	Control Settler Control Resources	
R17	Mining is only possible if the mantle has been completely drilled through.	testing	must have	problem description	Control Settler View Asteroid Belt	

R18	A settler can only carry 10 units of resources.	testing	must have	problem description	Control Settler Control Resources	
R19	Settler can store one unit of a resource in a hollow asteroid.	testing	must have	problem description	Control Settler Control Resources	
R20	If a settler drills into the core of a radioactive asteroid, and the asteroid is at perihelion, the asteroid explodes, and the settler dies.	testing	must have	problem description	Control Asteroid Control Settler	
R21	Radioactive material can only be mined when the asteroid is at aphelion.	testing	must have	problem description	Control Asteroid Control Settler Control Resources	
R22	A settler can only survive a sun storm if they hide in the core of a hollow asteroid.	demonstration	should have	problem description	Control Sunstorm Control Settler	
R23	Hiding is only possible if the mantle has been drilled through.	demonstration	must have	problem description	Control Asteroid Control Settler	
R23	Robots can travel between asteroids.	demonstration	should have	problem description	Control Robot	
R24	Robots can drill holes.	demonstration	should have	problem description	Control Robot	
R25	Robots cannot mine.	demonstration	should have	problem description	Control Robot	

R26	Robots cannot transport the resources.	demonstration	should have	problem description	Control Robot Control Resources	
R27	Robots survive radioactive explosions, and they land on a neighboring asteroid.	demonstration	should have	problem description	Control Robot Control Asteroid	
R28	Sun storms do damage robots unless they hide in a hollow asteroid.	demonstration	must have	problem description	Control Robot Control Sunstorms	
R29	If all the settlers die, players lose.	demonstration	must have	problem description	Control Settler	
	A settler can only carry 2 teleportation gates.	testing	must have	problem description	Control Settler	
R30	If the players mine at least 3 units of each resource and they collect those materials on a single asteroid, they can build the space station and the players win the game.	testing	must have	problem description	View Asteroid Belt Control Resources Control Settler	

2.3.2 Resource requirements

ID	Description	Check	Priority	Source	Comment
R01	The application will be programmed in Java.	demonstration, testing	must have	brainstorming	
R02	In order to run the application, JDK 8, or higher is recommended.	demonstration, testing	should have	brainstorming	
R03	The application is designed to run on Windows 10.	demonstration, testing	must have	brainstorming	
R04	The program will need user permission for file management.	demonstration, testing	must have	brainstorming	
R05	Recommended IDEs are IntelliJ IDEA, and Eclipse.	demonstration, testing	should have	brainstorming	
R06	Recommended tools for diagrams are STAR UML, and IDEA UML Plugins.	demonstration, testing	should have	brainstorming	

2.3.3 Non- functional requirements, Restrictions

ID	Description	Check	Priority	Source	Comment
R01	The application is designed to run on Windows 10.	demonstration, testing	must have	brainstorming	availability
R02	The user should have the basic knowledge of the game controls to play the game.	demonstration, testing	should have	brainstorming	learnability
R03	The user machine should have enough memory and storage to run the application successfully.	testing	should have	brainstorming	performance, testability

2.4 Essential use-case

2.4.1 Use-case diagram

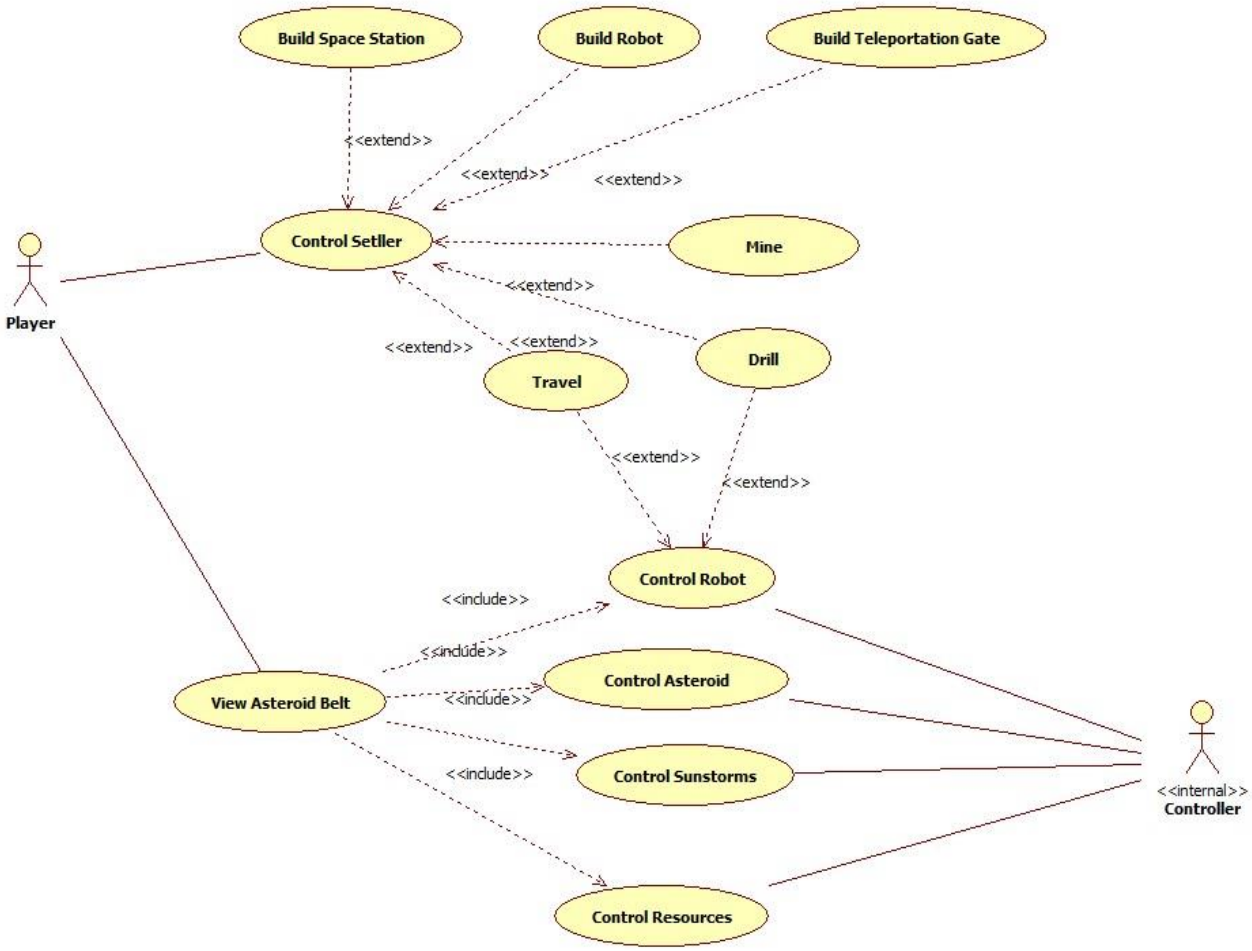


Figure 1-Arthur, 2021

2.4.2 Use-case descriptions

Title	Control Settler
Description	The player controls the settler.
Actors	Player
Main success scenario	The players win the game.
Alternate scenario	If all the settlers die, players lose.
Title	**View Asteroid Belt**
Description	The player views the asteroid belt.
Actors	Player
Main success scenario	1. The player views the Asteroid belt. 2. The system displays the whole map of the Asteroid belt.
Description	The player controls the settler to build a space station.
Title	**Build Space Station**
Actors	Player
Main success scenario	If the players mine at least one unit of each resource and they collect those materials on a single asteroid, they can build the space station and the players win the game.
Title	**Build Teleportation Gate**
Description	The player controls the settler to build teleportation gates..
Actors	Player
Main success scenario	If the player builds at least 2 teleportation gates, the player can use them to quickly travel.
Alternate Scenario	The player is unable to travel if there exists only 1 teleportation gate. A second gate is required to successfully travel.

Title	**Build Robot**
Description	The player builds robots.
Actors	Player
Main success scenario	If the player has sufficient resources, a robot can be built to assist the player.
Alternate Scenario	The player is unable to build a robot in case of insufficient resources.
Title	**Control Robot**
Description	Robots can help the settlers.
Actors	Controller
Main success scenario	1. Robots can travel between asteroids. 2. Robots can drill holes.
Alternate scenario	1. Robots cannot mine. 2. Robots cannot transport the resources.
Title	**Control Resources**
Description	Settlers need resources to build the space station, teleportation gates, and robots.
Actors	Controller
Main success scenario	Settlers extract the resource from the core of the asteroid.
Main success scenario	An asteroid can only have a single kind of material (water ice, carbon, etc.)

Title	Control Asteroid
Description	The asteroids are covered in rocks which can be drilled.
Actors	Controller
Main success scenario	1. The asteroid may contain resources that can be mined.
Alternate scenario	1.A. The asteroid may have a radioactive core. 1.A.1. If a settler drills into the core of a radioactive asteroid, and the asteroid is at perihelion, the asteroid explodes, and the settler dies. 1.A.1. If a robot drills into the core of a radioactive asteroid, and the asteroid is at perihelion, the asteroid explodes and damages the robot. 1.A.3. Radioactive material can only be mined when the asteroid is at aphelion.
Alternate scenario	1.B. The asteroid may be hollow. 1.B.1. A hollow asteroid cannot be mined.
Description	Sun storms are dangerous to the settlers and robots.
Title	**Control Sun storms**
Actors	Controller
Main success scenario	1. Sun storms do damage to settlers' health. 2. Sun storms do damage to robots' health.
Alternate scenario	Settlers can hide in the core of a hollow asteroid to avoid the sun storms.
Alternate scenario	Robots can hide in the core of a hollow asteroid to avoid the sun storms.

2.5 Glossary

Term	Description	Type
Settler	The miner spaceship with a 10 unit carry capacity.	Object
Travel action	Moves the Settler to a neighboring asteroid.	Action
Drill action	Destroys asteroid mantle by a unit. Drops nothing.	Action
Mine action	Destroys asteroid cores by a unit. Drops minerals.	Action
Fill action	Dispenses a unit of resource into the asteroid core.	Action
Hide action	Settler enters the asteroid core.	Action
Perihelion	The point in the orbit of an asteroid, at which it is closest to the sun.	Location
Radioactive explosion	Triggered when a radioactive asteroid comes at the perihelion. Kills any settler on the asteroid.	Event
Aphelion	The point in the orbit of an asteroid, at which it is farthest from the sun.	Location
Sun storm	Naturally occurring phenomena that may reach the asteroid belt. Fatal on contact.	Event
Robots	Autonomous robots controlled by artificial intelligence. Same as Settlers, except: - Can’t mine - Can’t take damage from radioactive explosion, Lands on any neighboring asteroid instead - Can’t build teleportation gates	Object
Space station	The win condition of the game. Can be built by filling an asteroid with 3 units of each resource	Event
Teleportation gate	Deployed in the vicinity of the asteroid the settler is on. The settler can bring the freshly built gates with themselves. Gate carrying capacity is 2.	Object
Mantle	Asteroid crust.	Object

2.6 Project plan

The purpose of this project is to challenge our interpersonal, organizational, and technical skills. The objective is to build, and deliver a product in a punctual manner. We expect to successfully deliver a proper user interface, fast and lightweight application, meet all the requirements and expectations, within the given timeframe.

List of the artifacts to be created and delivered:

Delivery date	Artifacts
Feb. 22	Requirements, project, functionality -- documentation
Mar. 1	Analysis Model - first version -- documentation
Mar. 8	Analysis Model - final version -- documentation
Mar. 16	Planning the Skeleton -- documentation
Mar. 22	Skeleton program -- documentation and source code
Mar. 29	Concept of Prototype -- documentation
Apr. 12	Detailed plans -- documentation
Apr. 19	Prototype program -- documentation and source code
Apr. 26	User interface specification -- documentation
May 10	Complete program -- documentation and source code
May 18	Summary

2.7 Organizational Structure

While each team member are equally responsible for the development of the application, each have special role to support the development processes

Name	**Role**	**Description**
Ashiq Muhammad	Tester	Responsible for the quality of software development, and deployment. Performs automated and manual tests to ensure the software created by developers is fit for purpose.
Ibrar Shaqoor	Documentation Lead	Lead the team to prepare, edit, and publish product documentation for software releases including user guides, release notes, knowledge base articles, and other resources.
Wali Ullah	Project Manager	Responsible for planning, overseeing and leading projects from ideation through to completion.
Asad Idrees Razak	Tester	Responsible for the quality of software development, and deployment. Performs automated and manual tests to ensure the software created by developers is fit for purpose.
Igbineweka Vincent	Quality Assurance	Responsible for guaranteeing a level of quality for the end client, and to help the team to identify problems early in the process.
Orgil Luis Macha	UX/UI Designer	Responsible for creating the optimal, and pleasant user experience.

2.8 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
20.02.2021. 14:00	1	Team	Meeting: Team meeting intended to divide the tasks for this document. Ashiq and Asad Idrees are responsible for 2.1. Ashiq, Asad Idrees, and Vincent are responsible for 2.2. Ibrar and Wali are responsible for 2.3 and 2.4. Luis is responsible for 2.5 and 2.6.
21.02.2021. 13:00	1	Team	Meeting: Brainstorming
21.02.2021. 14:00	2	Ibrar Shakoor	completed artifact 2.3: Requirements and Documentation Formatting
21.02.2021. 16:00	2	Wali Ullah	completed artifact 2.4: Essential Use-Cases
21.02.2021. 16:13	1	Luis	completed artifact 2.5: Glossary
21.02.2021. 14:30	1.5	Asad Idrees	completed artifact 2.1 and 2.2: Introduction and Overview
21.02.2021. 14:30	1.5	Ashiq	completed artifact 2.1 and 2.2: Introduction and Overview
21.02.2021. 15:00	1.5	Vincent	completed artifact 2.2: Overview
21.02.2021. 22:00	1	Luis	completed artifact 2.6: Project plan

NR#3 Analysis Model - version 1

4 – Six Stars

Supervisor:
Dr. Balla Katalin

Members:

Name	Neptun	Email
Ashiq Muhammad	EDOXAM	muhammadashiqse10@gmail.com
Ibrar Shakoor	FS4J6L	ibrarraja11222@gmai.com
Wali Ullah	WTW2SV	wali6985@gmail.com
Asad Idrees Razak	XJJUK4	idreesshaikh6300@gmail.com
Igbineweka Vincent	TOM9S6	ifyvinz@gmail.com
Orgil Luis Macha	VH0JMU	orgilluismacha@gmail.com

3. Analysis model – version 1

3.1 Object catalog

3.1.1 Game

This object incorporates the logic of the game. It also keeps tracks of the players, dangers, resources, and executes the correct scenario depending on the current situation of the game.

3.1.2 Game Mission

Game mission object will store which conditions are required to win or lose the game. It also controls the amount of resources needed to build the space station.

3.1.3 Asteroid Belt

Asteroid belt acts like a map and determines the total asteroids present in the asteroid belt and acts as a navigation for travelling across the asteroids.

3.1.4 Location

Location object determines the current location of the settler/robot inside the asteroid belt. It is required for travelling and teleportation as it can also determine the next possible location for a player/robot to travel inside the asteroid belt.

3.1.5 Asteroid

Asteroid object keeps track of the type of asteroid, if it's hollow or not. It also determines which resources and the amount of the resources are present on a particular asteroid. It also knows the radioactive states of the asteroid i.e; whether it’s at perihelion or aphelion.

3.1.6 Dangers

A Danger object determines the dangers towards the settler or robot. Dangers can be radioactive blast, or sunstorm. The amount of damage done to the settler or the robot can also be determined by this object.

3.1.7 Resources

Resources object determines the type of the resources present on an asteroid and their particular characteristics. E.g: Uranium can be radioactive, or water-ice can undergo sublimation under special circumstances.

3.1.8 Travel

This object implements the travelling methods used by settlers or robots. There are 2 types of travelling methods: Normal Travel which takes longer, and Teleportation which is faster. This object also determines the amount of distance that can be travelled in one move.

3.1.9 Player

This object contains the type, health, and other characteristics of a player. Player can be a settler or a robot. Settlers can revive their health if they don't encounter a danger for a specific time.

3.1.10 Character Type

Describes the type of the player character. The type is either settler, or a robot.

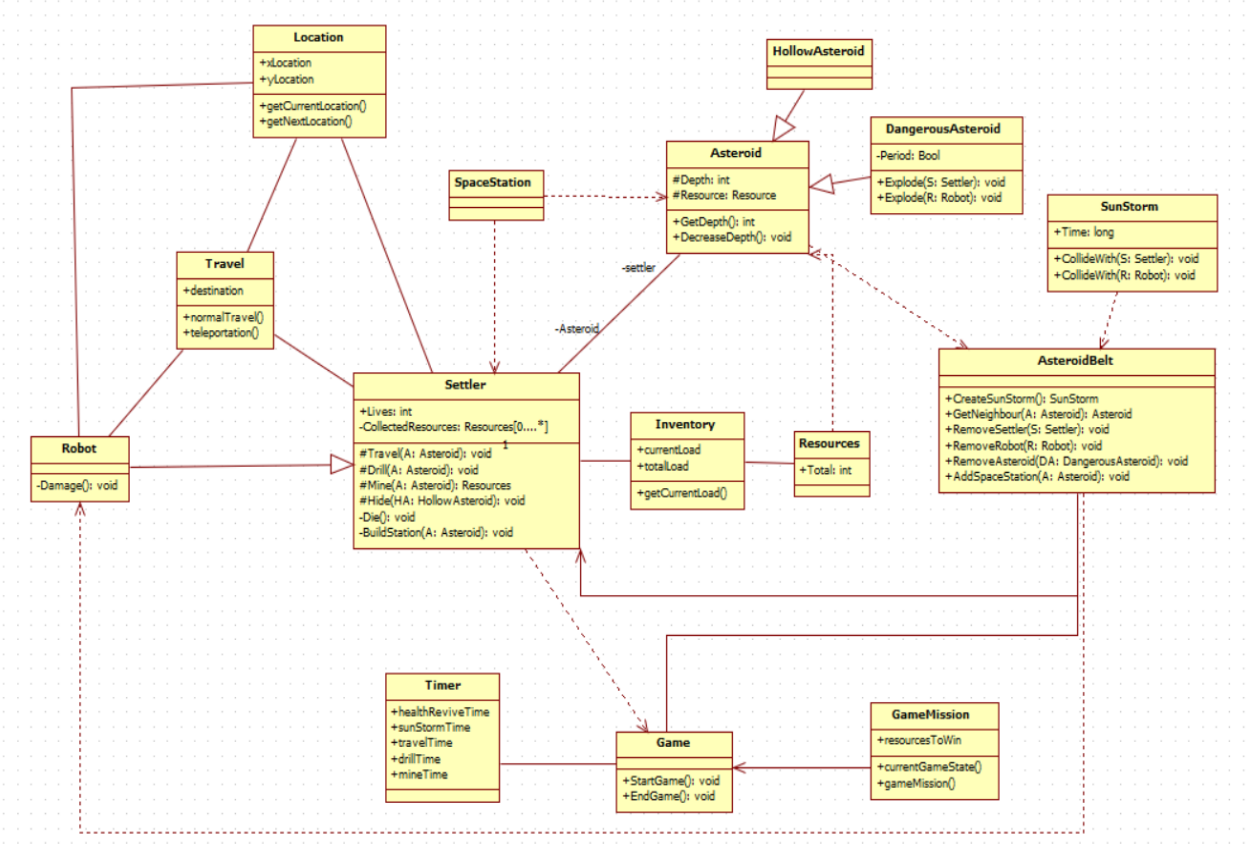
3.1.11 Timer

This object keeps track of the time for different scenarios like sun storms, travelling times, health revive time, day/night times, and times for performing different operations like drilling, mining etc.

3.1.12 Inventory

This object will determine which resources a settler has mined, and what is the total carrying capacity of a settler. The inventory is emptied if a settler dies, and the resources are lost.

3.2 Static structure diagrams



3.3 Class description

3.3.1 Asteroid

- **Responsibility**

To serve as a source for the settler to collect the resources to build the space station. It also contains information about the asteroid, whether it is radioactive or hollow.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+Depth:int** – This variable stores the depth of the asteroid
  - **+Resource:Resources** – This variable stores the type of resources which can be found by the settler

- **Methods**
  - **int GetDepth()**: Returns an integer variable to show the depth of the asteroid/rocks
  - **void DecreaseDepth()**: If the settler mines the asteroid, the depth is decreased with the help of this method.

3.3.2 HollowAsteroid

- **Responsibility**

To protect the settler and the robot from the sun storm.

- **Superclasses**

Asteroid.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**

  -

3.3.3 DangerousAsteroid

- **Responsibility**

To carry radioactive resources and explode if the settler is mining/drilling it during the perihelion period.

- **Superclasses**

Asteroid.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **-Period:bool –** This variable stores the state of the rock according to the position of the sun (aphelion or perihelion)

- **Methods**
  - **void Explode(Settler s)**: This function explodes the asteroid in case of drilling and can cause the settler death
  - **void Explode(Robot r)**: This function explodes the asteroid but it keeps the robot intact

3.3.4 AsteroidBelt

- **Responsibility**

To serve as a basis for the settler to build the space station.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **void CreateSunStorm()**: This function creates the sun storm object
  - **Asteroid GetNeighbour(Asteroid a)**: Returns the object of neighbouring asteroid for the settler
  - **void AddSettler(Asteroid a, Settler s)**: Adds the settler in the chosen asteroid
  - **void AddRobot(Asteroid a, Robot r):** Adds the robot in the chosen asteroid to protect the settler
  - **void RemoveSettler(Settler s):** Moves/Remove the settler from the asteroid belt
  - **void RemoveSettler(Settler s):** Moves/Remove the robot from the asteroid belt

3.3.5 Settler

- **Responsibility**

Settler travel, drill and mine the resources from the core of asteroid to build the space station

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

  - **+lives:int** – This variable stores the presence of settlers in the asteroid belt as an integer number.

  - **-collectedResources:Resources[0..*]** – This array of integers stores the number of resources collected by settlers.

- **Methods**
  - **void Travel(Asteroid a)**: Settler moves around neighbour asteroid to find the resources for his survival
  - **void Drill(Asteroid a)**: The settler drills the asteroid to make it hollow for his survival
  - **Resources Mine(Asteroid a)**: Settler collects the resources from the asteroid and add them into collected resources list
  - **void Hide(Asteroid a)**: Settler needs to drill the asteroid to make it hollow and hide itself for protection against radioactive explosion and sun storm
  - **void Die()**: Settler dies due to the sun storm, radioactive explosion or improper drilling of the asteroid
  - **void BuildSpacestation(Asteroid a)**: Settler collects resources from neighbouring asteroids to a single asteroid to build the space station

3.3.6 Robot

- **Responsibility**

Help the Settlers to drill the asteroid's rock. Robots cannot be destroyed by radioactive explosions.

- **Superclasses**

Settler.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

  - **Damage():void** – If the robot cannot find a hollow asteroid, they can be destroyed by the sun storm.

- **Methods**
  - **void Damage()**: If the robot cannot find a hollow asteroid, they can be destroyed by the sun storm.

3.3.7 Resources

- **Responsibility**

Indicates the resources present in the asteroid.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+Total:int –** This variable stores the number of resources collected by the settler
- **Methods**
  - **-**

3.3.8 Space Station

- **Responsibility**

Helps the settler to survive on the asteroid. It constitutes the main goal of the game.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **-**

3.3.9 Game

- **Responsibility**

Player handles the movement of the settler and robot.

- **Superclasses**

There is no Superclass

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **void StartGame()**: Player starts the game at any time
  - **void StartGame()**: Player ends the game at any time

3.3.10 GameMission

- **Responsibility**

Tells the condition for winning or losing (in regards to the amount of resources for building the space station) and gives the current state of the game.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+resourcesToWin:** Amount of resources needed to build the space station therefore winning the game

- **Methods**
  - **void currentGameState()**: Tells the current state of the game
  - **gameMission()**: Sets the requirements of the game e.g. the amount of resources needed to build the space station

3.3.11 SunStorm

- **Responsibility**

To cause damage to both the Settlers and Robots for a particular duration of time.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+Time:Long** – This variable stores the arrival time of the sun stom.
- **Methods**
  - **CollideWith(Settler s)**: This function destroys the settler upon collision
  - **CollideWith(Robot r)**: This function confirms the damage of the robot in case of a collision

3.3.12 Location

- **Responsibility**

Tells the current position of both settler and robot with respect to their x and y coordinates in the asteroid belt. Also tells the next position change of Settler/Robots in the asteroid belt after going through the Teleportation Gate.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+xLocation:int** X coordinate of the position of the Settler in the asteroid belt.
  - **+yLocation:int** Y coordinate of the position of the Settler in the asteroid belt.
- **Methods**
  - **Asteroid GetCurrentLocation()**: Gives the current location of the settler or robot on the asteroid belt
  - **Asteroid GetNextLocation():** Tells about the next location of the settler/robot

3.3.13 Travel

- **Responsibility**

Tell the travelling method used by Settler or the Robot, and also determine the amount of distance covered by them.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+destination:int** This variable stores the distance by which the Setter/Robot is travelling to (change of position)

- **Methods**
  - **NormalTravel()**: It is the slower kind of traveling. If settler or robot travel using this, they take lot of time
  - **Teleportation()**: It is the special travelling method using teleportation gates

3.3.14 Timer

- **Responsibility**

It tells about the required time to perform different operations like drilling, mining, travelling, reviving the settler and also the duration of the sunstorm.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+healthReviveTime:int -** This variable stores the amount of time it takes to revive either Settler or Robot
  - **+sunStormTime:int -** This variable stores the amount of time for which the SunStorm hits
  - **+travelTime:int -** This variable stores the time taken for Settler/Robot to travel from one place to another(change of location)
  - **+drillTime:int -** This variable stores the time taken for drilling the asteroid core
  - **+mineTime:int -** This variable stores the time taken to mine the asteroid core

- **Methods**
  - **-**

3.3.15 Inventory

- **Responsibility**

It determines the type and total amount of resources a settler has mined and also the total carrying capacity of the settler.

- **Superclasses**

There is no Superclass.

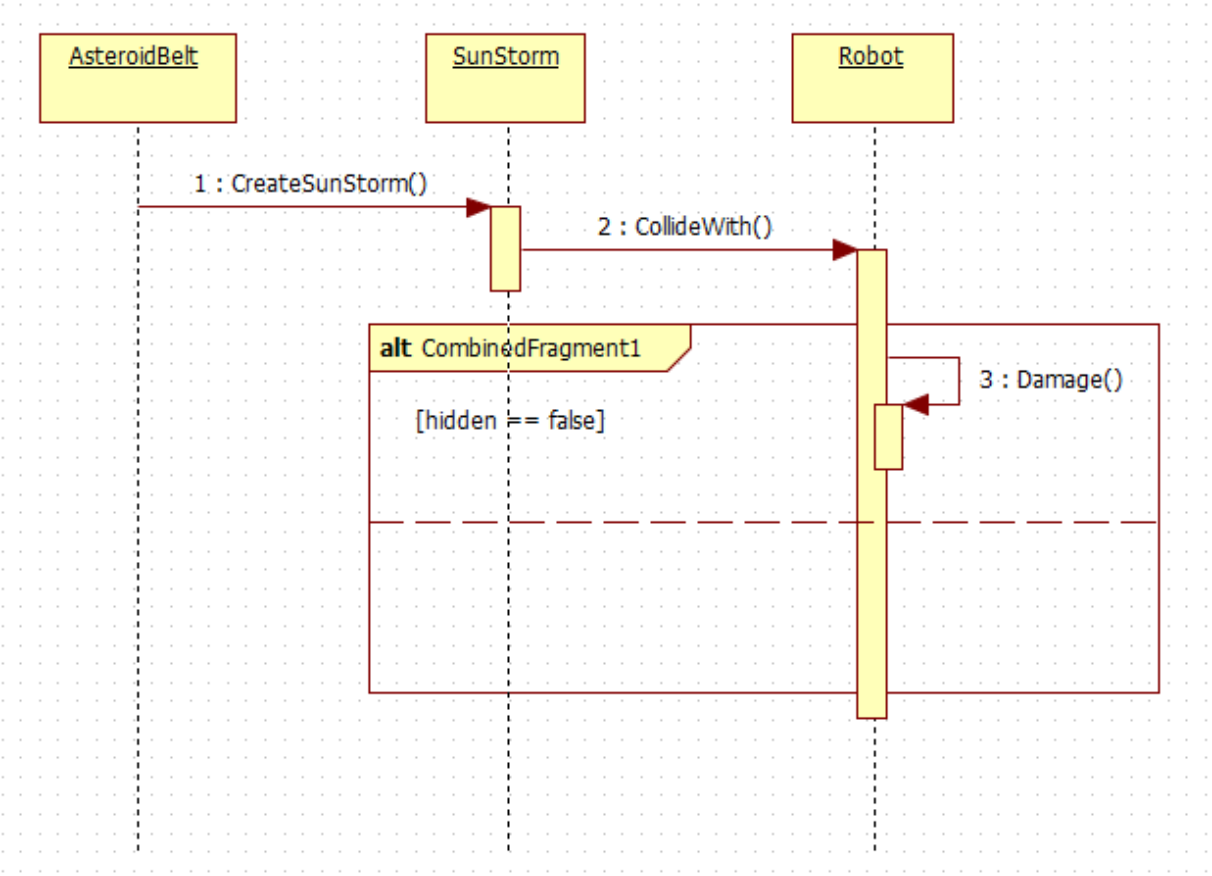
- **Interfaces**

There are no interfaces implemented in this class.

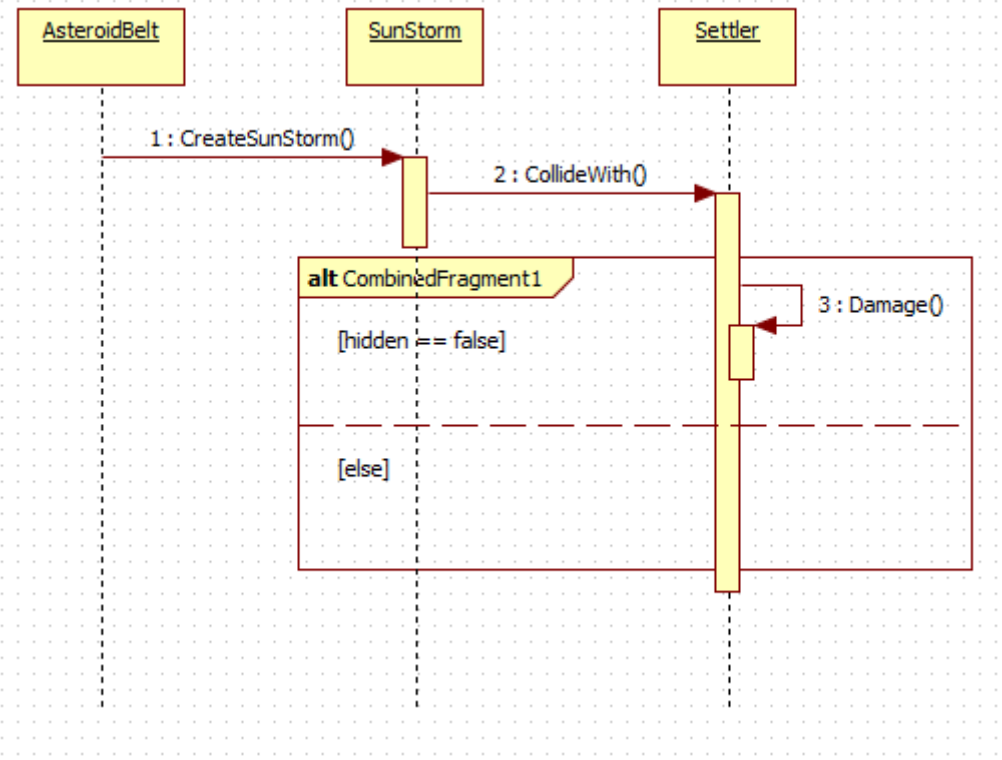
- **Attributes**
  - **+currentLoad:ArrayList<int> -** This the variable stores the current amount of resources(current capacity) that has been mined by the Settler
  - **+totalLoad:ArrayList<int> -** This variable stores the total amount of mined resources(total capacity) a Settler can carry

- **Methods**
  - **GetCurrentLoad():** Gives the total amount of the resources mined by the settler

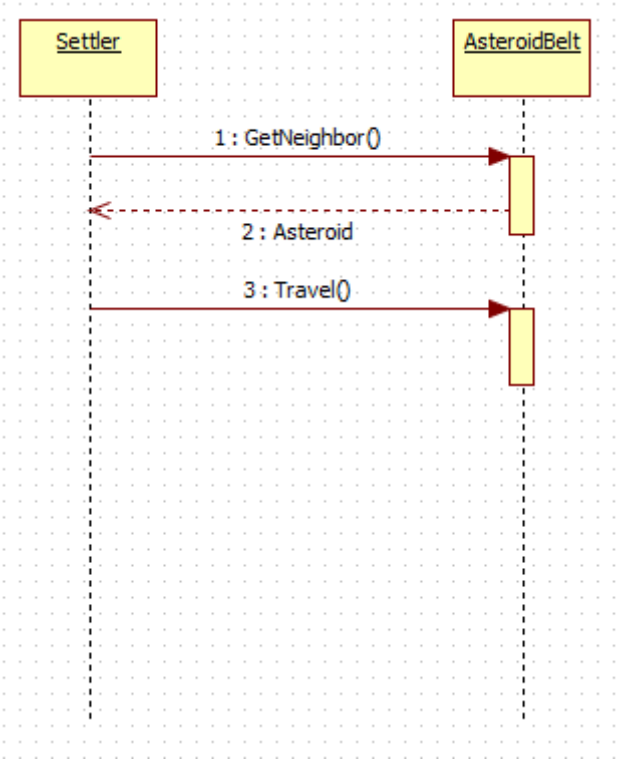
3.4 Sequence diagrams



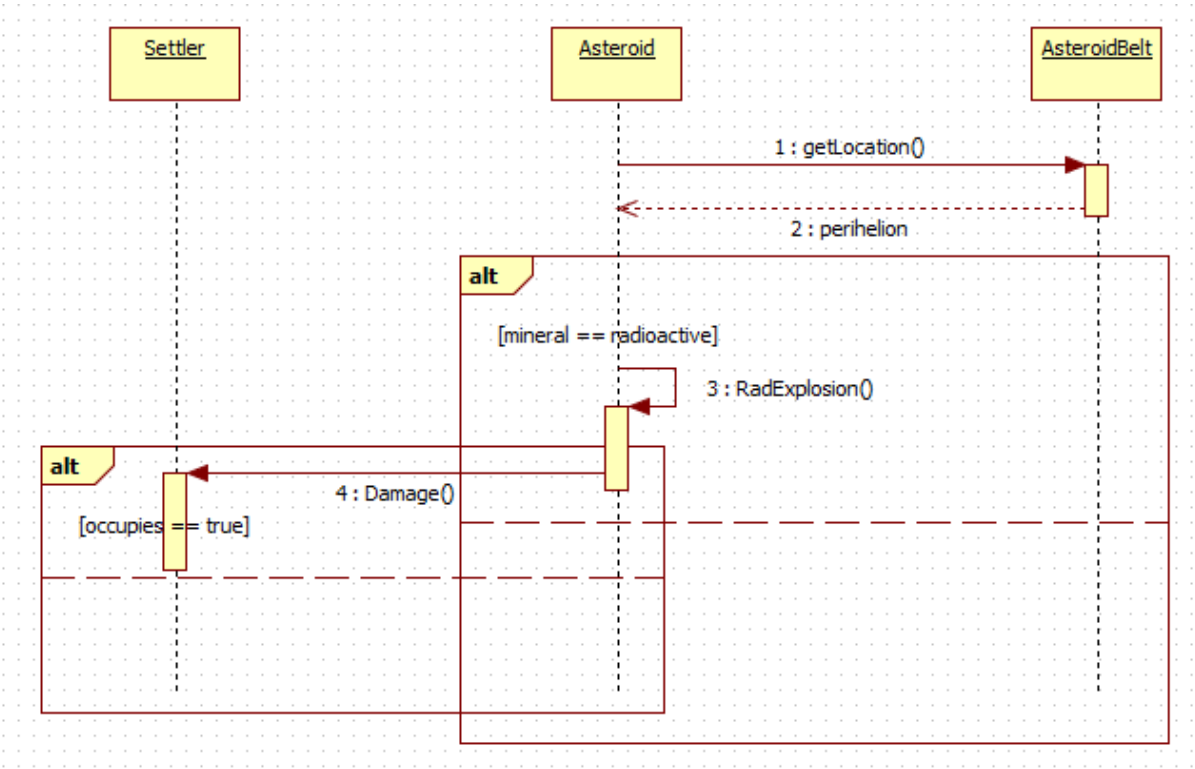
Sun storm strikes robot



Sun storm strikes settler

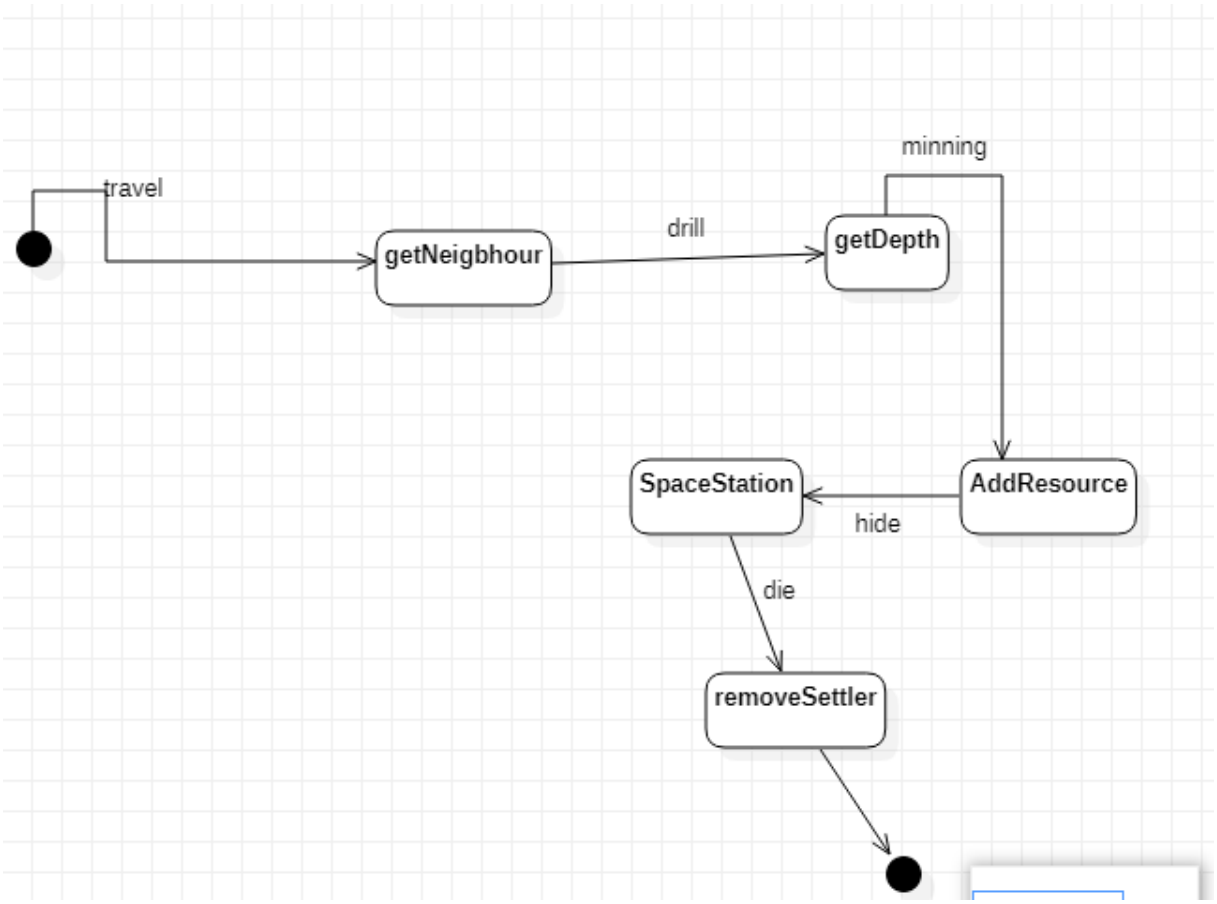


Settler/robot travels



Radioactive explosion with settler

3.5 State-charts



3.6 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
25.02.2021. 18:00	2	Team	Meeting: Decisions were made regarding this document and responsibilities were divided as follows: Wali Ullah 3.1 Ibrar Shakoor 3.2 Ashiq and Idrees 3.3 Orgil 3.4 Vincent 3.5
26.02.2021. 12:00	1+1.5	Wali Ullah	Brainstorming + Completed artifact 3.1
26.02.2021. 3:00	2+1	Ibrar Shakoor	Brainstorming + Completed artifact 3.2
26.02.2021. 22:00	1 + 2	Asad Idrees	Brainstorming + Completed artifact 3.3
26.02.2021. 22:00	1 + 2	Ashiq	Brainstorming + Completed artifact 3.3
27.02.2021. 18:25	1	Orgil Luis Macha	Completed artifact 3.4
28.02.2021	1	Vincent	Completed artifact 3.5

4. Analysis model – version 2

4.1 *Object catalog*

4.1.1 Game

This object incorporates the logic of the game. It also keeps tracks of the players, dangers, resources, and executes the correct scenario depending on the current situation of the game. It also stores which conditions are required to win or lose the game. It also controls the amount of resources needed to build the space station. It will be accessible by the players of the game.

4.1.2 Asteroid Map

Asteroid map acts like a map and determines the total asteroids present in the asteroid belt and acts as a navigation for travelling across the asteroids. At any given time, it can be used by the player of the game to check the neighboring asteroids. It can be accessed by settlers and robots. It can also keep track of settlers and robots present in the asteroid belt.

4.1.3 Location

Location object determines the current location of the settler/robot inside the asteroid belt. It is required for travelling and teleportation as it can also determine the next possible location for a player/robot to travel inside the asteroid belt.

4.1.4 Asteroid

Asteroid object keeps track of the type of asteroid, if it's hollow or not. It also determines which resources and the amount of the resources are present on a particular asteroid. It also knows the radioactive states of the asteroid i.e; whether it’s at perihelion or aphelion. It also knows if there is a teleportation gate constructed on the asteroid or not.

4.1.5 Radioactive Asteroid

This object is responsible for the radioactive explosions that can occur on an asteroid. It keeps track of the radioactive phases (aphelion/perihelion). When the explosion happens, the settler loses all its health, but a robot can survive. This behavior is controlled by this object.

4.1.6 SunStorm

This object is responsible for the sun storms that can occur in the asteroid belt. The amount of damage done by the sunstorm to the settler can also be determined by this object. Settlers and robots can avoid being damaged by sun storms if they hide inside a hollow asteroid.

4.1.7 Resources

Resources object determines the type of the resource present on an asteroid. It also determines all the different types of resources/minerals present in the asteroid belt. If a new type of resource/mineral is added to the game, or removed, this object also keeps track of such changes.

4.1.8 Travel Method

This object implements the travelling methods used by settlers or robots. There are 2 types of travelling methods: Normal Travel which takes longer, and Teleportation which is faster. This object also determines the amount of distance that can be travelled in one move.

4.1.9 Settler

This object contains health, and other functions of a settler such as drilling, mining, travelling, etc. It also allows the settler to hide during a sunstorm, travel or teleport, die or revive the health and other functions. The settler also knows how many teleportation gates it is carrying, and the pairs of the gates.

4.1.10 Robot

This object shares some common behavior as a settler such as health, and other functions such as drilling, and travelling, etc. It does not allow the robot to mine, or carry resources.

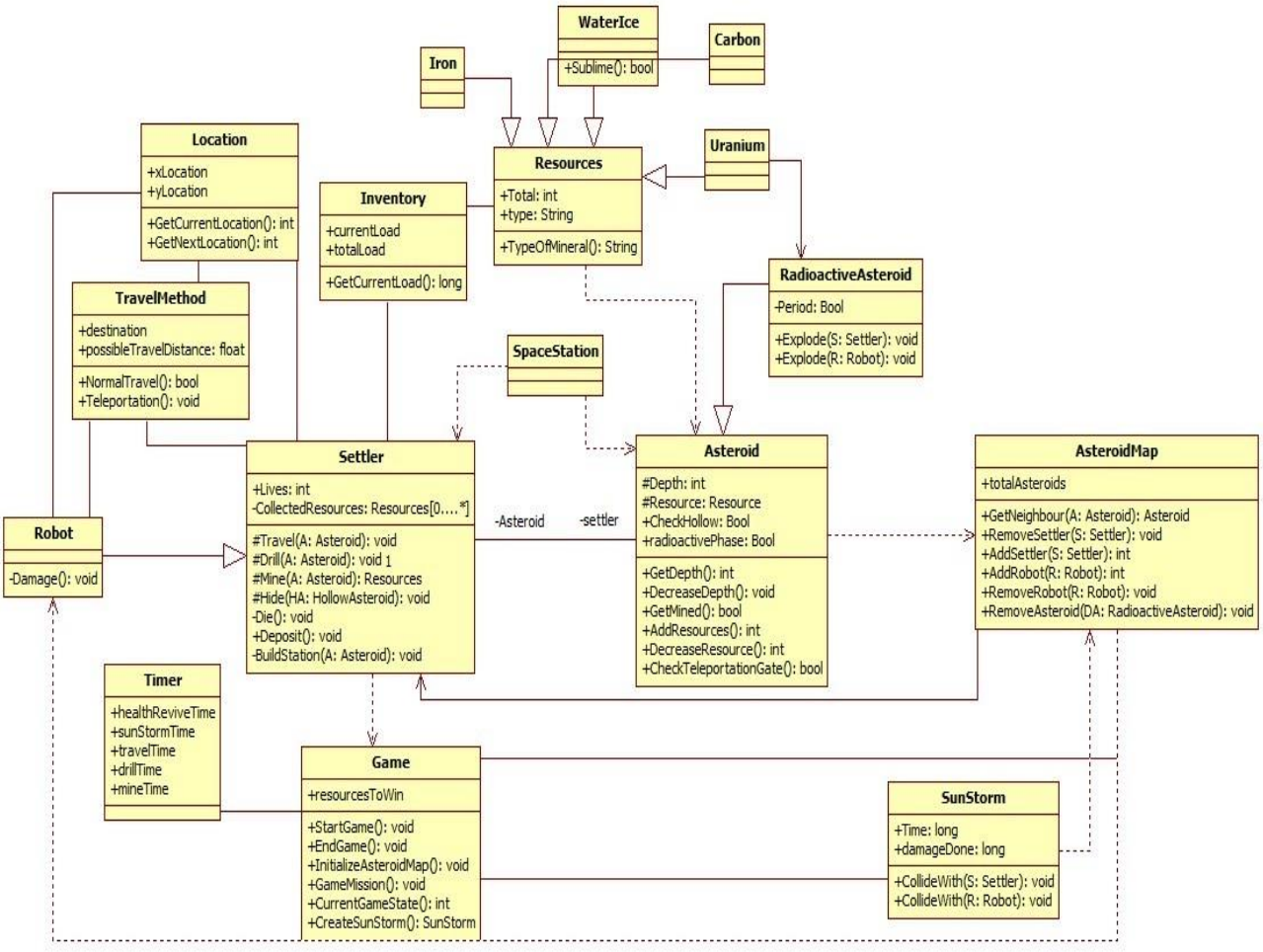
4.1.11 Timer

This object keeps track of the time for different scenarios like sun storms, travelling times, health revive time, day/night times, and times for performing different operations like drilling, mining etc.

4.1.12 Inventory

This object will determine which resources a settler has mined, and what is the total carrying capacity of a settler. The inventory is emptied if a settler dies, and the resources are lost. A settler can always access its inventory to check which resources/minerals it is carrying.

4.2 Static structure diagrams



4.3 Class description

4.3.1 Asteroid

- **Responsibility**

To serve as a source for the settler to collect the resources to build the space station. It also contains information about the asteroid, whether it is radioactive or not as well as hollow or not.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **#Depth:int** – This variable stores the depth of the asteroid
  - **#Resource:Resources** – This variable stores the type of resources and their amounts which can be found by the settler
  - **+CheckHollow:Bool** – This variable checks whether the Asteroid is hollow or not
  - **+radioactivePhase:Bool** – This variable tells whether the state of radioactive asteroid is at perihelion or aphelion

- **Methods**
  - **+int GetDepth()**: Returns an integer variable to show the depth of the asteroid/rocks
  - **+void DecreaseDepth()**: If the settler mines the asteroid the depth value of that particular asteroid is changed
  - **+Bool GetMined()**: It shows whether the asteroid is minable or not
  - **+int AddResources():** This method assigns the resources to the asteroids at the start of the game
  - **+int DecreaseResource():** If the settler mines the asteroid the amount of resources of that particular asteroid is decreased with the help of this method.
  - **+Bool CheckTeleportationGate():** It checks whether there is a teleportation gate constructed on the asteroid or not.

4.3.2 AsteroidMap

- **Responsibility**

It determines the total asteroids present in the asteroid belt and helps us to navigate to travel across the asteroids.

- **Superclasses**

Asteroid.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+totalAsteroids:ArrayList<Asteroid>** – This variable shows all the asteroids instantiated and contained in a list

- **Methods**
  - **+Asteroid GetNeighbour(Asteroid a)**: Returns the object of neighbouring asteroid for the settler
  - **+void AddSettler(Asteroid a, Settler s)**: Adds the settler in the chosen asteroid
  - **+void AddRobot(Asteroid a, Robot r):** Adds the robot in the chosen asteroid to assist the settler
    +void RemoveSettler(Settler s): Moves/Remove the settler from the asteroid
  - **+void RemoveRobot(Robot r):** Moves/Remove the robot from the asteroid
  - **+void RemoveAsteroid(RA: RadioactiveAsteroid):** If asteroid is radioactive this method is used to remove that asteroid to help us keep track of asteroids

4.3.3 RadioactiveAsteroid

- **Responsibility**

To carry radioactive resources and explode if the settler is mining/drilling it during the perihelion period.

- **Superclasses**

Asteroid.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **-Period:Bool –** This variable stores the state of the asteroid according to its position (aphelion or perihelion)

- **Methods**
  - **+void Explode(Settler s)**: This method is used for explosion of the asteroid while drilling by Settler causing its death
  - **+void Explode(Robot r)**: This method is used for explosion of the asteroid while drilling by Robot but keeping it intact

4.3.4 Settler

- **Responsibility**

Settler travel, drill, and mine the resources from the core of asteroid to build the space station

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+lives:int** – This variable stores the presence of settlers in the asteroid belt as an integer number.
  - **-collectedResources:Resources[0..*]** – This array of integers stores the number of resources collected by settlers.
- **Methods**
  - **#void Travel(Asteroid a)**: Settler moves around neighbour asteroid to find the resources for his survival
  - **#void Drill(Asteroid a)**: The settler drills the asteroid to make it hollow for its survival
  - **#Resources Mine(Asteroid a)**: Settler collects the resources from the asteroid
  - **#void Hide(Asteroid a)**: Settler needs to drill the asteroid to make it hollow and hide itself for protection against radioactive explosion and sun storm
  - **-void Die()**: Settler dies due to the sun storm, radioactive explosion or improper drilling of the asteroid
  - **+void Deposit():** The collected resources are added into the resource list
  - **-void BuildSpacestation(Asteroid a)**: Settler collects resources from neighbouring asteroids to a single asteroid to build the space station

4.3.5 Robot

- **Responsibility**

Help the Settlers to drill the asteroid's core. Robots cannot be destroyed by radioactive explosions.

- **Superclasses**

Settler.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **-void Damage()**: If the robot cannot find a hollow asteroid, they can be destroyed by the sun storm.

4.3.6 Resources

- **Responsibility**

Indicates the resources present in the asteroid.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+Total: int –** This variable stores the number of resources collected by the settler
  - **+Type: String –** It shows the type of the particular resource collected

- **Methods**
  - **+String TypeOfMineral():** This method determines the different type of resources/minerals present in the asteroid

4.3.7 Iron

- **Responsibility**

It is one of the minerals needed for the making of SpaceStation collected by Settler.

- **Superclasses**

Resources.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **-**

4.3.8 WaterIce

- **Responsibility**

It is one of the minerals needed for the making of SpaceStation collected by Settler.

- **Superclasses**

Resources.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **+Bool Sublime():** It checks whether the WaterIce is a solid or gas while mining.

4.3.9 Carbon

- **Responsibility**

It is one of the minerals needed for the making of SpaceStation collected by Settler.

- **Superclasses**

Resources.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  -

4.3.10 Uranium

- **Responsibility**

It is one of the minerals needed for the making of SpaceStation collected by Settler.

- **Superclasses**

Resources.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  -

4.3.11 Space Station

- **Responsibility**

Helps the settler to survive on the asteroid. It constitutes the main goal of the game.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**

  -

4.3.12 Game

- **Responsibility**

Tells the condition for winning or losing (in regards to the amount of resources for building the space station) and gives the current state of the game. Here, player controls the movement of the settler.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+resourcesToWin:Map<String, int> –** Amount of resources needed to build the space station therefore winning the game

- **Methods**
  - **+int currentGameState()**: Tells the current state of the game at any point
  - **+void StartGame()**: Player starts the game at any time
  - **+void EndGame()**: Player ends the game at any time or when game is lost or won
  - **+void GameMission()**: Sets the requirements of the game e.g. the amount of resources needed to build the space station
  - **+void InitializeAsteroidMap():** When the game is started, this method is used to initialize the asteroid map
  - **+void CreateSunStorm():** This method causes the sunstorm on the asteroid belt

4.3.13 SunStorm

- **Responsibility**

To cause damage to both the Settlers and Robots for a particular duration of time.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+Time:Long** – This variable stores the arrival time of the sunstorm
  - **+damageDone:Long** – This variable tells the amount damage caused to a Settler or Robot by decreasing their health
- **Methods**
  - **+void CollideWith(Settler s)**: This method destroys the settler upon collision
  - **+void CollideWith(Robot r)**: This method confirms the damage of the robot in case of a collision

4.3.14 Location

- **Responsibility**

Tells the current position of both settler and robot with respect to their x and y coordinates in the asteroid belt. Also tells the position change of Settler/Robots in the asteroid belt after going through the Teleportation Gate.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+xLocation:int** X coordinate of the position of the Settler in the asteroid belt.
  - **+yLocation:int** Y coordinate of the position of the Settler in the asteroid belt.
- **Methods**
  - **+Asteroid GetCurrentLocation()**: Gives the current location of the settler or robot on the asteroid belt
  - **+Asteroid GetNextLocation():** Tells about the next location of the settler/robot

4.3.15 TravelMethod

- **Responsibility**

Tell the travelling method used by Settler or the Robot, and also determine the amount of distance covered by them and their destination.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+destination:int** This variable tells the destination (neighbouring asteroid)where Settler/Robot is travelling to.
  - **+possibleTravelDistance:float** – This variable stores the distance by which the Setter/Robot will travel (change of position)

- **Methods**
  - **+Bool NormalTravel()**: It is the slower kind of traveling. If settler or robot travel using this, they take lot of time
  - **+void Teleportation()**: It is the special travelling method using teleportation gates

4.3.16 Timer

- **Responsibility**

It tells about the required time to perform different operations like drilling, mining, travelling, reviving the settler and also the duration of the sunstorm.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+healthReviveTime:int -** This variable stores the amount of time it takes to revive either Settler or Robot
  - **+sunStormTime:int -** This variable stores the amount of time for which the SunStorm hits
  - **+travelTime:int -** This variable stores the time taken for Settler/Robot to travel from one place to another (change of location)
  - **+drillTime:int -** This variable stores the time taken for drilling the asteroid core
  - **+mineTime:int -** This variable stores the time taken to mine the asteroid core

- **Methods**
  - **-**

4.3.17 Inventory

- **Responsibility**

It determines the type and total amount of resources a settler has mined and also the total carrying capacity of the settler.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+currentLoad:ArrayList<Long> -** This the variable stores the current amount of resources(current capacity) that has been mined by the Settler
  - **+totalLoad:ArrayList<Long> -** This variable stores the total amount of mined resources(total capacity) a Settler can carry

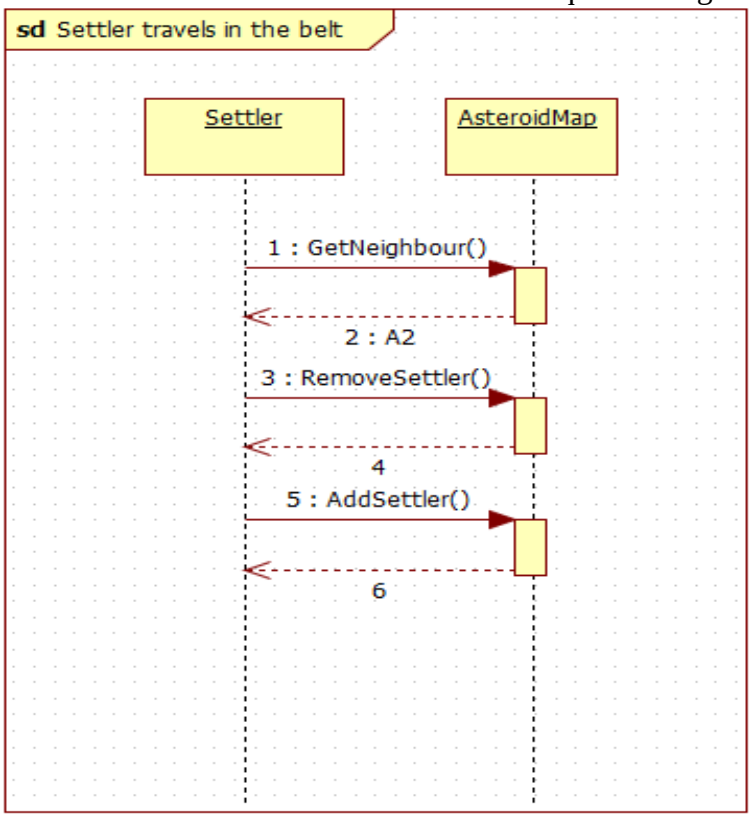
- **Methods**
  - **+Long GetCurrentLoad():** Gives the total amount of the resources mined by the settler

4.4 Sequence diagrams

Settler

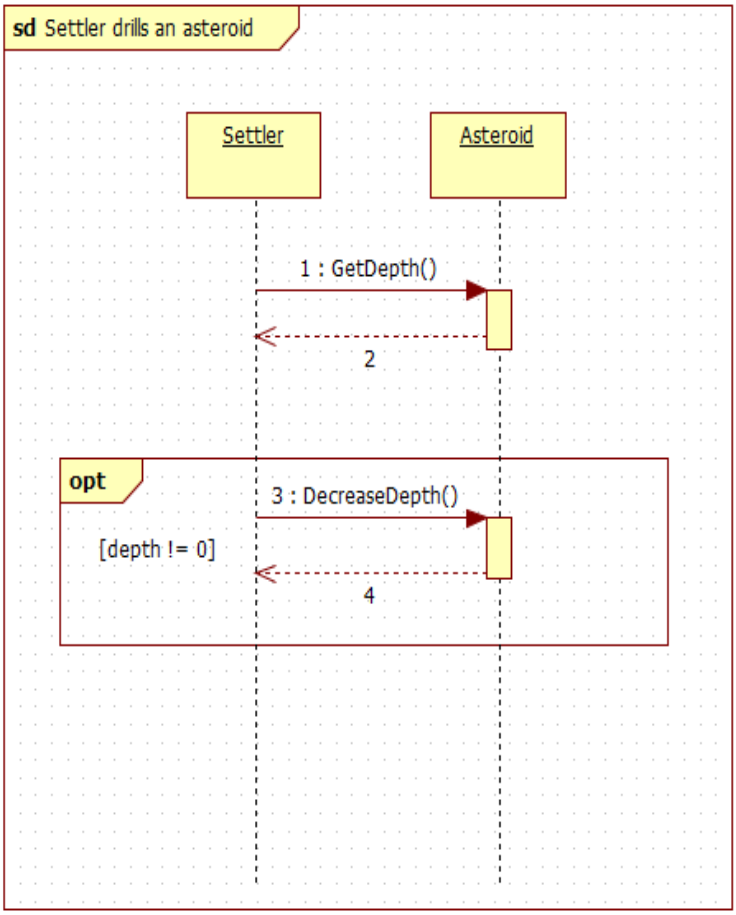
Settler travels in the belt

Figure 1 - Settler travels in the Asteroid Belt sequence diagram



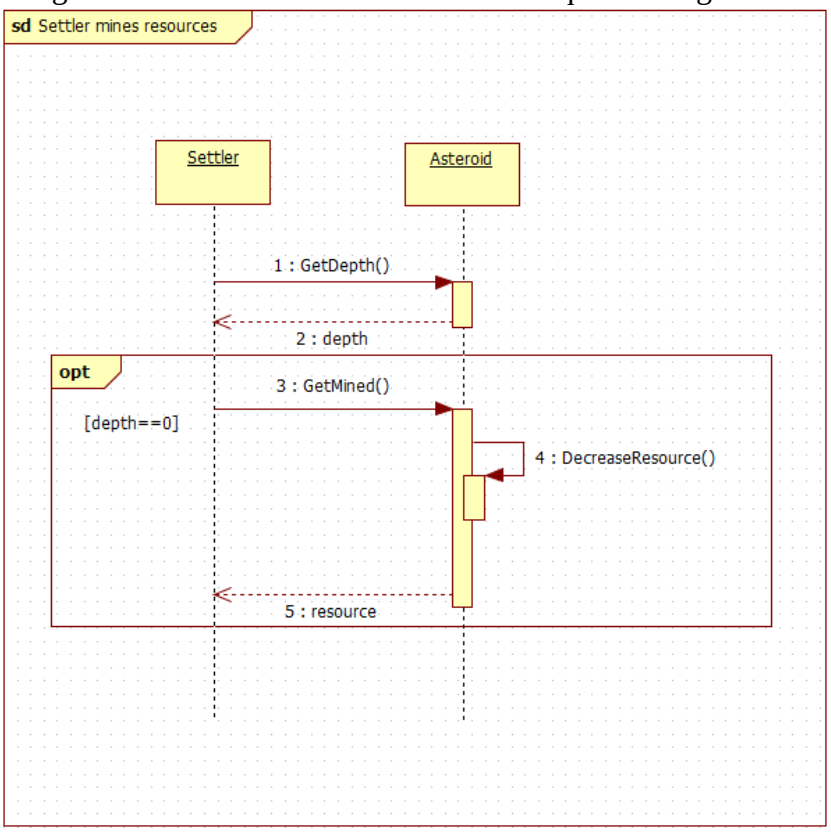
Settler drills an asteroid

Figure 2 - Settler travels in the Asteroid Belt sequence diagram



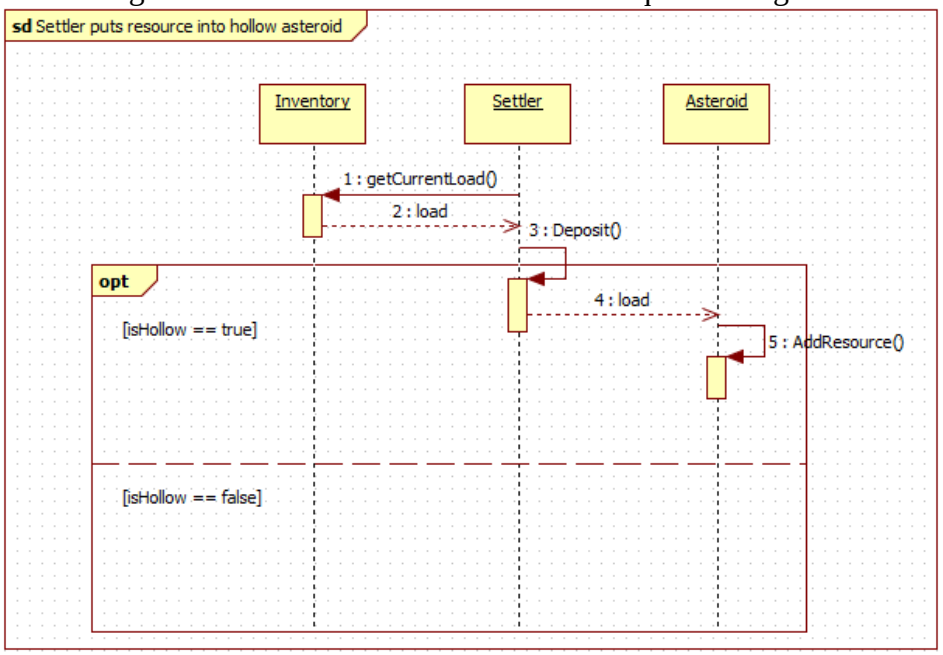
Settler mines some resources

Figure 3- Settler mines some resources sequence diagram



Settler unloads resources into a hollow asteroid

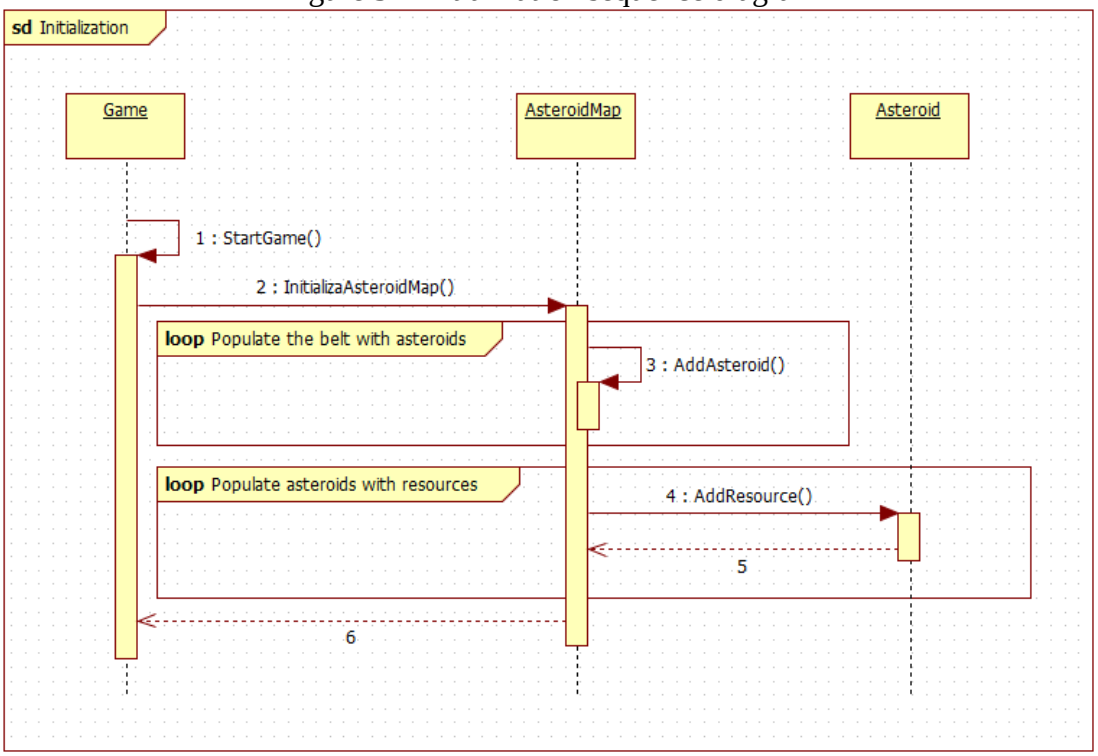
Figure 4- Settler mines some resources sequence diagram



Initialization

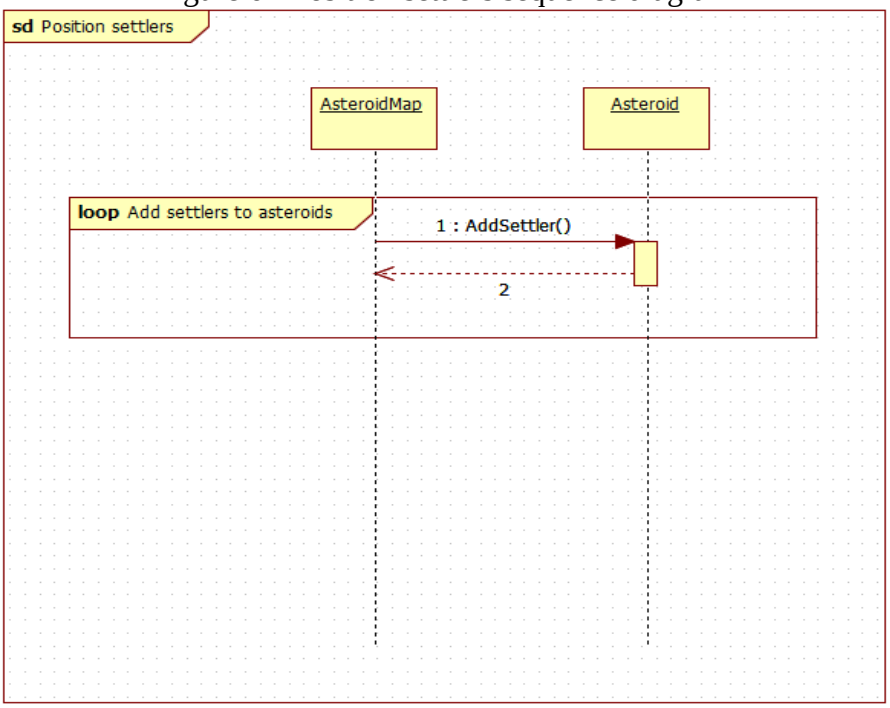
Asteroid belt is built, and resources are allocated to asteroids

Figure 5 - Initialization sequence diagram



Settlers are positioned on asteroids

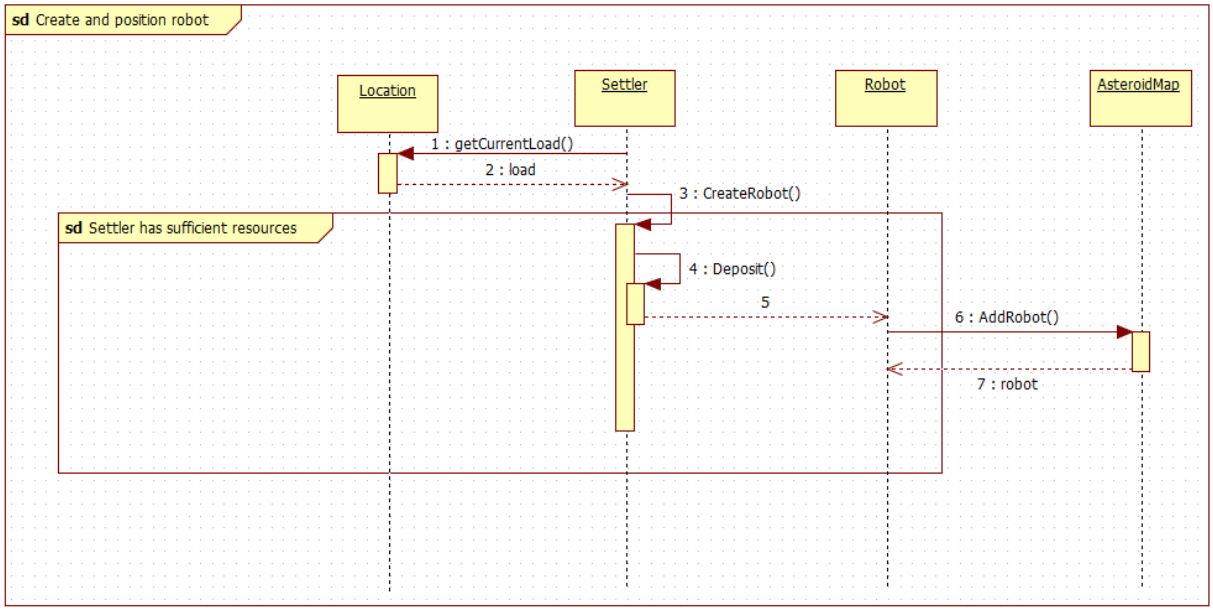
Figure 6 - Position settlers sequence diagram



Robot

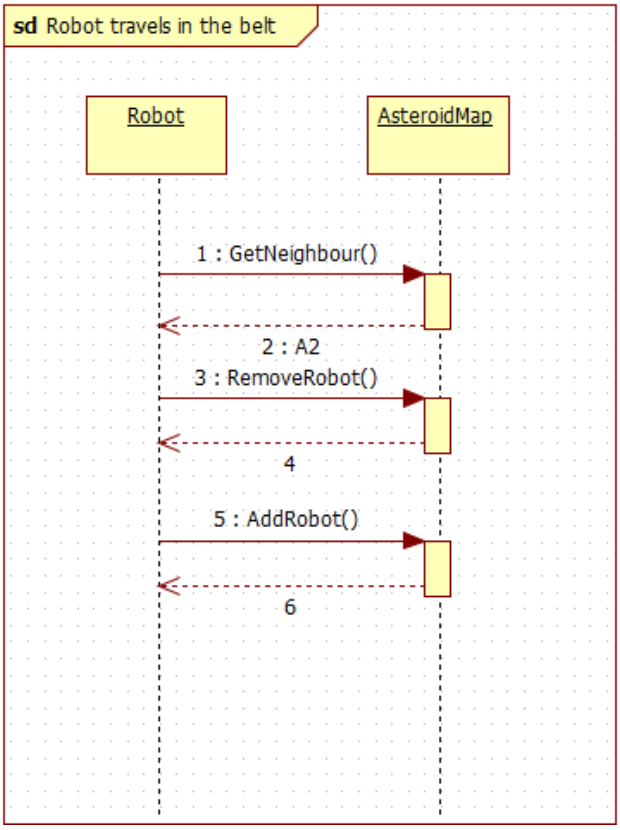
Robot is created

Figure 8 - Robot is created sequence diagram



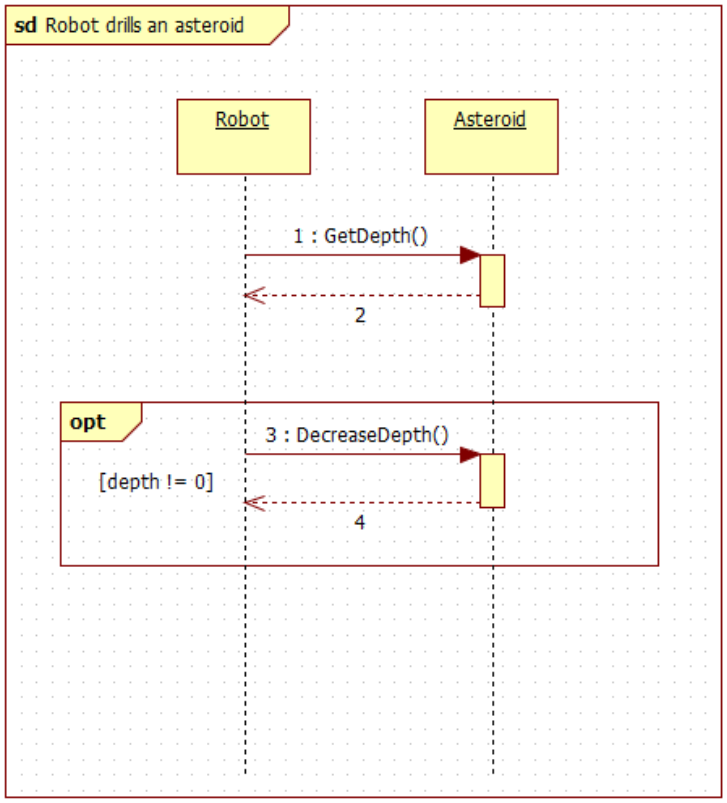
Robot travels in the belt

Figure 9 - Robot travels in the Asteroid Belt sequence diagram



Robot drills an asteroid

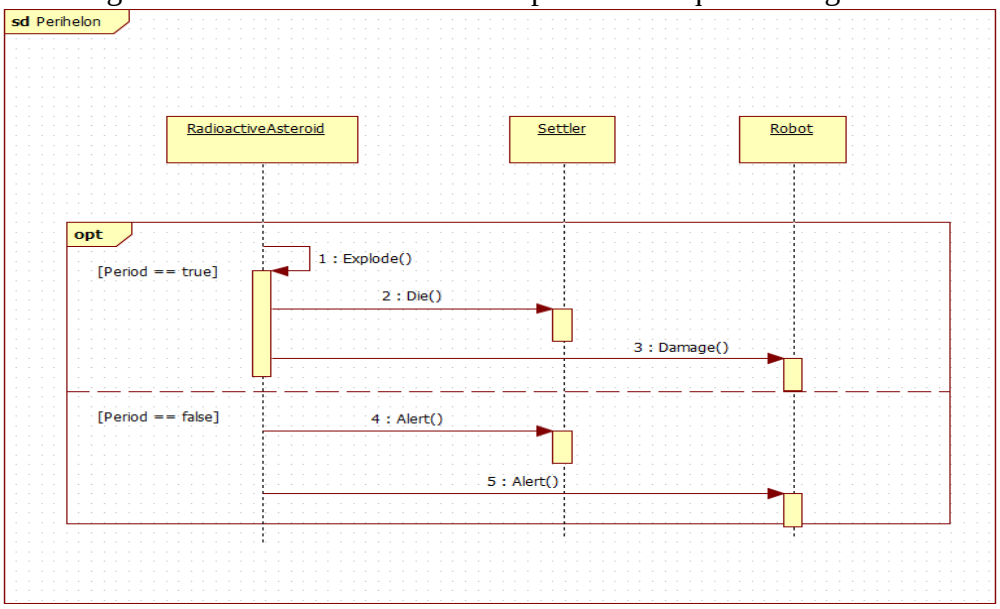
Figure 10 - Robot drills an asteroid sequence diagram



AsteroidMap

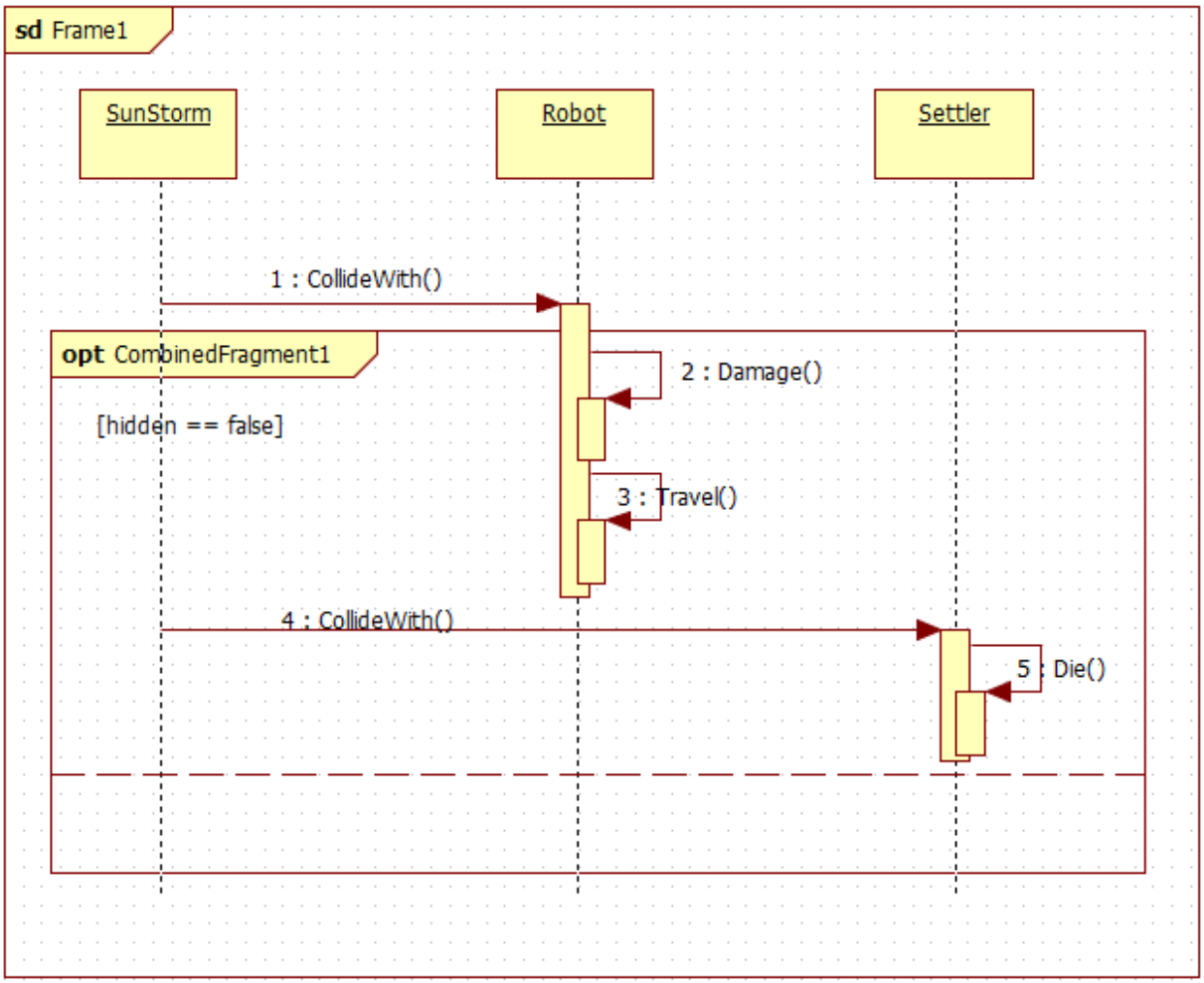
Perihelon

Figure 11 - Radioactive asteroid at perihelon sequence diagram



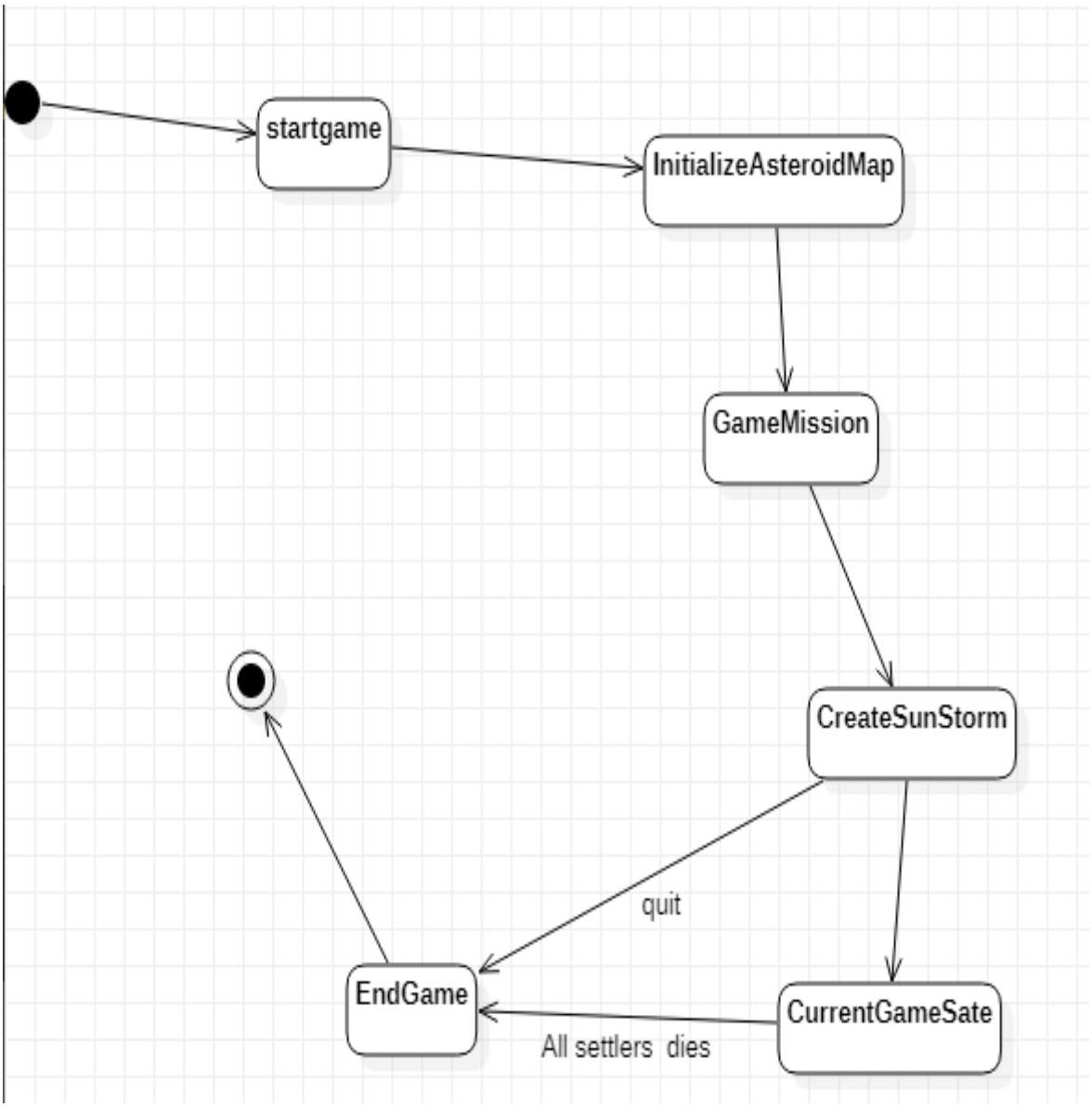
Sun storm

Figure 12 - Sun storm sequence diagram

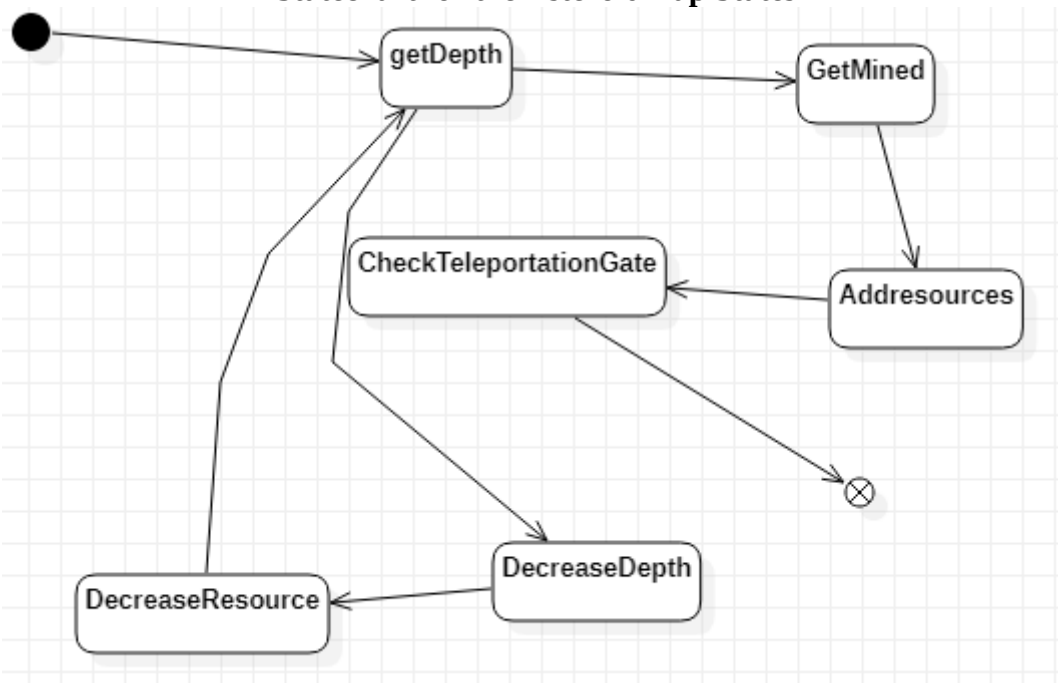


4.5 State-charts

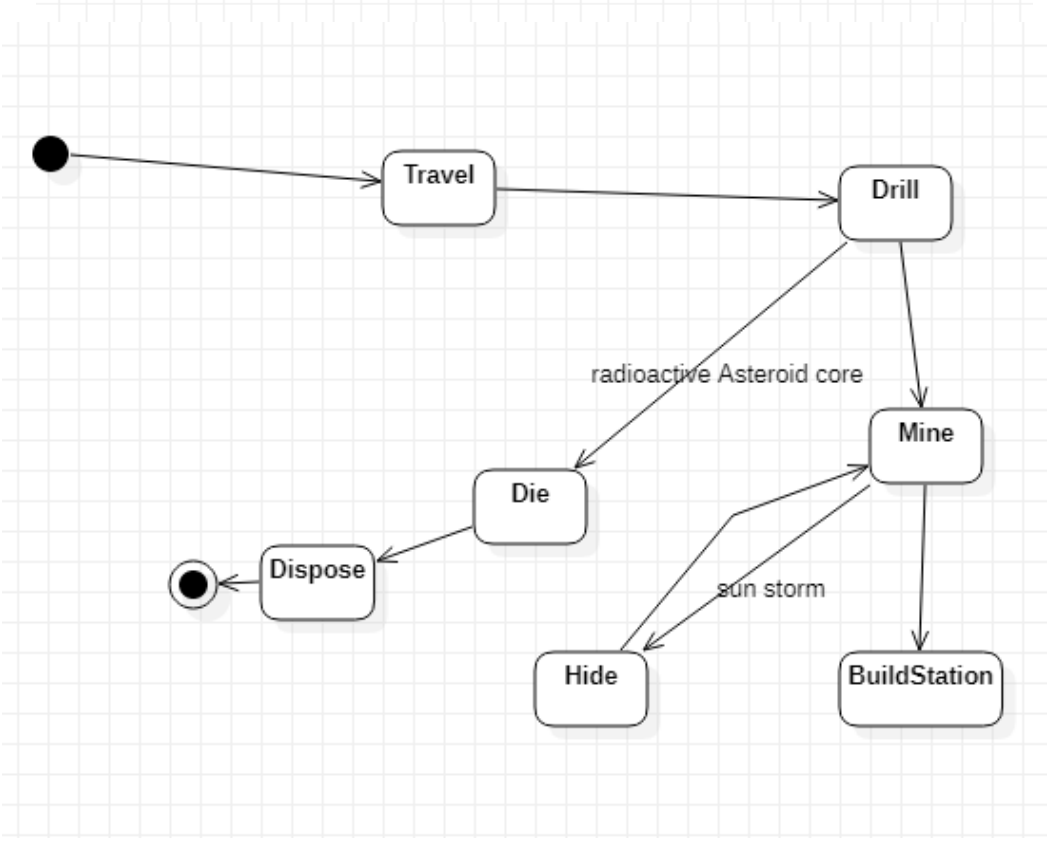
Statechart for the Game States



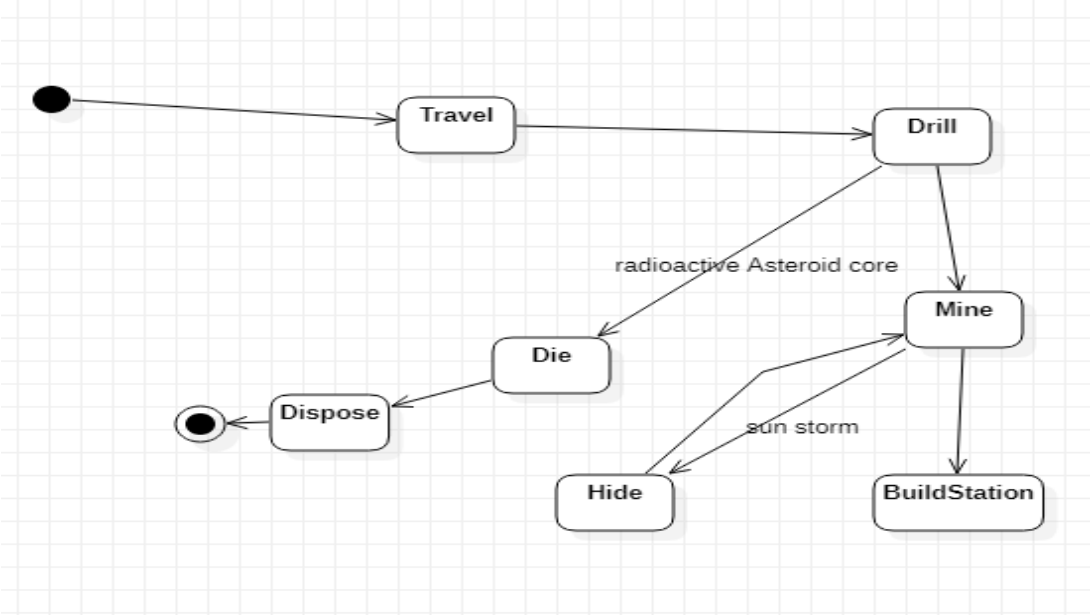
Statechart for the AsteroidMap States



Statechart for Asteroid States



Statechart for *Settler* States



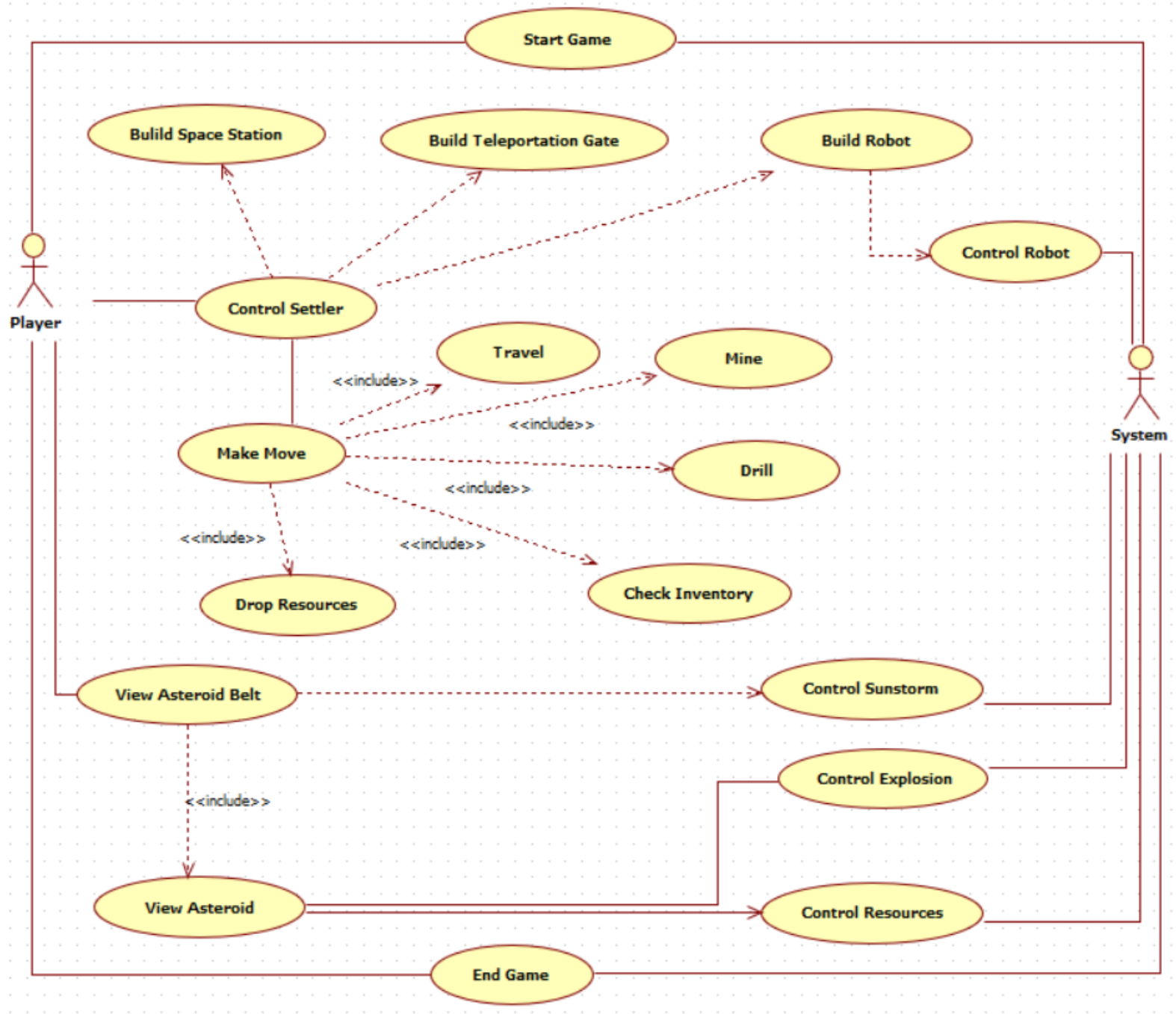
4.6 Protocol

Start (date & time)	**Duration (hours)**	**Performer(s) name**	**Activity description**
03.03.2021. 18:00	2	Team	Meeting: Decisions were made regarding this document and responsibilities were divided as follows: Wali Ullah 3.1 Ibrar Shakoor 3.2+Documentation Ashiq and Idrees 4.3 Orgil 3.4 Vincent 3.5
04.02.2021. 12:00	1+1.5	Wali Ullah	Brainstorming + Completed artifact 4.1
04.02.2021. 3:00	2+1	Ibrar Shakoor	Brainstorming + Completed artifact 4.2
05.03.2021. 15:00	1 + 1.5	Asad Idrees	Brainstorming + Completed artifact 4.3
07.03.2021. 15:00	1 + 1.5	Ashiq	Brainstorming + Completed artifact 4.3
07.02.2021. 18:25	1	Orgil Luis Macha	Completed artifact 4.4
07.02.2021 19:30	1+1.5	Vincent	Completed artifact 4.5+Documentation

Planning the skeleton

5.1 Real use-cases of the skeleton model

5.1.1 Use-case diagram



5.1.2 Use-case descriptions

Use-case name	Start Game
Short textual description	Player can choose to start the game after selecting the number of settlers. Once the game is started, the system generates asteroids in the asteroid belt.
Actors	Player
Dialog, scenario	Player starts the game, the map is initialized with all the elements on it (asteroids, settlers, resources etc). After that, the game is waiting for user input.

Use-case name	End Game
Short textual description	The system can end the game if all the settlers die, or if the space station is built. The player can also manually quit the game at any given time.
Actors	Player, System
Dialog, scenario	1. Player chooses the action to end the game manually. 2. The system realizes that either all the settlers are dead, or the space station is built successfully and ends the game.

Use-case name	Control Settler
Short textual description	Player controls the settler and can choose various operations to be performed by the settler like travel, drill, mine etc.
Actors	Player
Dialog, scenario	Player chooses the action that is to be performed by the settler, the settler performs it, and makes the necessary changes in the game.

Use-case name	Build Space Station
Short textual description	Player builds the space station using necessary resources.
Actors	Player
Dialog, scenario	Player can check the inventory for resources necessary to build a space station. 1. If the resources are present, the space station is built successfully. 2. In the absence of required resources, the system prompts the player to collect the missing resources.

Use-case name	Build Teleportation Gate
Short textual description	Player builds the teleportation gates using necessary resources. A pair of gates is necessary to teleport. The player knows where the pair of the gate is.
Actors	Player
Dialog, scenario	Player chooses the action to build a teleportation gate. 1. If the resources are present, the teleportation gate is built successfully. 2. In the absence of required resources, the system prompts the player to collect the missing resources. 3. If the player tries to build more than 2 gates using the same settler, the system does not allow it, and displays a warning.

Use-case name	Build Robot
Short textual description	Player controls a settler to build a robot using required resources.
Actors	Player
Dialog, scenario	Player chooses to build a robot. 1. If the resources are present, the robot is built successfully. 2. In the absence of required resources, the system prompts the player to collect the missing resources to build a robot.

Use-case name	Control Robot
Short textual description	The system controls the robot. The robot can assist the settler by performing various actions.
Actors	System
Dialog, scenario	1. The system is able to control the robot for drilling the asteroids, or hiding in a hollow asteroid to avoid being damaged by Sunstorms. 2. The robot is also unaffected by radioactive explosions. 3. The robot can travel using normal travel or teleportation.

Use-case name	Make Move
Short textual description	Player controls the settler to perform one move at a time.
Actors	Player
Dialog, scenario	When a player chooses this action, the game will show a menu of different operations that a settler can perform on an asteroid. The menu consists of following actions: 1. Travel 2. Mine 3. Drill 4. Check Inventory 5. Drop Resources

Use-case name	Travel
Short textual description	Player controls the settler to travel to neighboring asteroids. The system controls the robot to travel.
Actors	Player, System
Dialog, scenario	Player is presented by 2 choices to travel: 1. Normal Travel 2. Teleportation Before travelling, the player knows the location of the destination asteroid that it is travelling to. Before teleportation, the player has to check for the gate pair. The system can control the robot to travel in the asteroid belt.

Use-case name	Mine
Short textual description	Player can control the settler to mine the asteroid for resources/minerals.
Actors	Player
Dialog, scenario	Player chooses to mine, and the settler mines one unit of the resource in one move, and adds it to the inventory. The game updates the inventory status and modifies the quantity of resources. Resources can not be mined from a hollow asteroid.

Use-case name	Drill
Short textual description	Player controls the settler to drill an asteroid. System controls the robot to drill an asteroid.
Actors	Player, System
Dialog, scenario	Player chooses the action and the settler drills the asteroid one unit at a time. The game keeps check of the depth of the mantle as it's being drilled by a settler or the robot. When an asteroid is drilled to its core, then it is marked as a hollow asteroid. 1. The asteroid is successfully drilled to it's core if radioactivity is at aphelion, or if the asteroid is not radioactive at all. 2. The drilling is unsuccessful if the asteroid radioactivity is at perihelion.

Use-case name	Drop Resources
Short textual description	Player controls the settler to drop the resources in its inventory in a hollow asteroid.
Actors	Player
Dialog, scenario	Player chooses the action and the settler checks if the asteroid is empty. The game updates the inventory status only if the action is successful. 1. The resources are successfully dropped if the asteroid is hollow.. 2. The resources can not be dropped if the asteroid is not hollow.

Use-case name	Check Inventory
Short textual description	Player controls the settler to check the inventory of the resources.
Actors	Player
Dialog, scenario	Player chooses the action to view the amount and the types of resources present in the inventory of a settler. The inventory also knows how many gates the settler is carrying.

Use-case name	View Asteroid Belt
Short textual description	Player views the asteroid belt.
Actors	Player
Dialog, scenario	Player chooses the action to view a map of the asteroid belt, and the game generates the current status of the asteroid belt. This enables the player to view the neighboring asteroids and their locations.

Use-case name	View Asteroid
Short textual description	Player views the current state of an asteroid.
Actors	Player
Dialog, scenario	Player chooses the action and the game presents the details of the asteroid like the resources present, the depth of the mantle, the teleportation gates constructed nearby, the radioactive state etc.

Use-case name	Control SunStorm
Short textual description	The system controls the sun storms in the game.
Actors	System
Dialog, scenario	The system decides when to create a sunstorm in the asteroid belt. It also controls the duration, the damage, and the range of the sunstorm. 1. The sunstorm successfully damages the robot/settler if they are exposed. 2. The sunstorm does not damage the robot/settler in case they are hiding in a hollow asteroid.

Use-case name	Control Explosion
Short textual description	The system controls the explosions of the radioactive asteroids in the game.
Actors	System
Dialog, scenario	The system determines the radioactive state of an asteroid. 1. The explosion damages the settler if they are exposed to the explosion. The settler dies and loses all the resources it is carrying. In this case, the teleportation gates built by that settler are marked inactive and can not be used. 2. The robots can survive the radioactive explosion, but they are immediately transported to the nearest asteroid by the game system.

Use-case name	Control Resources
Short textual description	The system controls the resources in the game.
Actors	System
Dialog, scenario	The system generates the resources in the asteroid belt when the game is initialized. The system also keeps track of the changes when the settlers mine or drop the resources.

5.2 Plans of the skeleton's UI, dialogs

When starting the program the system get initialized by the Game class to run the game loop: "Please, press any key to start the game!"

User input: $AnyKey$
System output: StartGame()
The game has started!

Test cases:

Case 0 : Setting the number of Players

When the game has started the system will ask the players how many players are playing. Therefore the players have to provide the number of people playing.

1. **System Output**: "How many players are playing?"
   User Input: $Number$
   System Output: NumberOfPlayers()
   $Number$ players has been created!

Case 1 : Initialization of Asteroid

1. **System Output**: InitializeAsteroidMap()
   Let's set up the Asteroid and it's resources! Please input two numbers to set up a range to randomly create the number of Asteroid.
   User Input: $lowNumber$ $highNumber$.

2. **System Output**: AddAsteroid()
   Asteroid {i} has been added now assigning resources to Asteroid {i}.

3. **System Output**: AddResources()
   Uranium
   Randomly assigning 'Uranium' resources to Asteroid {i}
   WaterIce
   Randomly assigning 'WaterIce' resource to Asteroid. {i}
   Iron
   Randomly assigning 'Iron' resources to Asteroid {i}
   Carbon
   Randomly assigning 'Carbon' resources to Asteroid.

Case 2 : Position of Settlers on Asteroids

1. **System Output:** AddSettler()
   Please add the number of Settlers you want to add into the game.
   User Input: $Number$
   System Output: "Adding Settlers to the Asteroid from AsteroidMap"

Case 3 : Settler travels

1. **System Output:** Press 'F' to find the neighbouring asteroids.
   User Input: F_Key
   System Output: GetNeighbour()
   Press 'E' to move onto one of the Asteroids.
   User Input: E_Key

2. **System Output:** RemoveSettler()

3. **System Output:** AddSettler()
   Settler moved to the new Asteroid

Case 4 : Settler drills

1. **System Output:** Press 'G' to check the Depth of the Asteroid.
   User Input : G_Key
   System Output: GetDepth()
   Checking the Depth of the current Asteroid!

2. <If the Depth is not equals to zero>
   System Output: Press 'D' to drill the Asteroid.
   User Input : D_Key
   System Output: DecreaseDepth()
   Decreasing the depth of the Asteroid to expose the core!

Case 5: Settler Mines

1. **System Output:** Press 'G' to check the Depth of the Asteroid.
   User Input : G_Key
   System Output: GetDepth()
   Checking the Depth of the current Asteroid!

2. <If the Depth is equals to zero>
   System Output: Press 'M' to mine the Asteroid.
   User Input : M_Key
   System Output: GetMined()
   Mining the Asteroid Core!

3. <If GetMined() is invoked>
   System Output : DecreaseResource()
   Adding the resources to the Settlers Inventory!

Case 6: Settler unload resources in the hollow Asteroid

1. **System Output:** Press 'L' to check the inventory of the settler.
   User Input : L_Key
   System Output: GetCurrentLoad()
   Checking the Inventory of the Settler!

2. <If the Asteroid IsHollow true>
   User Input : P_Key
   System Output: Deposit()
   Taking out the resources from the Inventory.

3. <If Deposit() is Invoked>
   System Output: AddResource()
   Resources taken out from the Inventory will be added to the Asteroid.

Case 7: Robot Creation

1. **System Output:** Press 'L' to check the inventory of the settler.
   User Input : L_Key
   System Output: GetCurrentLoad()
   Checking the Inventory of the Settler!

2. <If Inventory is sufficient>
   System Output : Deposit()
   Some of the Resources have been dropped/deposited by the settler on the asteroid.

3. <If Deposit() is Invoked >
   System Output : CreateRobot()
   A Robot has been created from the dropped resources to assist the setter.

4. <If CreateRobot() is invoked >
   System Output : AddRobot()
   Adding robots to the Asteroid Map.

Case 8: Robot travels

1. **System Output:** Press 'F' to find the neighbouring asteroids.
   User Input: F_Key
   System Output: GetNeighbour()
   Press 'E' to move onto one of the Asteroids.
   User Input: E_Key

2. **System Output:** RemoveRobot()

3. **System Output:** AddRobot()
   Robot moved to the new Asteroid

Case 9: Robot drills an Asteroid

1. **System Output:** Press 'G' to check the Depth of the Asteroid.
   User Input : G_Key
   System Output: GetDepth()
   Checking the Depth of the current Asteroid!

2. <If the Depth is not equals to zero>
   System Output: Press 'D' to drill the Asteroid.
   User Input : D_Key
   System Output: DecreaseDepth()
   Decreasing the depth of the Asteroid to expose the core!

Case 10: Gate creation

1. **System Output:** Press 'L' to check the inventory of the settler.
   User Input : L_Key
   System Output: GetCurrentLoad()
   Checking the Inventory of the Settler!

2. <If sufficientResources is true>
   System Output: builtGate()
   Building the teleportation gate.

3. <If builtGate() is invoked>
   System Output: Deposit(resource)
   Drop the resources for building gates.
   System Output: AddGates()
   Adding gates to the inventory.

Case 11: Gate deployment

1. <If the gate has been built>
   System Output: "Press 'X' to deploy the gate"
   User Input: X_Key
   System Out: DeployGate()
   Deploying the gate in the vicinity of the asteroid.

2. **System Output:** Press the 'R' to remove the gate!
   User Input: R_Key
   System Output: removeGate()
   Removing the gate from the asteroid.

3. <If gates=0 in the inventory(no gate in the inventory)>
   System Output: activateGates()
   Activating the deployed gates.

Case 12: Gate teleportation

1. <If teleportation gates have deployed >
   System Output: "Press 'T' to teleport"
   User Input: T_Key
   System Output: Teleportation()
   Now you can teleport.

2. <If gateActive is true>
   System Output: getNeighbour()
   Getting the destination asteroid to move to.

3. **System Output:** Travel(a: Asteroid)
   Moving to the next teleportation gate.

Case 13: Radioactive explosion for Settler and Robot

1. <If Period is true>
   System Output: Explode()
   Explosion has happened!

2. <If Explode(S: Settler) is invoked>
   System Output : Die()
   The Settler died!
   System Output: EndGame()
   Game is Over because Settler died!

3. <If Explode(R: Robot) is invoked>
   System Output :Damage()
   The Robot has suffered damage.

4. <If Period is false>
   System Output: Alert()
   Both Settler and Robot should be alerted that we're on Radioactive Asteroid.

Case 14: Sunstorm collides with Settler and Robot

1. <If CollideWith(R: Robot) is invoked and hidden is false>
   System Output: Damage()
   Robot took damage because of the SunStorm.
   System Output : Travel()
   Robot can still travel even after taking the damage!

2. <If CollideWith(S: Settler) is invoked and hidden is false>
   System Output: Die()
   Settler died due to the Sunstorm!
   System Output: EndGame()
   "Game Over: You lost the game!"

Case 15: Settlers build the space station

1. **System Output:** Press 'L' to check the inventory of the settler.
   User Input : L_Key
   System Output: GetCurrentLoad()
   Checking the Inventory of the Settler!

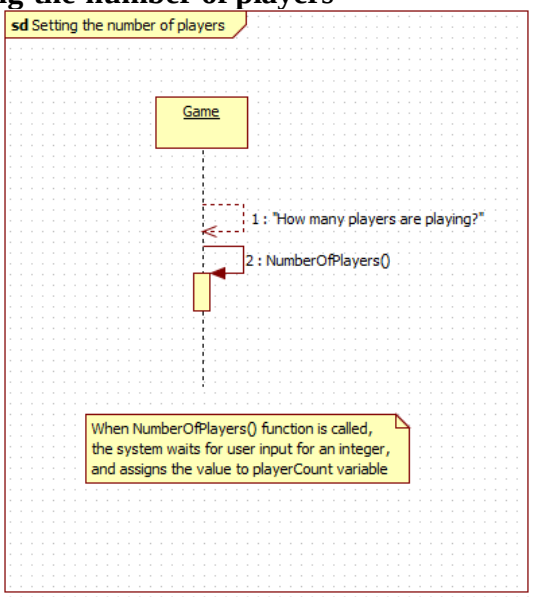
2. **System Output:** BuildStation()
   Building the Space Station.

3. <If BuildStation() is invoked and isHollow is true>
   System Output: AddResource(r: Resource)
   Adding resources to the asteroid for building the space station.

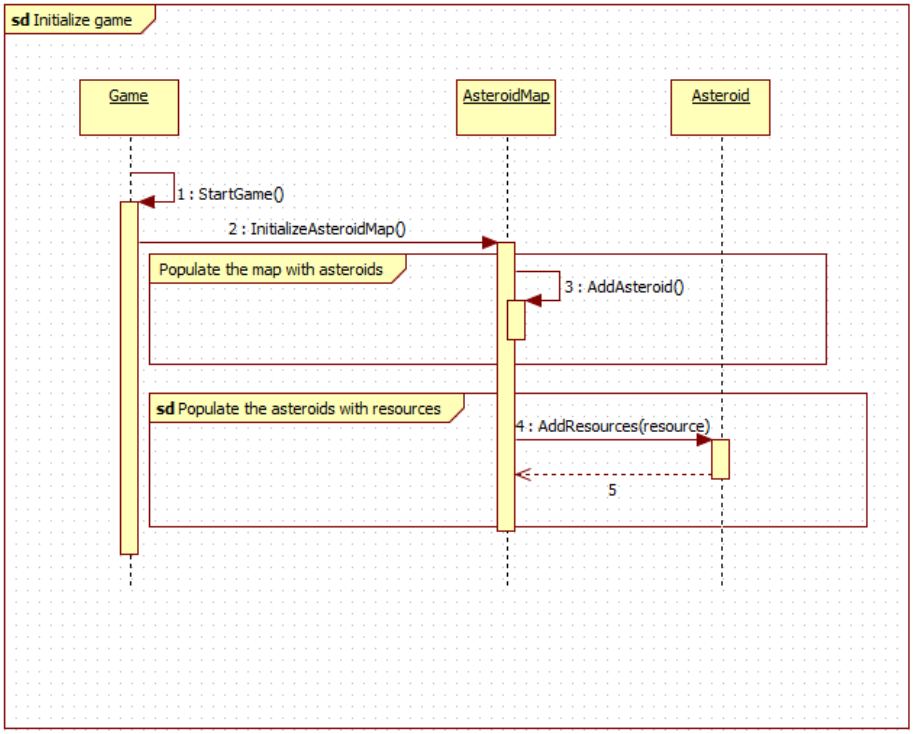
4. (If sufficientResources is true)
   System Output: EndGame()
   " Game Over: Congratulations you won the game!"

5.3 Detailed sequence diagrams for internal activities

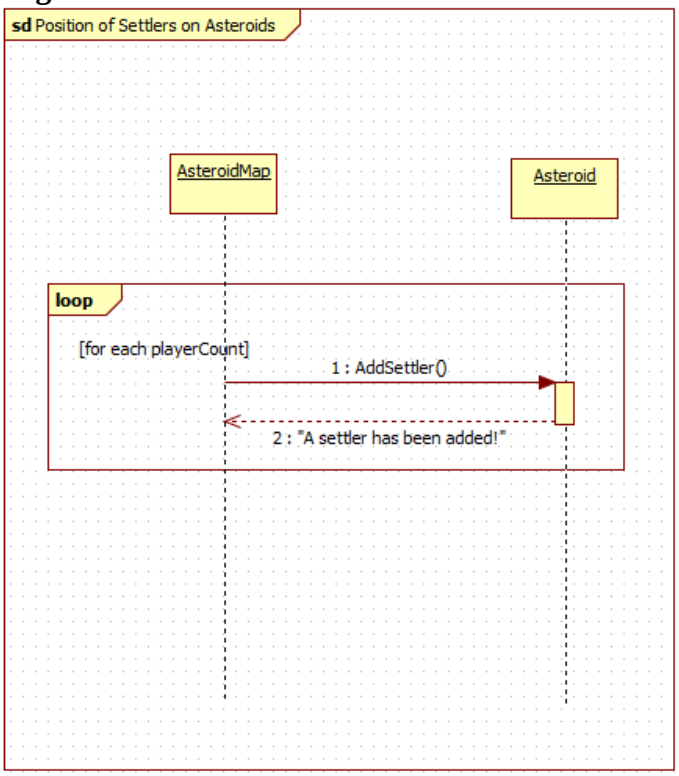
5.3.1 Initialization: Setting the number of players



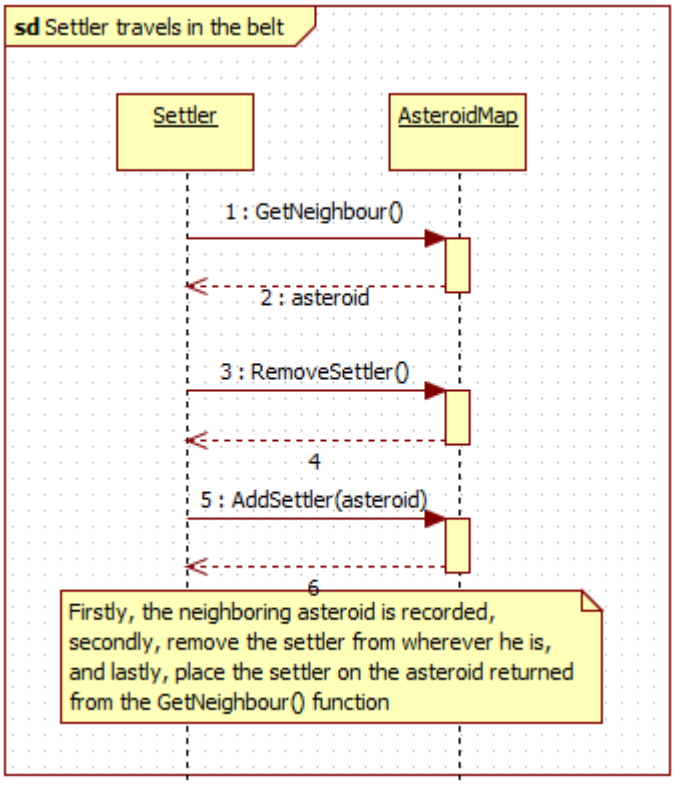
5.3.2 Asteroid belt is built, and resources are allocated to asteroids



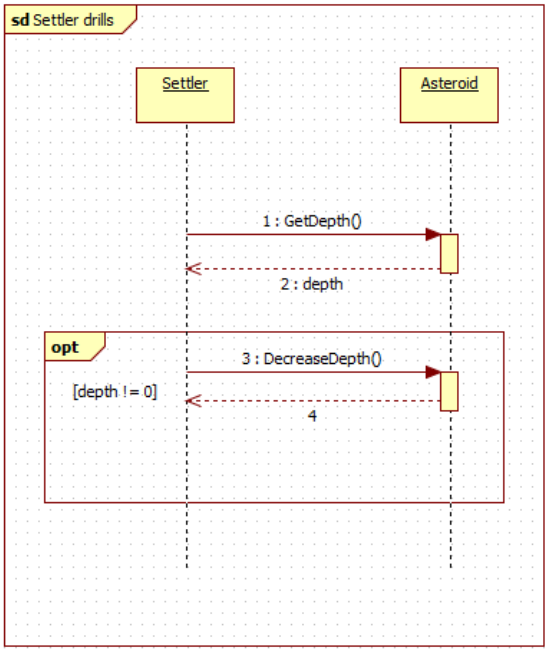
5.3.3 Positioning the settlers



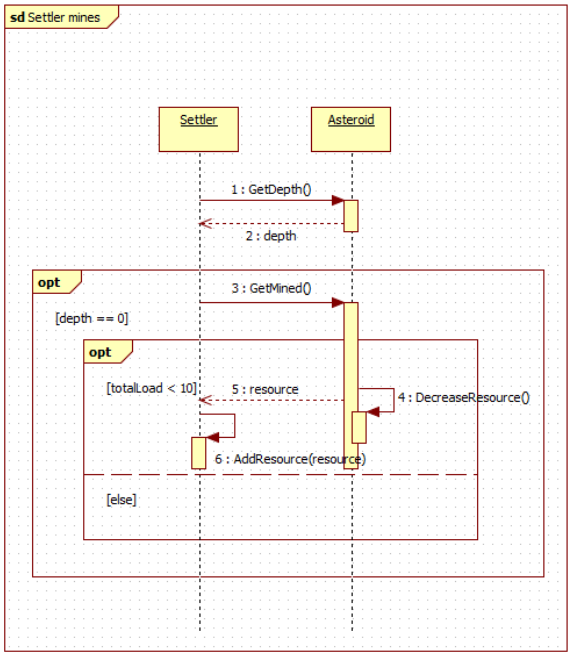
5.3.4 Settler travels



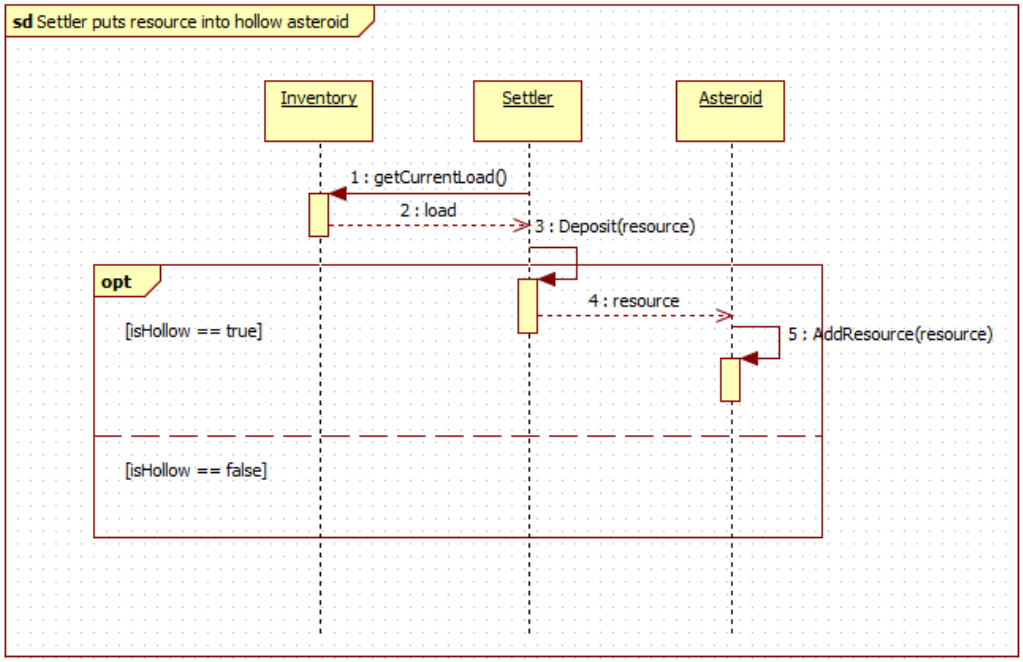
5.3.5 Settler drills



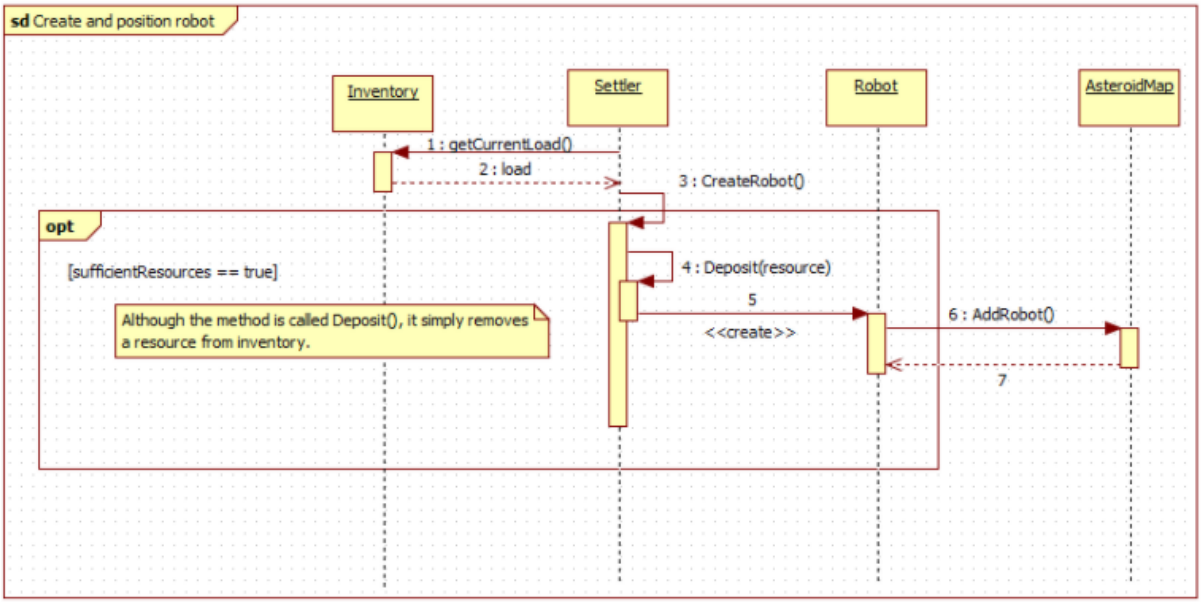
5.3.6 Settler mines



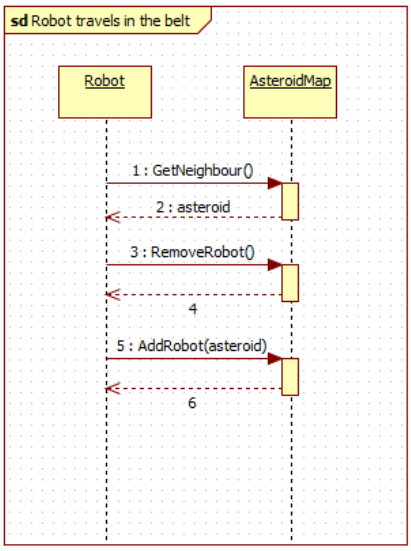
5.3.7 Settler unloads resources to hollow asteroid



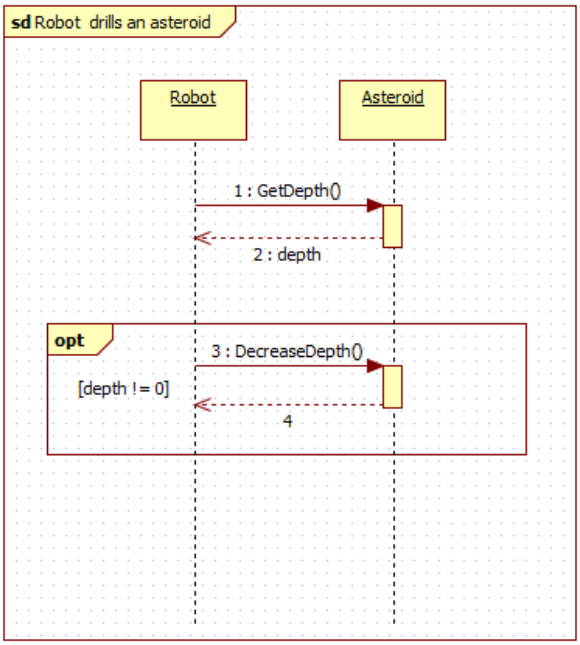
5.3.8 Robot creation



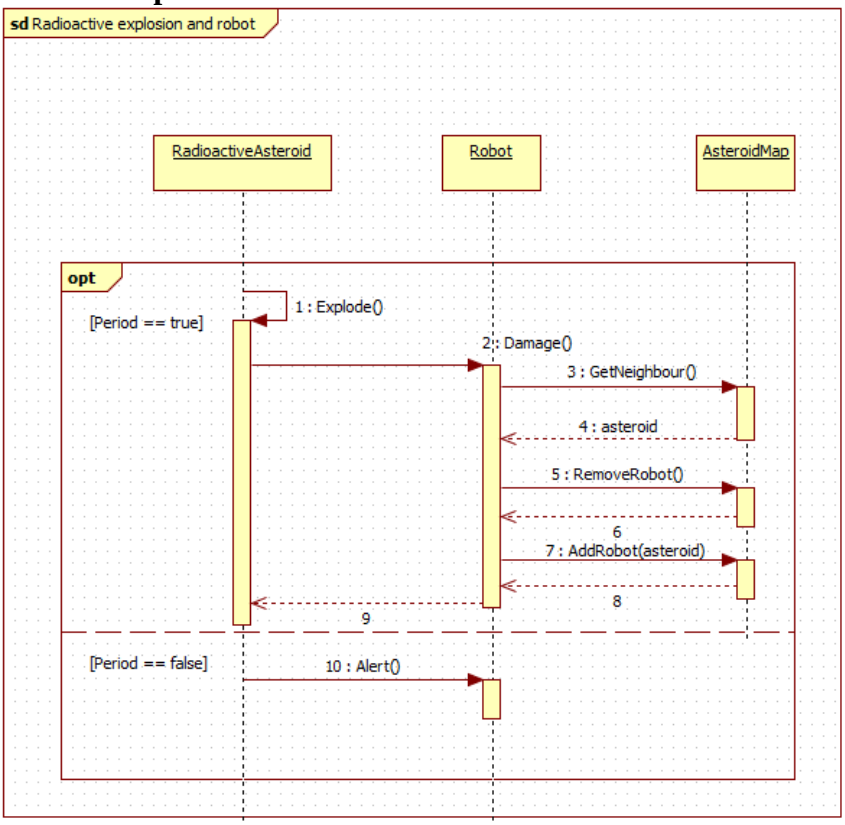
5.3.9 Robot travels



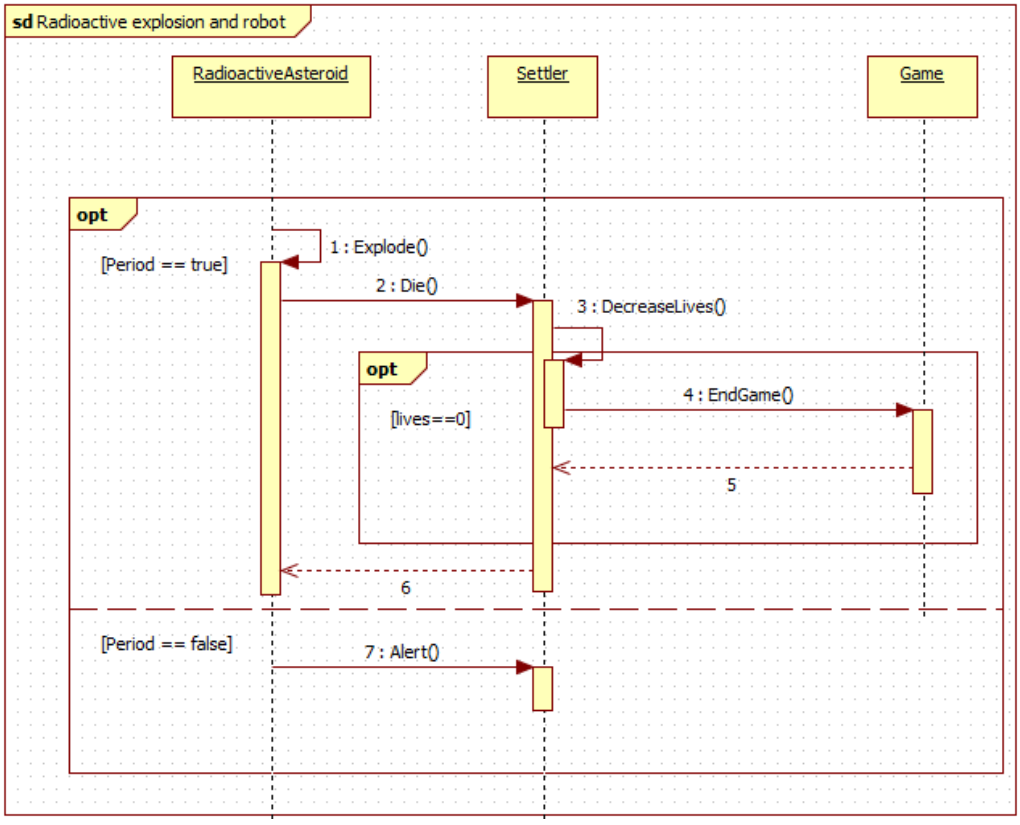
5.3.10 Robot drills



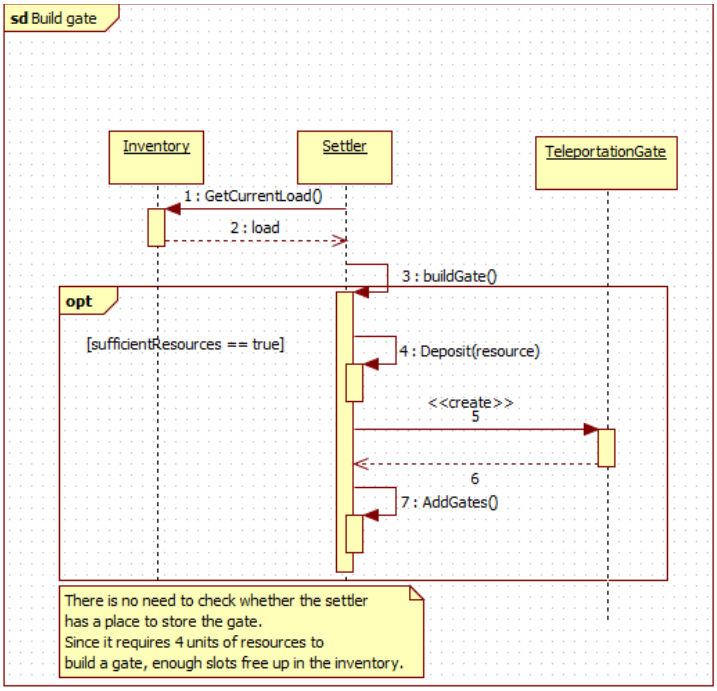
5.3.11 Radioactive explosion and robot



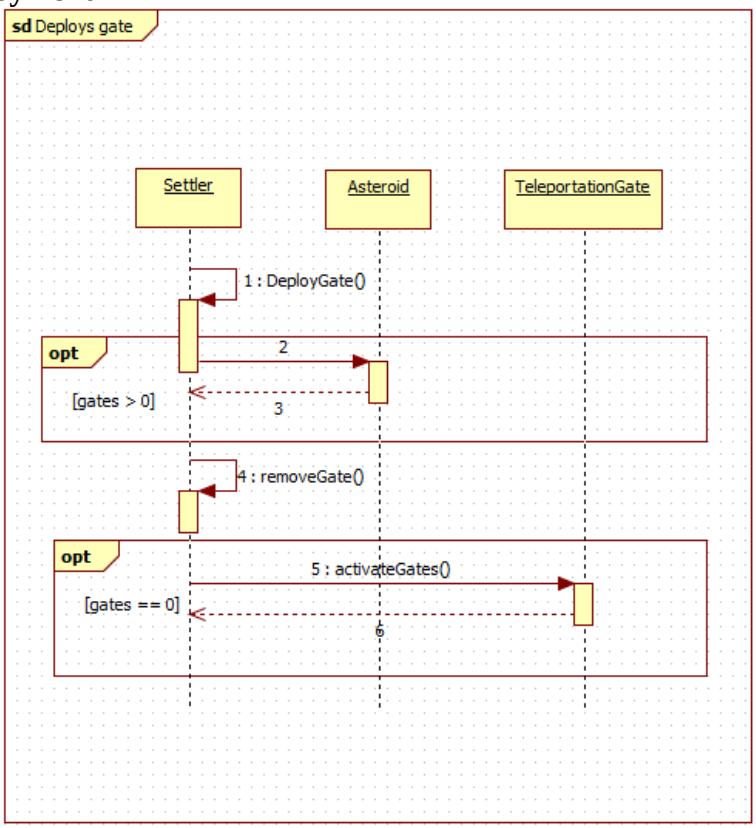
5.3.12 Radioactive explosion and settler



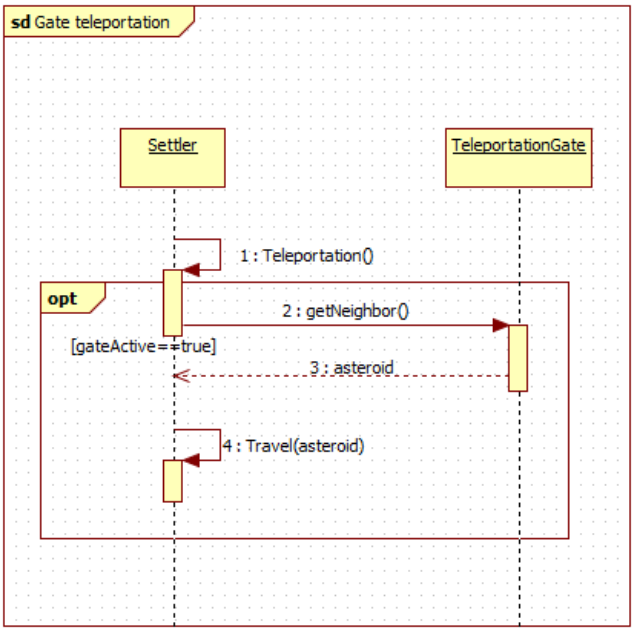
5.3.13 Gate creation



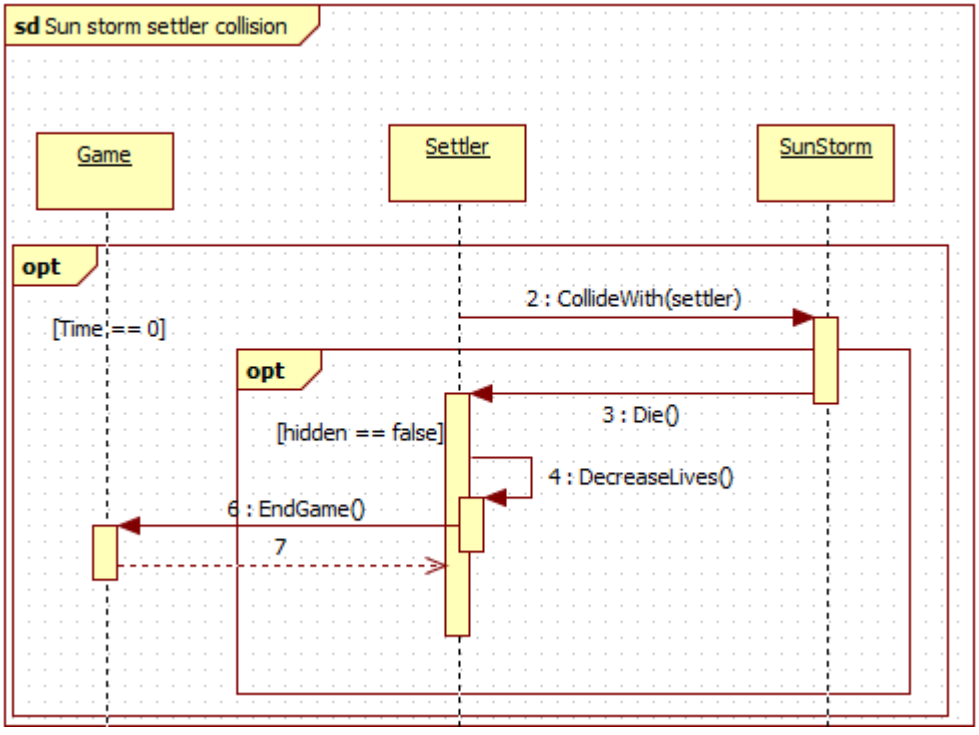
5.3.14 Gate deployment



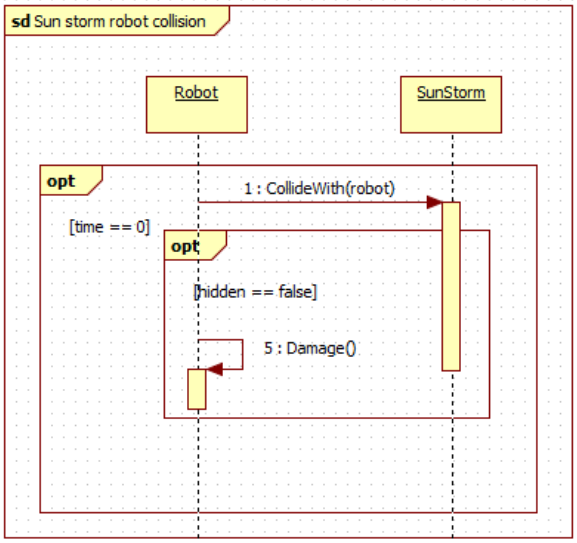
5.3.15 Gate teleportation



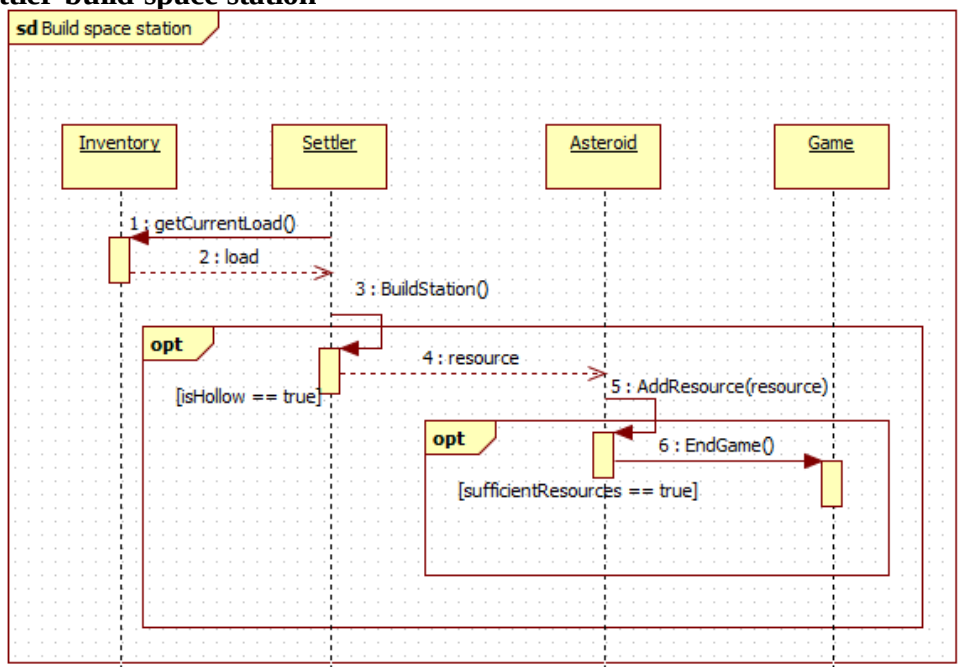
5.3.16 Sun storm collides with settler



5.3.17 Sun storm collides with robot

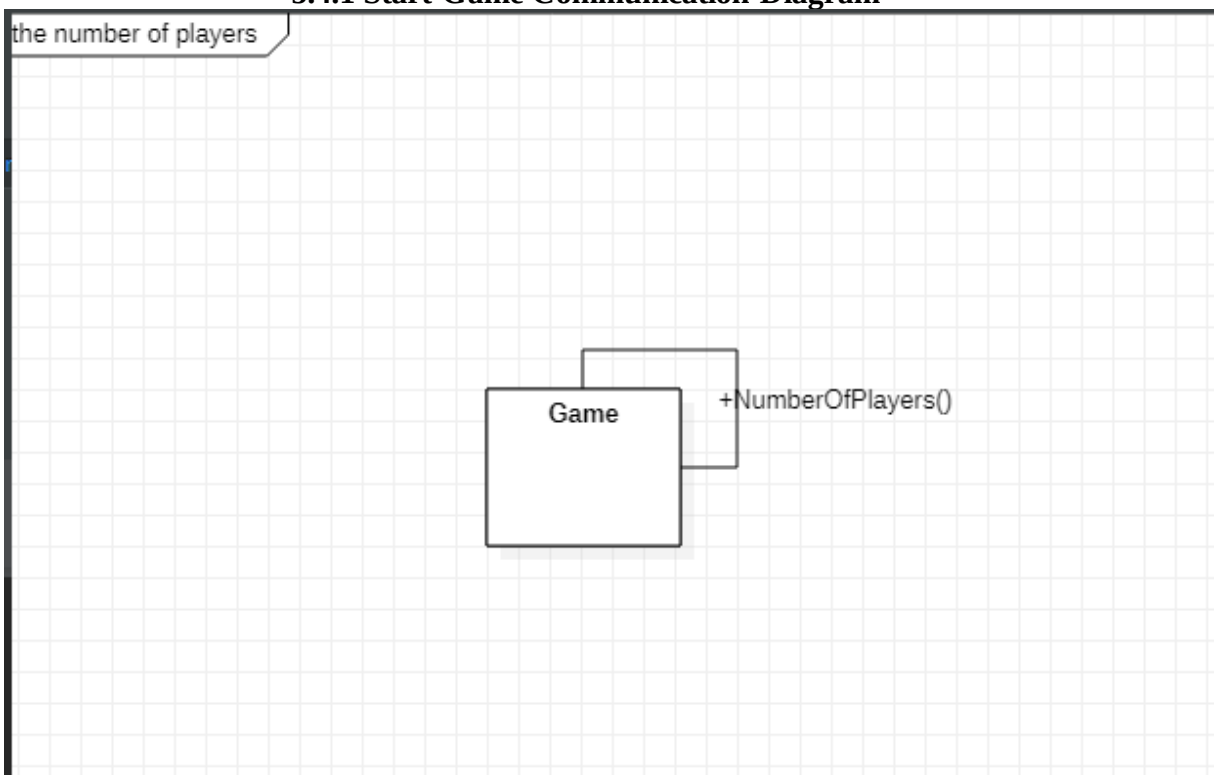


5.3.18 Settler build space station

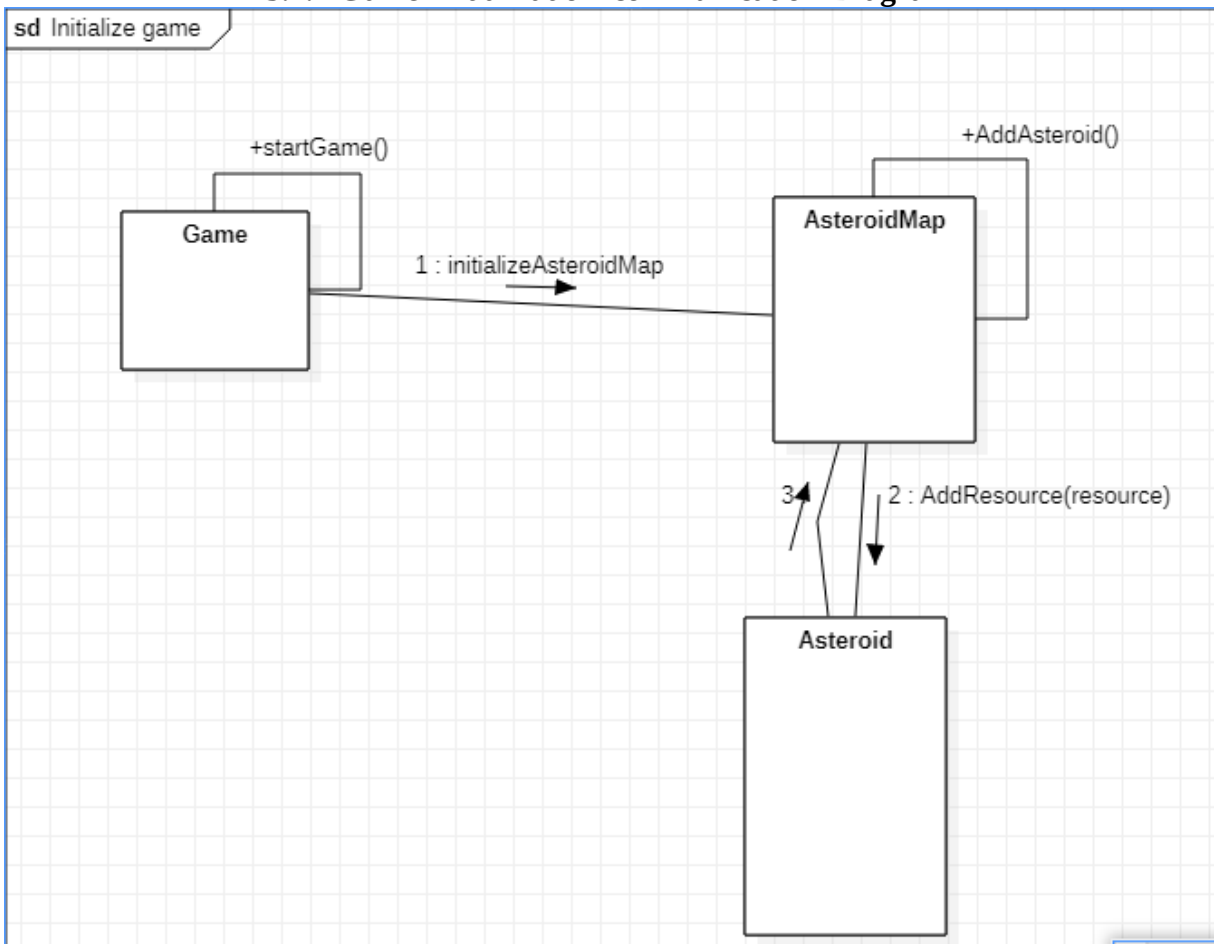


5.4 Communication diagrams

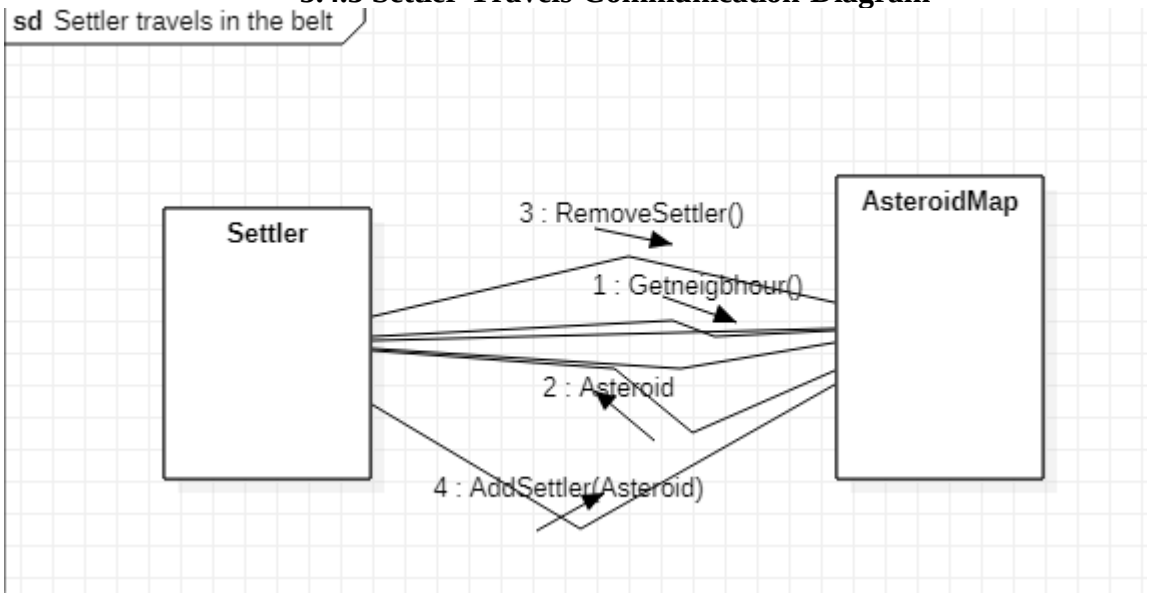
5.4.1 Start Game Communication Diagram



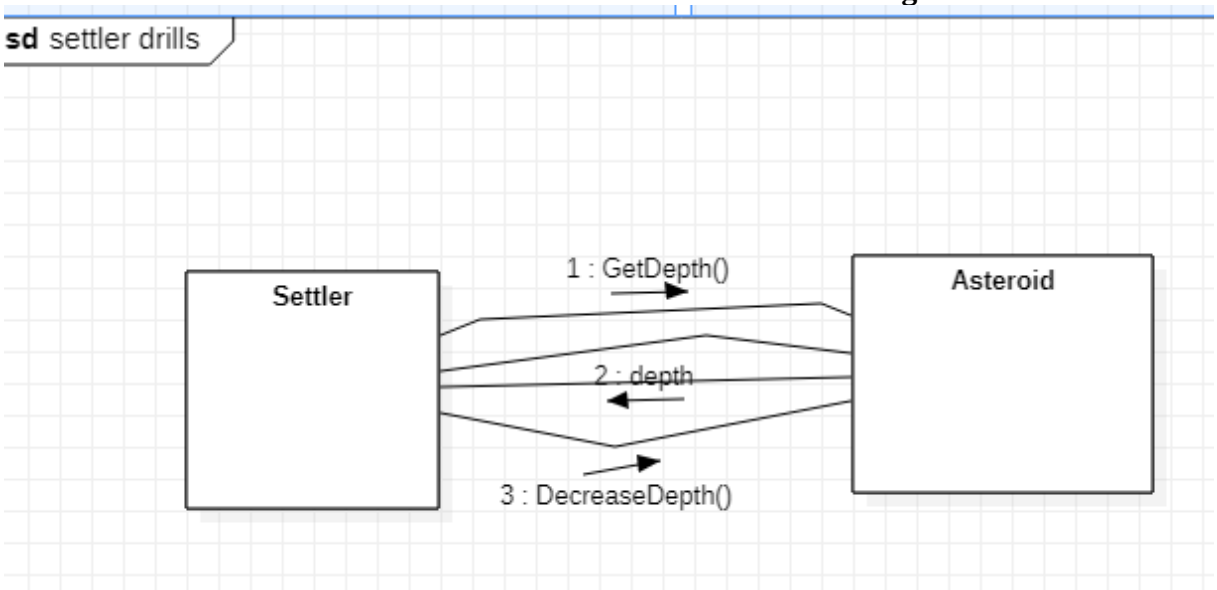
5.4.2 Game Initialization communication Diagram



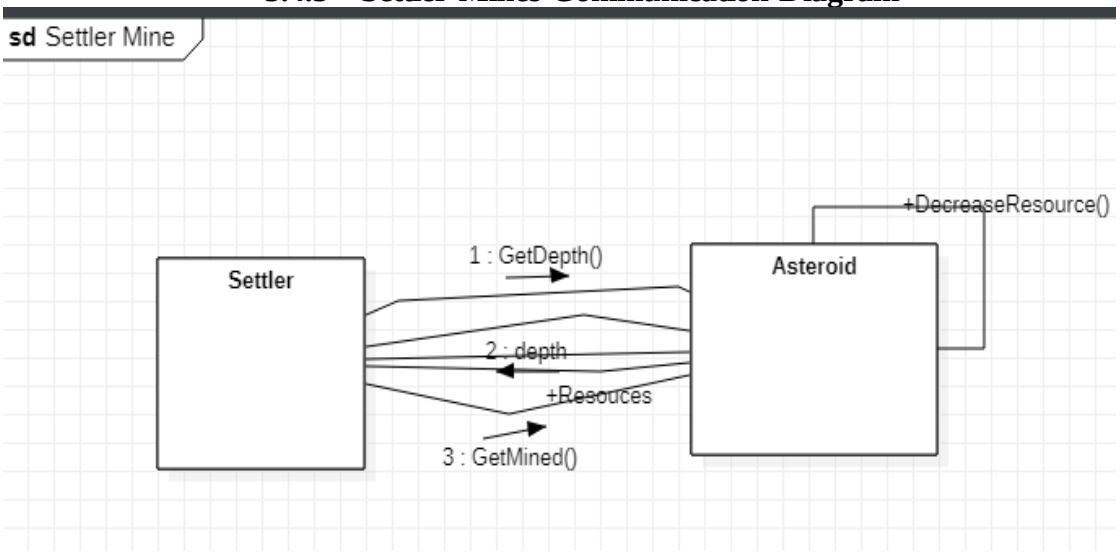
5.4.3 Settler Travels Communication Diagram



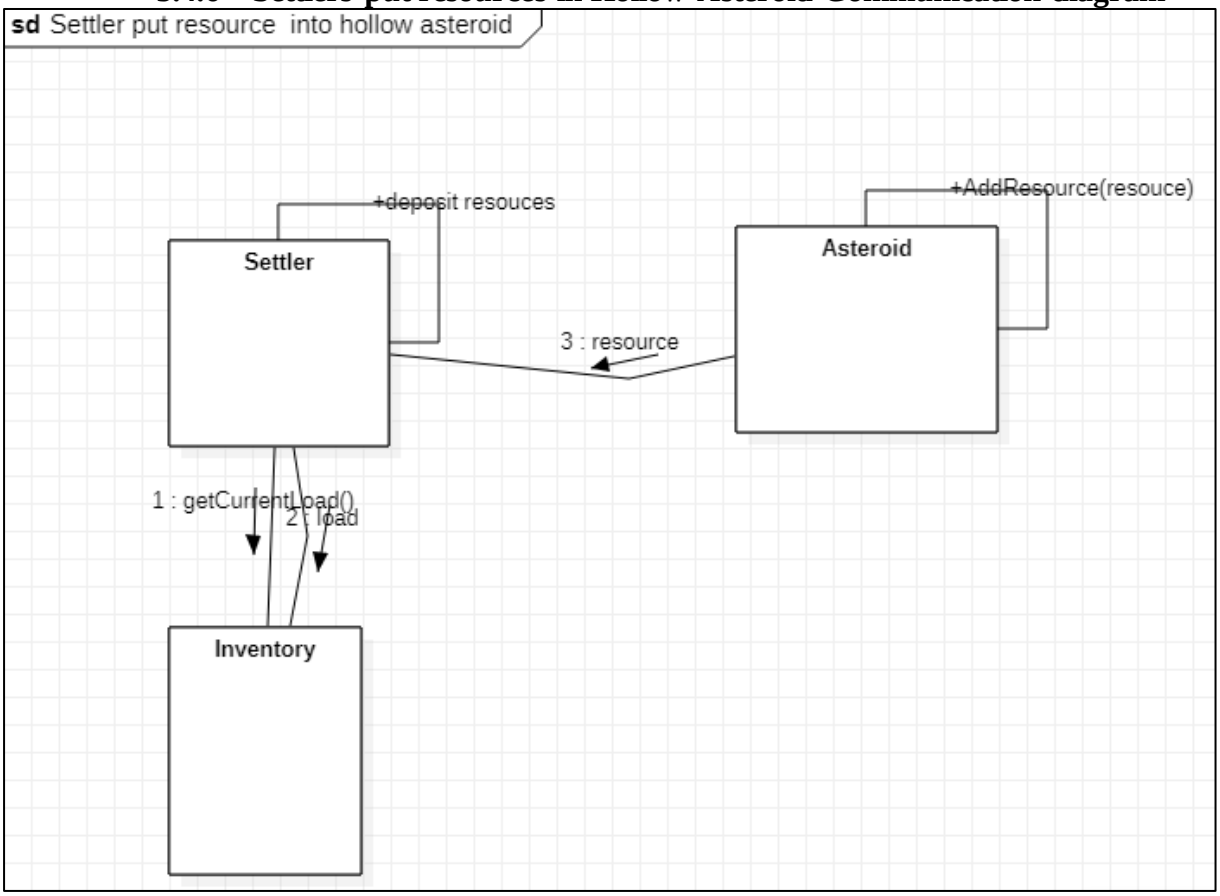
5.4.4 Settler Drills Communication Diagram



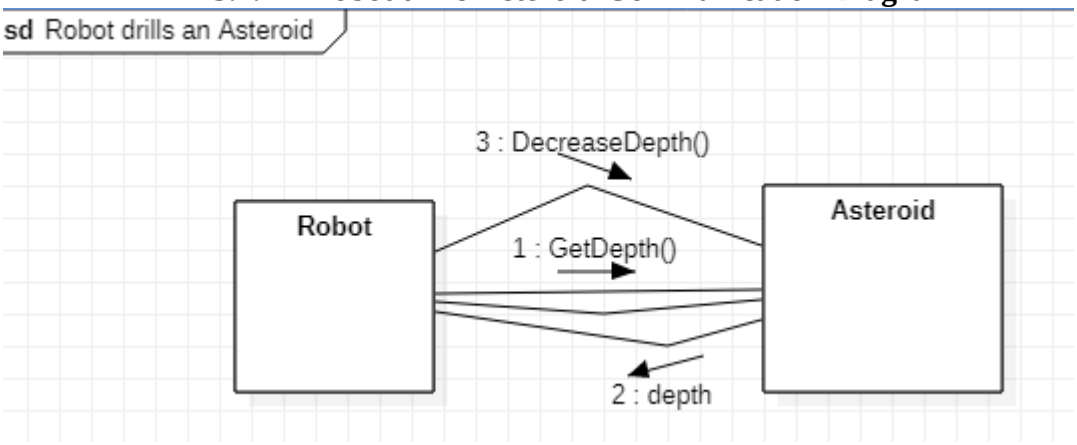
5.4.5 Settler Mines Communication Diagram



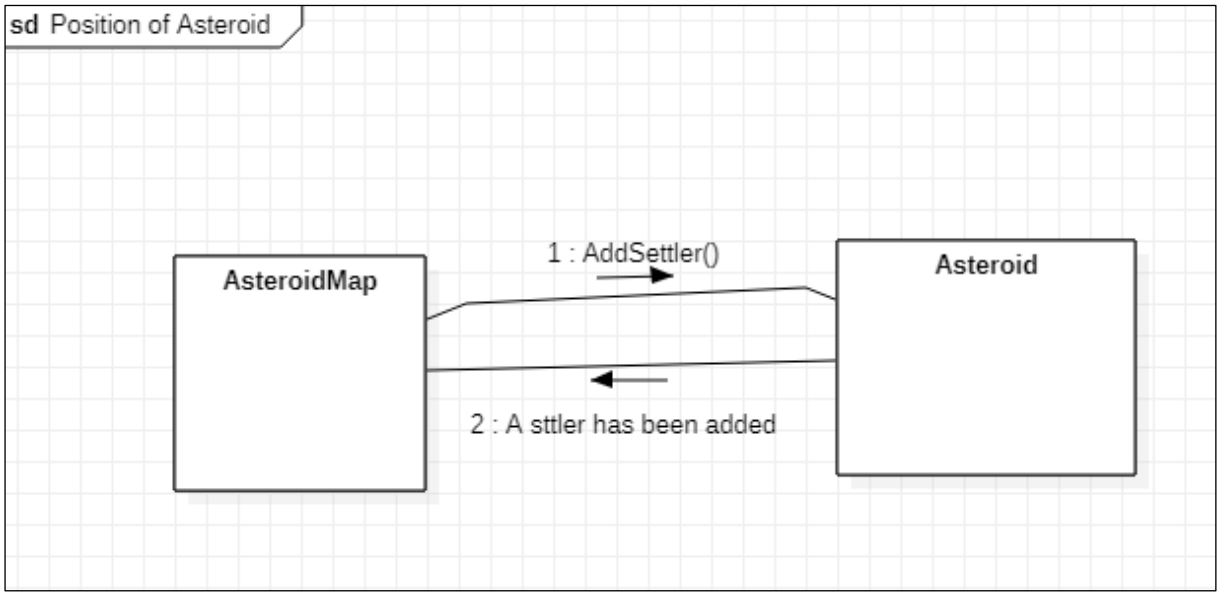
5.4.6 Settlers put resources in Hollow Asteroid Communication diagram



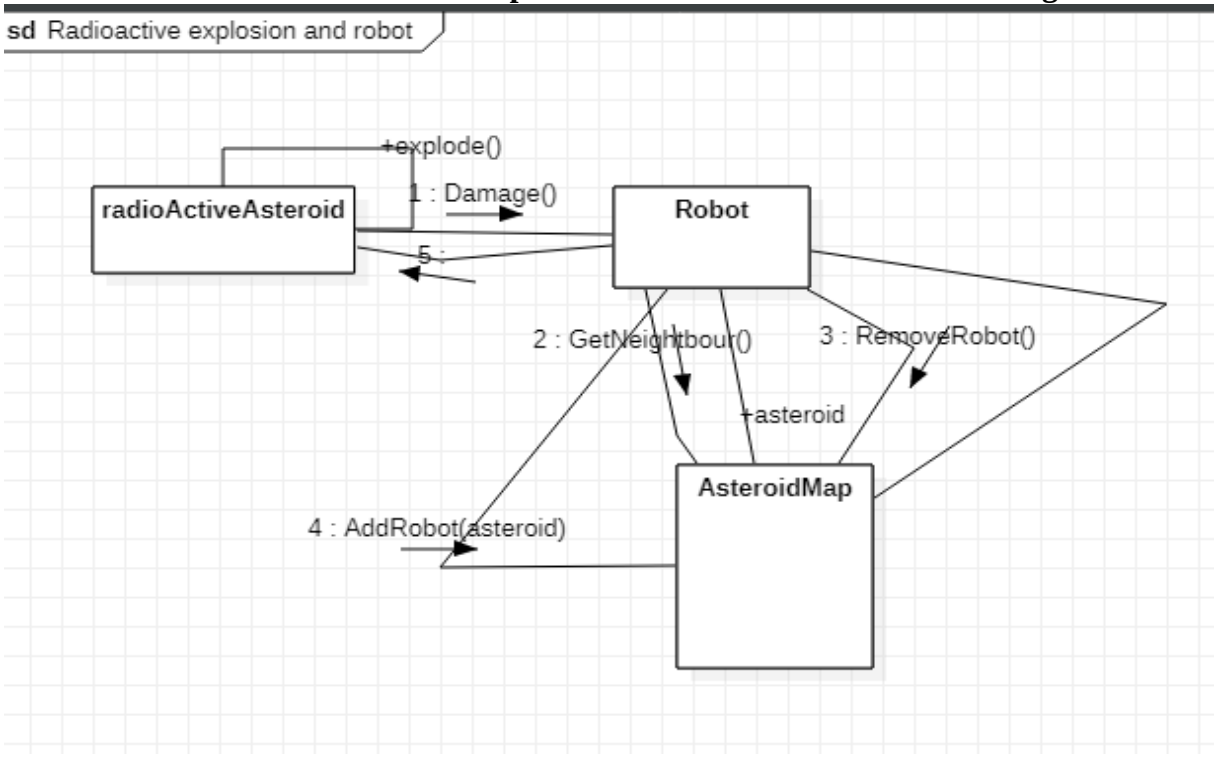
5.4.7 Robot drills Asteroid Communication Diagram



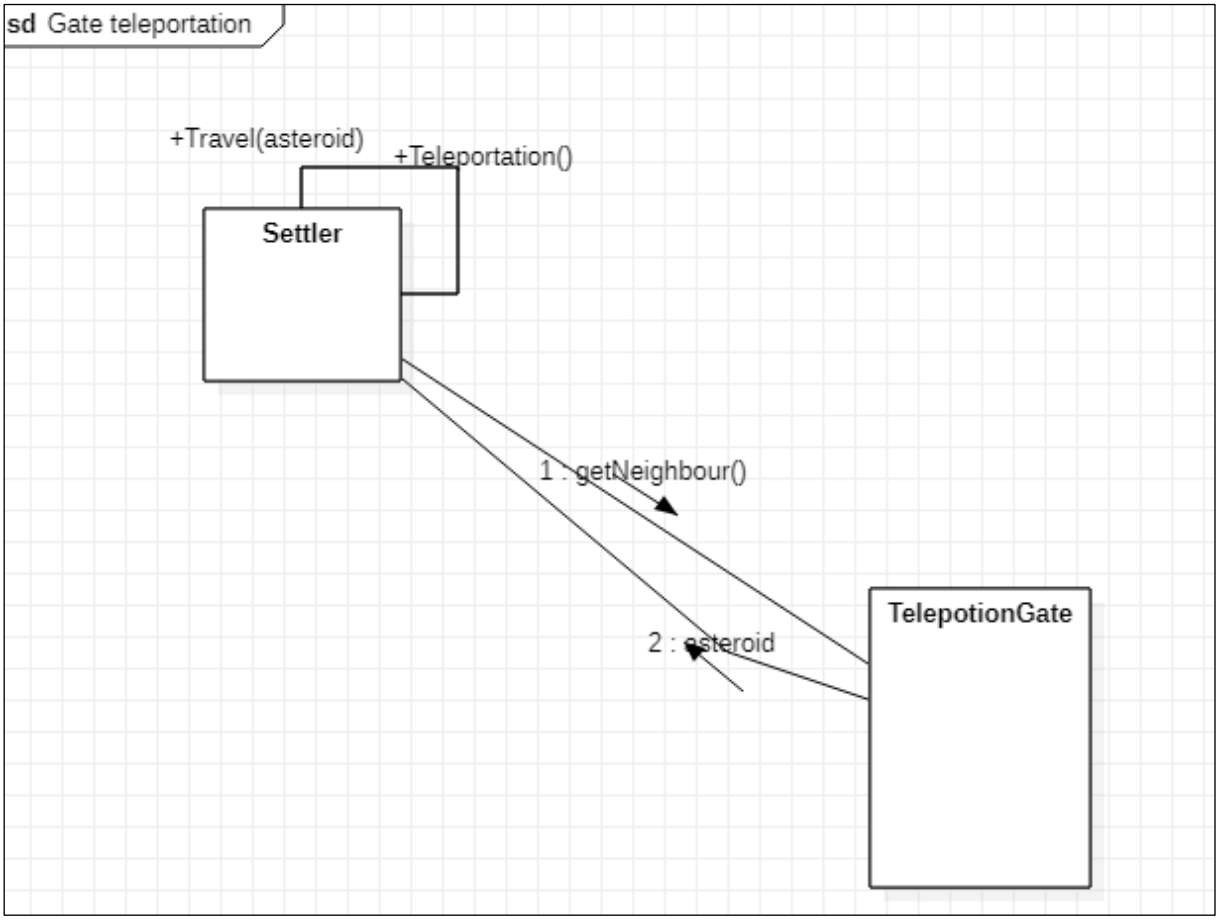
5.4.8 Position of Asteroid Communication Diagram



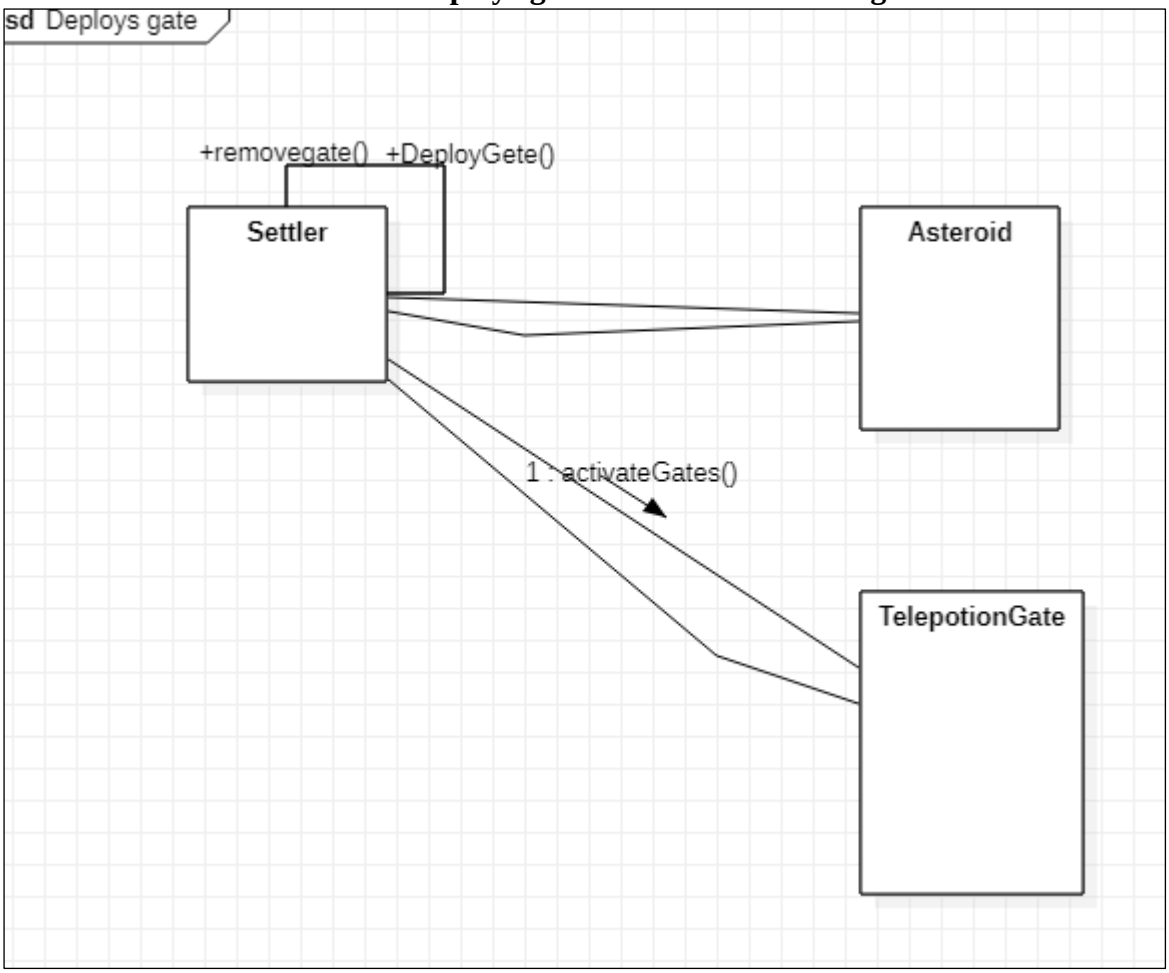
5.4.9 Radioactive Explosion and Robot Communication Diagram



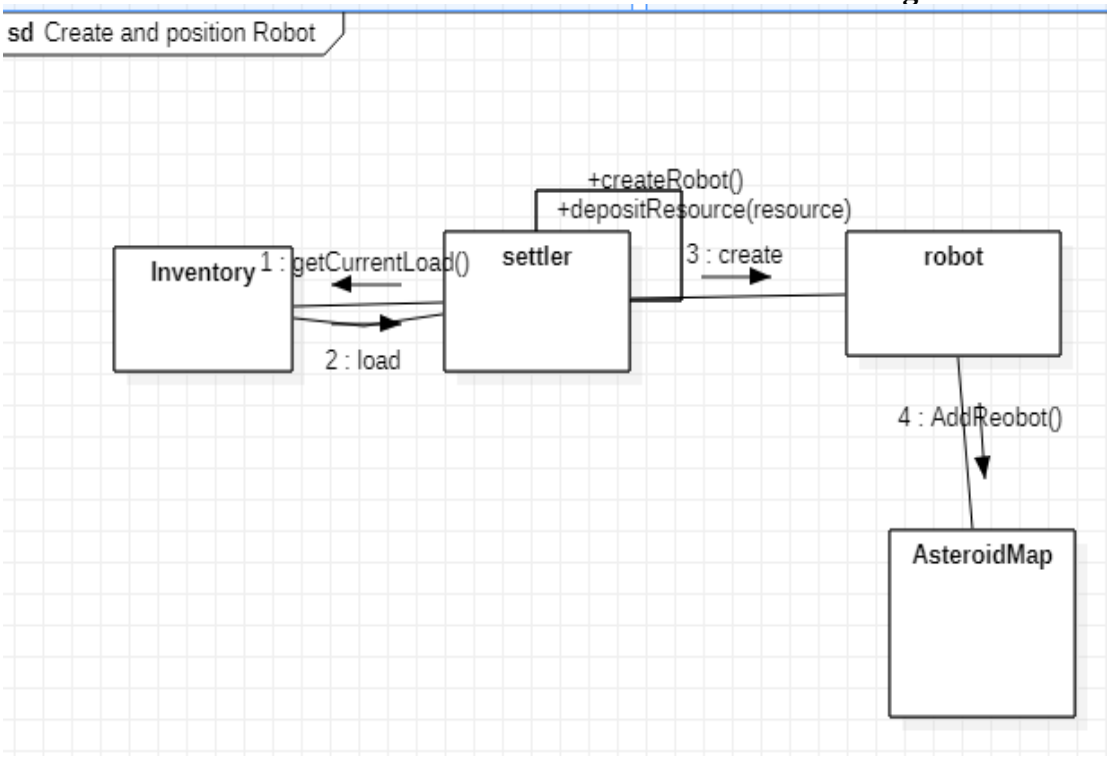
5.4.10 Teleportation Gate Communication Diagram



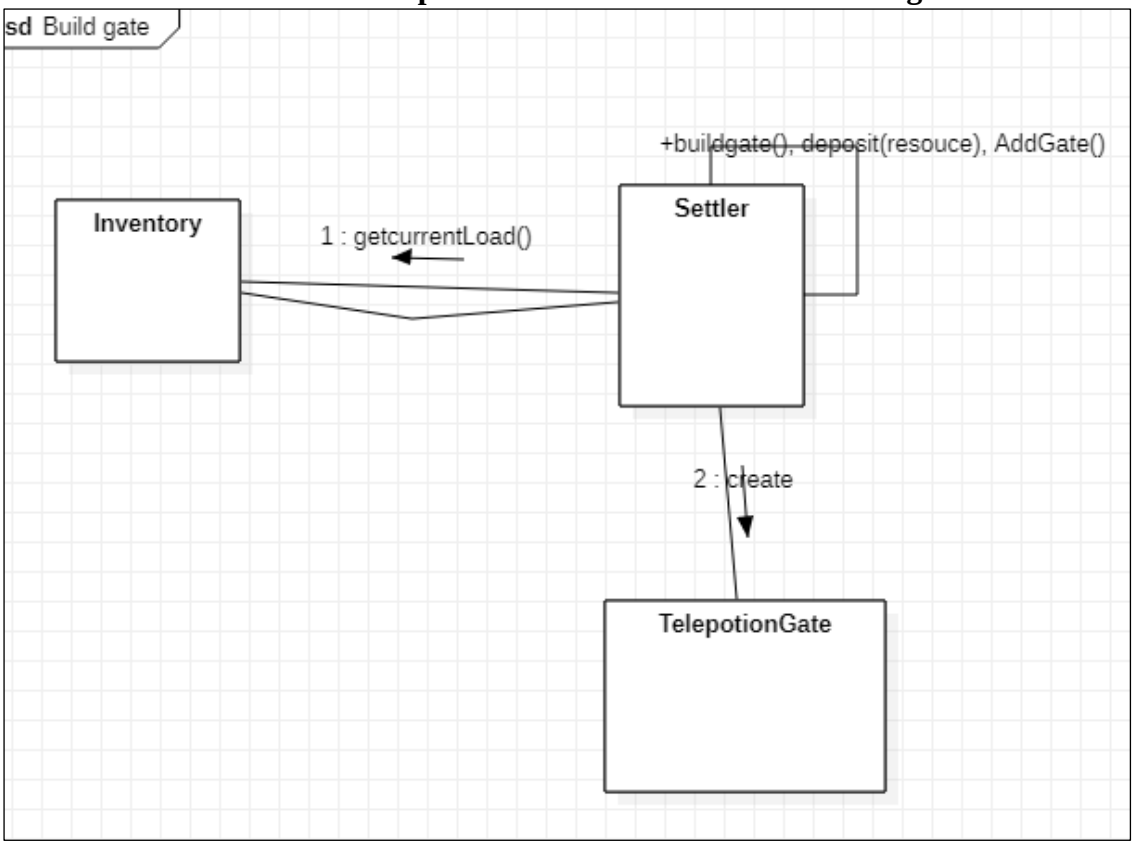
5.4.11 Settler Deploys gate Communication Diagram



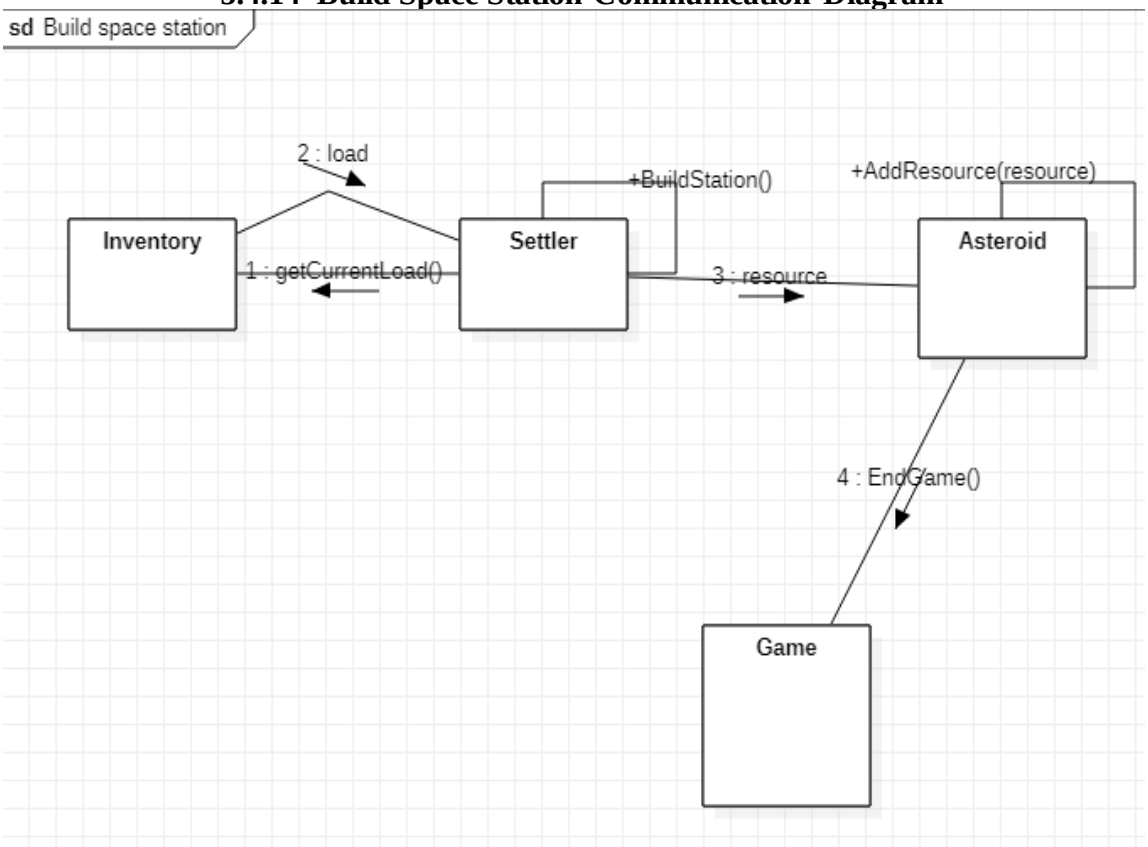
5.4.12 Create and Position Robot Communication Diagram



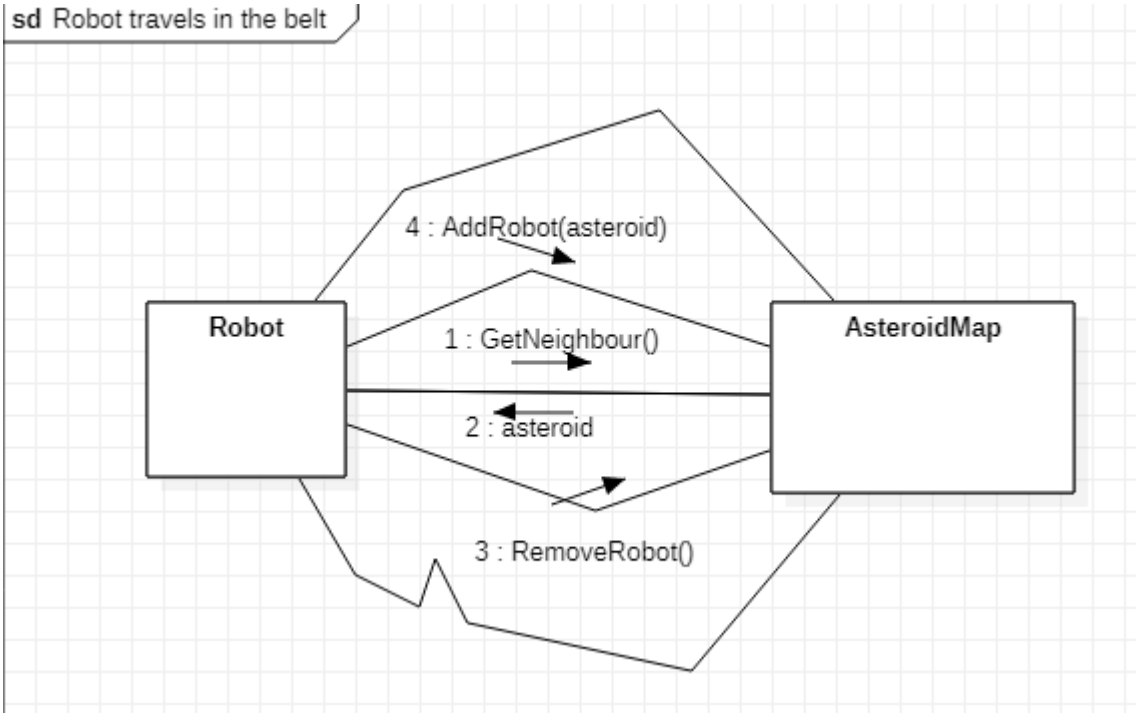
5.4.13 Build Teleportation Gate Communication Diagram



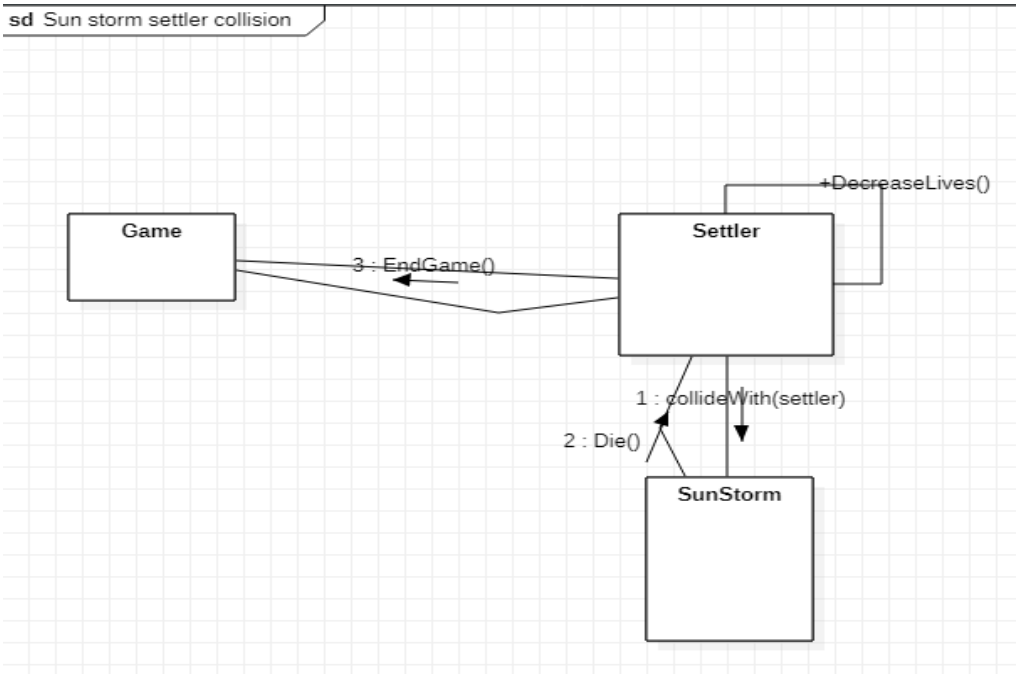
5.4.14 Build Space Station Communication Diagram



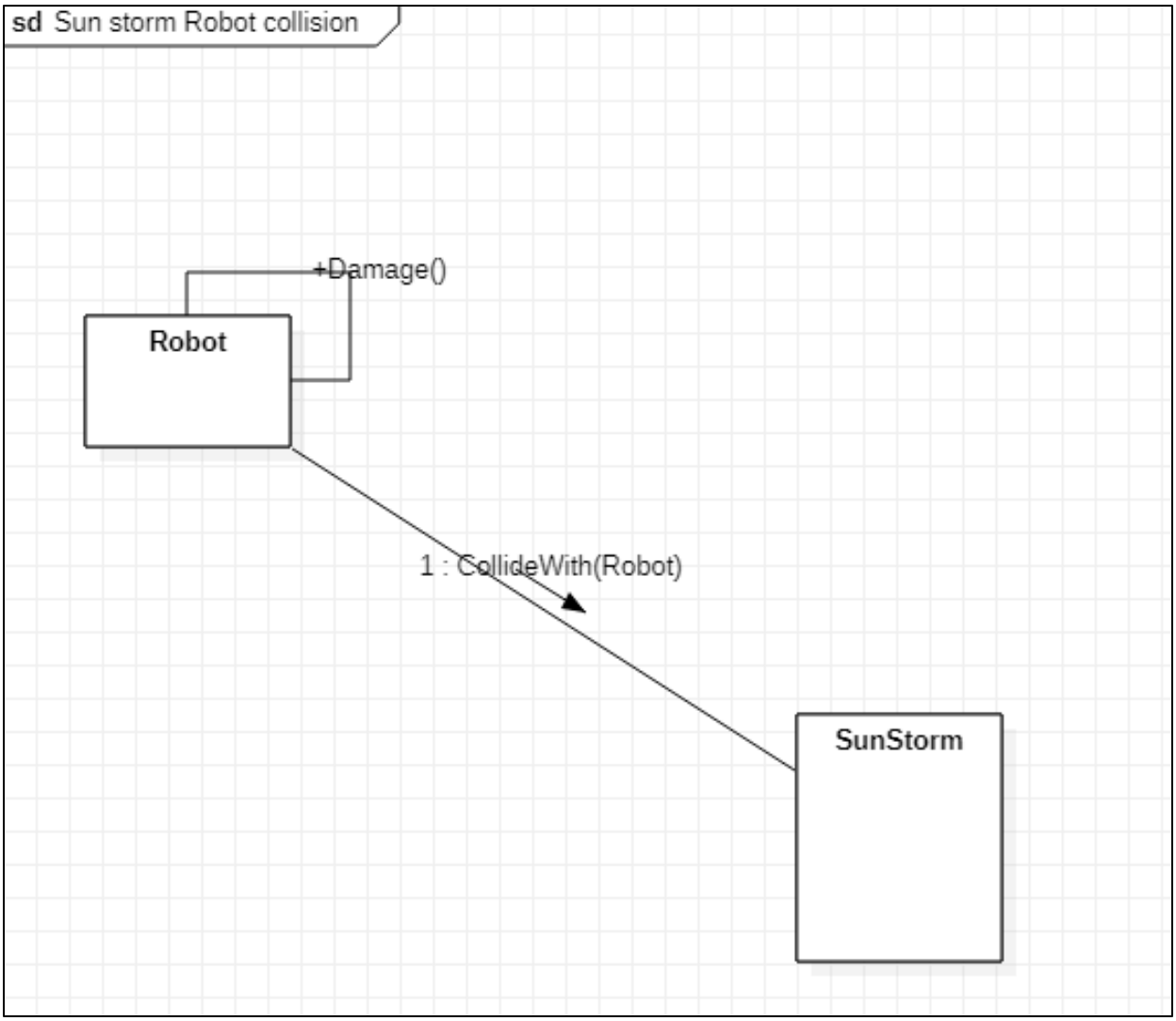
5.4.15 Robot travels in the belt communication diagram



5.4.16 SunStorm Collides with Settler Communication Diagram



5.4.17 SunStorm Collides with Robot Communication Diagram



Protocol

Start (date & time)	Duration (hours)	Performers) name	Activity description
10.03.2021 19:00	2 hours	Team	Meeting: discussion of the task assigned, discussing new requirements, structure of the system, and future implementation details. Wali Ullah 5.1 Ibrar Shakoor 5.1 Ashiq and Idrees 5.2 Orgil 5.3 Vincent 5.4
11.03.2021 14:00	2.5 hours	Wali Ullah	Brainstorming + artifact 5.1
11.03.2021 14:00	2.5 hours	Ibrar Shakoor	Brainstorming + artifact 5.1
13.03.2021 15:00	2.5 hours	Asad Idrees Razak	Completed 5.2 (Plans of the skeleton's UI, dialogs)
13.03.2021 15:00	2.5 hours	Muhammad Ashiq	Completed 5.2 (Plans of the skeleton's UI, dialogs)
14.03.2021 12:00	2+1 hours	Orgil Luis	Brainstorming + artifact 5.3
15:03:2021 06:00	3.0 hours	Vincent Igbineweka	Completed 5.4

6. Skeleton program

6.1 Deployment guide

6.1.1 List of files

File name	Size	Date	Content
Game.java	3KB	18-03-2021	Contains the “Game” class with its attributes and methods.
Settler.java	4KB	18-03-2021	Contains the “Settler” class with its attributes and methods.
Robot.java	1KB	18-03-2021	Contains the “Robot” class with its attributes and methods.
TravelMethod.java	1KB	18-03-2021	Contains the “TravelMethod” class with its attributes and methods.
Timer.java	1KB	18-03-2021	Contains the “Timer” class with its attributes and methods.
Asteroid.java	3KB	18-03-2021	Contains the “Asteroid” class with its attributes and methods.
AsteroidMap.java	2KB	18-03-2021	Contains the “AsteroidMap” class with its attributes and methods.
RadioactiveAsteroid.java	1KB	18-03-2021	Contains the “RadioactiveAsteroid” class with its attributes and methods.
SunStorm.java	1KB	18-03-2021	Contains the “SunStorm” class with its attributes and methods.
Resource.java	1KB	18-03-2021	Contains the “Resource” class with its attributes and methods.
Carbon.java	1KB	18-03-2021	Contains the “Carbon” class with its attributes and methods.
Iron.java	1KB	18-03-2021	Contains the “Iron” class with its attributes and methods.
Uranium.java	1KB	18-03-2021	Contains the “Uranium” class with its attributes and methods.
WaterIce.java	1KB	18-03-2021	Contains the “WaterIce” class with its attributes and methods.
Inventory.java	1KB	18-03-2021	Contains the “Inventory” class with its attributes and methods.
SpaceStation.java	1KB	18-03-2021	Contains the “`SpaceStation” class with its attributes and methods.

6.1.2 Compilation

The main requirements for successful compilation are as following:

- Operating System: Windows
- IDE: IntelliJ IDEA
- JDK 8 or higher

6.1.3 Run

In order to run the application, follow the steps listed below:

1. Open the project files in Intellij IDEA, and navigate to “Game” class
2. Navigate to the main method “public static void main(String []args)” and run using the shortcut: Shift + F10.
   Alternatively, you can press the green run button on the left side of the code. Its right next to the line numbers.
3. You will be presented with a welcome screen in console, and a choice to either start the game or exit. Follow the on-screen instructions to continue the game.

6.2 Evaluation

Name of the team member	Participation (%)
Wali Ullah	16.66%
Ibrar Shakoor	16.66%
Idrees Razak	16.66%
Ashiq Muhammad	16.66%
Macha Orgil Luis	16.66%
Igbineweka Vincent Ifeanyi	16.66%

6.3 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
17.03.2021 20:00	2	Team	Meeting Discussion of skeleton program, Github repository, task assignment, Ibrar and Wali are responsible for: Game, Settler, Robot, Timer, TravelMethod Idrees and Ashiq are responsible for: Asteroid, AsteroidMap, RadioactiveAsteroid, SunStorm, Resource Luis and Vincent are responsible for: Location, Inventory, SpaceStation, Iron, WaterIce, Carbon, Uranium
17.03.2021 18:00	1+3	Wali Ullah Ibrar Shakoor	Implementation of Classes, Attributes, and Methods for: Game, Settler, Robot, Timer, TravelMethod
20.03.2021 20:00	1	Wali Ullah Ibrar Shakoor	Updating implemented methods + adding more comments
21.03.2021 20:00	0.5	Wali Ullah	Completed Artifact 6.1.1: List of Files
21.03.2021 20:30	0.5	Wali Ullah	Completed Artifact 6.1.2: Compilation
21.03.2021 20:30	0.5	Wali Ullah	Completed Artifact 6.1.3: Run
21.03.2021 20:30	0.5	Wali Ullah	Completed Artifact 6.2: List of Files
18.03.2021 21:30	3	Ashiq Muhammad Idrees Razak	Implementation of Classes: Asteroid, AsteroidMap, RadioactiveAsteroid, SunStorm, Resource
20.3.2021 21:00	1	Ashiq Muhammad Idrees Razak	Review, updating and adding comments
18.03.2021	1	Orgil Luis Macha Vincent	Implementation of Classes: Resource, Inventory, Iron, Uranium, WaterIce, Carbon

7. Concept of prototype

7.1 Interface definition of Prototype

7.1.1 General description

Our application will have two ways of receiving input. One of them, the default one will be through the Console. As soon as the application is started it will ask the user for input, giving them a number of choices. The user will introduce the appropriate key depending on his intentions and the application will perform the corresponding action. Another way will be through reading a text file which contains the parameters necessary for generating the asteroid map. That way certain scenarios can be set up. Most probably the text file will contain the parameters separated by the new line "\n". The program will process the commands in the input file line by line. The commands will be grouped logically, such that the next command takes into consideration the result of the previous one.

7.1.2 Input language

<Integer>

Description: At the beginning, a menu is provided where the user is asked to start or exit the game.

Options: 1 - Start the Game
2 - Exit the Game

<Virtual Key>

Description: Now the user is presented with an output which tells that he or she can exit the game any time by pressing the 'Esc' key.

Add Settler<Integer Settler Number>

Description: The user can choose to add a settler in the game.

Options: The user enters the number of the settlers he wishes to add to the game.

Remove Settler<Integer Number of Settler>

Description: The user can choose to remove a number of settlers from the game. Meanwhile, 0 means no settler removed.

Options: The user enters the amount number of the settlers he wishes to remove.

Load<String file path>

Description: The user can choose to load the input parameters from a file or choose to go the manual way.

<Integer>

Description: User is asked for the number of asteroids to input in the asteroid map
Options: The user can choose how many asteroids he/she wants.

<Character>

Description: The user is asked whether he/she wants the asteroid being instantiated to be hollow or not.
Options: The user can enter a character “y/Y” as Yes or “n/N” as No as the answer.

<Integer>

Description: The user is asked to enter the depth of the asteroid he/she just instantiated.
Options: The user can enter the depth as an integer.

<Character>

Description: The user is asked to enter the radioactive state of the asteroid being instantiated.
Options: ‘y’ or ‘Y’ - yes
‘n’ or ‘N’ - no

Remove Asteroid<Integer LocationX, Integer LocationY>

Description: The user can choose to remove an asteroid from the asteroid belt.
Options: the user enters 2 integers as coordinates of the asteroid he wishes to remove.

<Boolean>

Description: The user is asked to enter the teleportable state of the asteroid being instantiated.
Options: 0 - yes
1 - no

Now you can play the Game...

Travel<Integer LocationX, Integer LocationY>

Description: The user is asked to enter the x and y location of the asteroid he wants to travel to.
Options: The user enters 2 integers as coordinates.

Drill<Integer Depth>

Description: The user is asked to enter the amount of core depth he wants to drill,
Options: The user enters an integer value.

Build Robot<>

Description: The user is asked to build a robot.
Options: No options.

Robot Travel<Integer LocationX, Integer LocationY>

Description: The user is asked to enter the x and y location of the asteroid he wants the Robot to travel to.
Options: The user enters 2 integers as coordinates.

Robot Drill<Integer Depth>

Description: The user is asked to enter the amount of core depth he wants to drill by Robot.
Options: The user enters an integer.

Mine<>

Description: The user is asked to mine the resources from the asteroid.
Options: no options.

Build Teleportation Gate<>

Description: The user is asked to build a teleportation gate.
Options: no options.

Deploy Teleportation Gate<>

Description: The user is asked to deploy a teleport gate on an asteroid.
Options: no options.

Pickup Teleportation Gate<>

Description: The user is asked to pick up a teleport gate from the asteroid.
Options: no options.

Deposit Resource<>

Description: The user is asked to deposit the resources he is carrying in a hollow asteroid.
Options: no options.

Restore Health<>

Description: The user can choose to restore his/her health after being damaged by a SunStorm.
Options: no options.

Create SunStorm<Integer>

Description: The SunStorm in the asteroid belt is created with the help of a Random function.
Options: The user can enter the time as an integer for the amount of time the sunstorm will be on.

Create Explosion<Integer>

Description: The game will automatically choose to create an explosion when the Settler drills and mines or when Robot drills only on a Radioactive asteroid.
Options: The integer value provided will determine the damage caused.

Build Space Station<>

Description: The user can enter this command to build the space station.
Options: No option.

Game Over!

7.1.3 Output language

Program started :

Menu: Asteroid Mining
Chose the following options<Integer>:
Description: The game just started and the player is provided with two options whether to play the game or exit it outrightly.

Exit anytime:
You can press ‘Esc’ to end the game anytime.
Description: The player can end the game anytime he wants, therefore the exit() method will be called.

Selection of settlers:
How many setters<Integer> do you want in the game?
Description: Here the player can choose the number of settlers he/she wants them to be part of the game.

Remove the selected settlers:
You can also remove the number of Settlers you want. Just input the number.
Description: Here the imputed value is subtracted from the number of settlers provided earlier but it can not be greater or equal to that number otherwise it will throw an error and will ask to input the number again.

Load from the file:
Do you want to set input parameters for the game from the file or put them manually? Yes or No?
Description: Here the player can provide a character value for 'Y/y' as Yes and 'N/n' as No to initialize the Asteroid, Asteroid Map, Resources etc.

Number of Asteroids:
How many asteroids<Integer> would you like to have in the game?:
Description: Here the player can set the number of Asteroids he/she would like to have in AsteroidMap.

Make Hollow:
Do you want your Asteroid, the current asteroid you just instantiated, to be Hollow<String> or not?
Enter 'Y' or 'y' for YES and 'No' or 'n' for No.
Description: The player can put the value to ensure the Asteroid he/she instantiated is hollow or not.

Give depth:
The asteroid you just instantiated how much depth<Integer> do you want it to have?
Description: Here the Player can provide the depth to the Asteroid which he later will drill into.

Make Radioactive:
The asteroid you just instantiated do you want it to be Radioactive<Character> or not?
Enter 'Y' or 'y' for YES and 'No' or 'n' for No.
Description: Here the Player can input the value to make the asteroid Radioactive or not.

Remove asteroid from asteroid map:
You can also Remove the Asteroid… Just provide it's X and Y coordinates.
Description: Here the player is supposed to enter two integer values which are the X and Y coordinates of the Asteroid he/she wishes to remove. The Remove Asteroid<Integer LocationX, Integer Location Y> is called.

Give Teleportation Gate:
Does the asteroid you just instantiated have a Teleportation Gate<Boolean> or not? Enter 0 for true and 1 for false.
Description: Here the Player can input the value to give the asteroid teleportable state or not.

Now you can play the Game…

Travel:
Do you want to travel< int x, int y>? then put the x and y location to travel to.
Description: Here the Player can input the x and y coordinates in integer form to travel to that location.

Drill:
Do you want to drill< int depth>? the asteroid you're on!
Description: The player is asked whether he/she wants to drill the current asteroid or not. And this awaits for an integer input value.

Build Robot:
Do you want a Robot<> to help you out, Yes or No?
Description: If the player has enough resources he/she can create/build a robot to help him out in drilling and carrying the resources. Therefore the answer is in Yes or No.

Robot Travel:
Do you want the Robot to travel<Integer X, Integer Y>, therefore please provide us the location in x and y coordinates for the Robot to travel to?
Description: Here the player is asked the location coordinates X and Y for Robot to travel to any asteroid located on those coordinates. If the asteroid is not present on the given coordinates the player has to provide again.

Robot drill:
Do you also want your Robot<Integer depth> to drill? Please specify the value you want to drill by?
Description: This awaits for the player to provide an integer value so that Robot can decrease the depth of the asteroid by that amount.

Mine:
Do you want to Mine<> too because your Robot cannot mine? Yes or No?
Description: Here the player can provide a character value for 'Y/y' as Yes and 'N/n' as No for the Settler to mine the asteroid and extract the resources.

Build Teleportation gate:
For your convenience in travel, do you want to build a Teleportation Gate<Char c>? Yes or No?
Description: Here the player can provide a character value for 'Y/y' as Yes and 'N/n' as No for it to build a teleportation gate.

Deploy Teleportation gate:
Alongside, do you want to deploy your Teleportation Gate<Char c>? Yes or No?
Description: Here the player can provide a character value for 'Y/y' as Yes and 'N/n' as No for it to deploy a teleportation gate on the asteroid.

Pick up Teleportation gate:
You can also pick up your Teleportation Gate<Char c>? Yes or No?
Description: Here the player can provide a character value for ‘Y/y’ as Yes and ‘N/n’ as No for it to pick up his/her teleportation gate or not. It will be stored in the inventory.

Deposit Resources:
Resources can be deposited in the hollow asteroid!
Description: Here the player is required to provide rather if the asteroid is hollow his/her resources will be deposited.

Restore Health:
Would you like to restore<> your health? If you do, some of your resources will be automatically decreased. Yes or No?
Description: Here the player can provide a character value for ‘Y/y’ as Yes and ‘N/n’ as No to restore his/her health. After the successful selection of any above option if the Settler doesn't have enough resources the program is going to output the following message.
“You have 30 seconds to collect resources, else you will die!~Good Luck”

Sunstorm:
The sunstorm is coming!!!
Description: Here the player is alerted about the incoming of the sunstorm therefore he/she has 15 seconds to hide or run away from the asteroid that is being affected by the sunstorm. This sunstorm comes from the timed method which is randomised.

Radioactive Explosion:
Boom Boom!!!
Description: This is invoked when the Settler drills and mines or when Robot drills only on a Radioactive asteroid. The Explosion<Integer X> method call causing the X amount of damage.

Build space station:
Do you want to build a SpaceStation?Yes or No?
Description: Here the player can provide a character value for ‘Y/y’ as Yes and ‘N/n’ as No to build a space station or not . If the player has enough resources it will be created else the following message will be printed:
“Not enough resources”

Else:
Your space station is successfully built. Hurray!!!!

Game Over!

7.2 Real use-cases

Use-case name	Start Game
Short textual description	Player can choose to start the game after selecting the number of settlers. Once the game is started, the system generates asteroids in the asteroid belt.
Actors	Player
Dialog, scenario	Player starts the game, the map is initialized with all the elements on it (asteroids, settlers, resources etc). After that, the game is waiting for user input.

Use-case name	View Asteroid
Short textual description	Player views the current state of an asteroid.
Actors	Player
Dialog, scenario	Player chooses the action and the game presents the details of the asteroid like the resources present, the depth of the mantle, the teleportation gates constructed nearby, the radioactive state etc.

Use-case name	View Asteroid Belt
Short textual description	Player views the asteroid belt.
Actors	Player
Dialog, scenario	Player chooses the action to view a map of the asteroid belt, and the game generates the current status of the asteroid belt. This enables the player to view the neighboring asteroids and their locations.

Use-case name	Control Settler
Short textual description	Player controls the settler and can choose various operations to be performed by the settler like travel, drill, mine etc.
Actors	Player
Dialog, scenario	Player chooses the action that is to be performed by the settler, the settler performs it, and makes the necessary changes in the game.

Use-case name	Build Space Station
Short textual description	Player builds the space station using necessary resources.
Actors	Player
Dialog, scenario	Player can check the inventory for resources necessary to build a space station. 1. If the resources are present, the space station is built successfully. 2. In the absence of required resources, the system prompts the player to collect the missing resources.

Use-case name	Build Teleportation Gate
Short textual description	Player builds the teleportation gates using necessary resources. A pair of gates is necessary to teleport. The player knows where the pair of the gate is.
Actors	Player
Dialog, scenario	Player chooses the action to build a teleportation gate. 1. If the resources are present, the teleportation gate is built successfully. 2. In the absence of required resources, the system prompts the player to collect the missing resources. 3. If the player tries to build more than 2 gates using the same settler, the system does not allow it, and displays a warning.

Use-case name	Build Robot
Short textual description	Player controls a settler to build a robot using required resources.
Actors	Player
Dialog, scenario	Player chooses to build a robot. 1. If the resources are present, the robot is built successfully. 2. In the absence of required resources, the system prompts the player to collect the missing resources to build a robot.

Use-case name	Control Robot
Short textual description	The system controls the robot. The robot can assist the settler by performing various actions.
Actors	System
Dialog, scenario	1. The system is able to control the robot for drilling the asteroids, or hiding in a hollow asteroid to avoid being damaged by Sunstorms. 2. The robot is also unaffected by radioactive explosions. 3. The robot can travel using normal travel or teleportation.

Use-case name	Make Move
Short textual description	Player controls the settler to perform one move at a time.
Actors	Player
Dialog, scenario	When a player chooses this action, the game will show a menu of different operations that a settler can perform on an asteroid. The menu consists of following actions: 1. Travel 2. Mine 3. Drill 4. Check Inventory 5. Drop Resources

Use-case name	Travel
Short textual description	Player controls the settler to travel to neighboring asteroids. The system controls the robot to travel.
Actors	Player, System
Dialog, scenario	Player is presented by 2 choices to travel: 1. Normal Travel 2. Teleportation Before travelling, the player knows the location of the destination asteroid that it is travelling to. Before teleportation, the player has to check for the gate pair. The system can control the robot to travel in the asteroid belt.

Use-case name	Mine
Short textual description	Player can control the settler to mine the asteroid for resources/minerals.
Actors	Player
Dialog, scenario	Player chooses to mine, and the settler mines one unit of the resource in one move, and adds it to the inventory. The game updates the inventory status and modifies the quantity of resources. Resources can not be mined from a hollow asteroid.

Use-case name	Drill
Short textual description	Player controls the settler to drill an asteroid. System controls the robot to drill an asteroid.
Actors	Player, System
Dialog, scenario	Player chooses the action and the settler drills the asteroid one unit at a time. The game keeps check of the depth of the mantle as it's being drilled by a settler or the robot. When an asteroid is drilled to its core, then it is marked as a hollow asteroid. 1. The asteroid is successfully drilled to it's core if radioactivity is at aphelion, or if the asteroid is not radioactive at all. 2. The drilling is unsuccessful if the asteroid radioactivity is at perihelion.

Use-case name	Drop Resources
Short textual description	Player controls the settler to drop the resources in its inventory in a hollow asteroid.
Actors	Player
Dialog, scenario	Player chooses the action and the settler checks if the asteroid is empty. The game updates the inventory status only if the action is successful. 1. The resources are successfully dropped if the asteroid is hollow.. 2. The resources can not be dropped if the asteroid is not hollow.

Use-case name	Check Inventory
Short textual description	Player controls the settler to check the inventory of the resources.
Actors	Player
Dialog, scenario	Player chooses the action to view the amount and the types of resources present in the inventory of a settler. The inventory also knows how many gates the settler is carrying.

Use-case name	Control SunStorm
Short textual description	The system controls the sun storms in the game.
Actors	System
Dialog, scenario	The system decides when to create a sunstorm in the asteroid belt. It also controls the duration, the damage, and the range of the sunstorm. 1. The sunstorm successfully damages the robot/settler if they are exposed. 2. The sunstorm does not damage the robot/settler in case they are hiding in a hollow asteroid.

Use-case name	Control Explosion
Short textual description	The system controls the explosions of the radioactive asteroids in the game.
Actors	System
Dialog, scenario	The system determines the radioactive state of an asteroid. 1. The explosion damages the settler if they are exposed to the explosion. The settler dies and loses all the resources it is carrying. In this case, the teleportation gates built by that settler are marked inactive and can not be used. 2. The robots can survive the radioactive explosion, but they are immediately transported to the nearest asteroid by the game system.

Use-case name	Control Resources
Short textual description	The system controls the resources in the game.
Actors	System
Dialog, scenario	The system generates the resources in the asteroid belt when the game is initialized. The system also keeps track of the changes when the settlers mine or drop the resources.

Use-case name	End Game
Short textual description	The system can end the game if all the settlers die, or if the space station is built. The player can also manually quit the game at any given time.
Actors	Player, System
Dialog, scenario	1. Player chooses the action to end the game manually. 2. The system realizes that either all the settlers are dead, or the space station is built successfully and ends the game.

7.3 Test plan

Name of the test-case	Player creation
Goal	Game behavior test
Short description	Check if a player has been created, and with an empty inventory
Name of the test-case	Mining a resource
Goal	Game logic test
Short description	Check if the asteroid has been drilled all the way through, and settler/robot has an adequate amount of free slots in their inventory, and most importantly, a settler/robot is in the vicinity of the asteroid.
Name of the test-case	Crafting a robot
Goal	Game logic test
Short description	Check if the settler has the required types and amount of resources to craft a robot. When craft command is given, and the requirement is met, check if a new robot was created. Otherwise, an error message regarding the situation is prompted.
Name of the test-case	Crafting a teleportation gate
Goal	Game logic test
Short description	Check if the settler has the required types and amount of resources to craft a teleportation gate. When craft command is given, and the requirement is met, check if a pair of teleportation gates is added to the inventory. Otherwise, an error message regarding the situation is prompted.
Name of the test-case	Building a space station
Goal	Game behavior test
Short description	Check if the settler has the required types and amount of resources to build a space station, and game state changes to ‘win condition is met’. Otherwise, an error message regarding the situation is prompted.
Name of the test-case	Activating a pair of teleportation gate
Goal	Game logic test
Short description	Checks if both of a single pair of teleportation gates were deployed on different asteroids. And check if they are active (they must be).
Name of the test-case	Inactive teleportation gate
Goal	Game logic test
Short description	Checks if an inactive teleportation gate has the other pair which is not yet deployed on an asteroid.

Name of the test-case	Sum storm deals damage to a traveller
Goal	Game behavior test
Short description	Checks if there is a currently active sun storm: 1. If it collides with a settler, the settler must die. 2. If it collides with a robot, the robot must take damage. 3. If the travellers were hidden, the settler must not die, and the robot must not be displaced.
Name of the test-case	Perihelion explosion
Goal	Game behavior test
Short description	Checks if a radioactive asteroid is at the perihelion has exploded: 1. If a settler was on the asteroid, it must die. 2. if a robot was on the asteroid, the robot must be displaced to a neighboring asteroid.
Name of the test-case	Drop resource to a hollow asteroid
Goal	Game logic test
Short description	Checks if the current asteroid is hollow, and has been drilled all the way through. Only then, the selected resources are dropped from the inventory. And finally checks if the dropped resource's type and amount are the same as the one in the asteroid.
Name of the test-case	Hiding in asteroid
Goal	Game logic test
Short description	Checks if the asteroid is empty, has been drilled all the way through. And finally checks if the state of the traveller who took the hide action is indeed changed to 'hidden'
Name of the test-case	All lives are lost
Goal	Game logic test
Short description	Checks if the game state is changed to 'failure' when the settler lives reach 0.
Name of the test-case	Performing operations
Goal	Game behaviour test
Short description	Check if the settle is not calling more than one operation. For example, a settler cannot travel and hide at the same time.
Name of the test-case	Checking the core of an Asteroid
Goal	Game logic test
Short description	Settlers check if the asteroid core is hollow or not before mining.

Name of the test-case	Mining Uranium
Goal	Game logic test
Short description	Check if the asteroid is at aphelion before mining

7.4 Support programs for testing

So far, we have developed a utility program which is responsible for instantiating the Asteroid Map and Asteroids and other elements included like resources and their quantities, locations, hollow and radioactive states etc. It allows us to run different test cases and check the printed output on the console.

For the future, we have planned to use JUnit testing for our application. It will allow us to validate code results and states for state testing, and can also execute sequence of events for behavior testing.

7.5 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
24.03.2021 19:00	2 hours	Team	Meeting: discussion of the task assigned, discussing new requirements, structure of the system, and future implementation details. Wali Ullah 7.1.1 & 7.4 Ibrar Shakoor 7.1.2 Ashiq 7.1.3 Asad Idrees 7.2 Orgil, Vincent 7.3
25.03.2021 14:00	2.5 hours	Wali Ullah	Brainstorming + artifact 7.1.1 & 7.4 Activity:Describing the ways in which our program will accept input.
25.03.2021 14:00	2.5 hours	Ibrar Shakoor	Brainstorming + artifact 7.1.2 Activity: Input Language
25.03.2021 15:00	2.5 hours	Asad Idrees Razak	Completed 7.2 Activity: Defining the real use cases and describing them.
25.03.2021 15:00	2.5 hours	Muhammad Ashiq	Completed 7.1.3 Activity: Output Language
26.03.2021 12:00	2.0 hours	Orgil Luis Macha	Brainstorming + artifact 7.3
26:03:2021 06:00	2.0 hours	Vincent Igbineweka	Completed 7.3

8. Detailed plans

8.1 Design level plan of classes

8.1.1 Game

- **Responsibility**

The responsibility of the Game class is to start the game. It initiates an asteroid map using a 2D array and randomly generates asteroids within the array, and also randomly assigns which resources are needed to build the space station and the teleportation gate. It is also responsible for causing sunstroms inside the asteroid belt. The class also has the responsibility to end the game if the space station is built, or if all the settlers die, or if the user wants to exit the game at any time.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+resourcesToWin:Map<String, int>** – Amount of resources needed to build the space station therefore winning the game
  - **+resourcesToBuildGate:Map<String, int>** – Amount of resources needed to build a teleportation gate used to travel
  - **+asteroidMap:[ ][ ]<int>** – 2D Array used as a reference for asteroid map. Each asteroid is marked at its index by value “1”, while the absence of an asteroid is marked by “0”.
  - **+numberOfAsteroids:<int>** – Total number of asteroid that the user wishes to have inside the asteroid map.
  - **+nameOfResources:Map<String, int>** – Hashmap that contains the name of all the resources as a String, and their quantities as Integer values respectively.

- **Methods**
  - **+ Game( ):** Constructor responsible for initializing different variables like asteroidMap, nameOfResources, resourcesToWin, and resourcesToBuildGate etc.
  - **+void StartGame( )**: Player starts the game at any time, the method makes further calls to InitializeAsteroidMap, InitializeResourcesToWin, InitializeResourcesToBuildGtae, and GameMission methods.
  - **+void EndGame( )**: Player ends the game at any time or when the game is lost or won.
  - **+void GameMission( )**: Prints the requirements of the game e.g. the amount of resources needed to build the space station
  - **+void InitializeAsteroidMap( ):** When the game is started, this method is used to initialize the asteroid map. This function distributes the asteroids across the asteroid belt randomly.
  - **+void InitializeResourcesToWin( ):** When the game is started, this method is used to randomly set the requirements of the game e.g. the amount of resources needed to build the space station.
  - **+void InitializeResourcesToBuildGtae( ):** When the game is started, this method is used to randomly set the amount of resources needed to build a teleportation gate used to travel
  - **+void CreateSunStorm( ):** This method causes the sunstorm on the asteroid belt.
  - **+void Random(int min, int max):** This method returns a random integers between the minimum and maximum parameters passed in the function.

8.1.2 Settler

- **Responsibility**

The responsibility of the Settler class is to travel, drill, and mine the resources from the core of the asteroid to build the space station. It also allows the settler to build and deploy teleportation gates. The settler can die from the damage, but can also revive its health. If the settler has the necessary resources, it can build a robot.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+Lives: <int>** – This variable stores the lives/health of settlers in the asteroid belt as an integer number.
  - **+-collectedResources:Map<String, int>** – This hashmap stores the number of resources collected by settlers.
  - **+isHidden:<bool>** – This boolean variable defines the state of the settler whether its hidden inside a hollow asteroid or not.
  - **+numberOfAsteroids:<int>** – Total number of asteroid that the user wishes to have inside the asteroid map.

- **Methods**
  - **+void Travel(Asteroid a, Travelmethod travelMethod)**: This function enables the settler to travel to an asteroid using normal travel or teleportation.
  - **+void Drill(Asteroid a)**: The settler drills the asteroid to make it hollow for its survival, or to mine the resources later.
  - **+void Mine(Asteroid a)**: Settler collects the resources from the asteroid, the quantity inside the asteroid is decreased, while the quantity inside the collectedResources is increased.
  - **+void Hide(Asteroid a)**: Settler needs to drill the asteroid to make it hollow and hide itself for protection against radioactive explosion and sun storm
  - **-void ReviveHealth( )**: This function allows the settler to revive health after it's damaged.
  - **+void Die( )**: Settler dies due to the sun storm, radioactive explosion or improper drilling of the asteroid
  - **+void Deposit( ):** The collected resources are added into the resource list and the quantity is increased.
  - **+void BuildTeleportationGate( )**: Settler collects resources from neighbouring asteroids to build a teleportation gate, which is added to the inventory.
  - **+void DeployTeleportationGate(Asteroid a)**: Settler deploys the teleportation gate, and it is removed from the inventory.
  - **+void PickUpTeleportationGate(Asteroid a)**: Settler picks up the teleportation gate, and is added to the inventory.
  - **+void BuildRobot(Asteroid a)**: Settler collects resources to build a robot.
  - **+void BuildSpacestation(Asteroid a)**: Settler collects resources from neighbouring asteroids to a single asteroid to build the space station

8.1.3 Robot

- **Responsibility**

The robot class shares some common responsibilities as settler class such as travel and drill. The robot can also get damaged, but its not destroyed by the radioactive explosions.

- **Superclasses**

Settler

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

There are no attributes in this class.

- **Methods**
  - **-void Damage()**: If the robot cannot find a hollow asteroid, they can be destroyed by the sun storm.

8.1.4 Location

- **Responsibility**

The class tells the current position of both settler and robot with respect to their x and y coordinates in the asteroid belt. Also tells the position change of Settler/Robots in the asteroid belt after going through the Teleportation Gate. It can also be used to get the x and y coordinates of an Asteroid which are critical to the game functions.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+xLocation:int** X coordinate of the position of an object in the asteroid belt.
  - **+yLocation:int** Y coordinate of the position of an object in the asteroid belt.

- **Methods**
  - **+Asteroid GetCurrentLocation( )**: Gives the current location of the object in the asteroid belt
  - **+Asteroid GetNextLocation( ):** Tells about the next location of the asteroid that the settler/robot will travel to.

8.1.5 Travel Method

- **Responsibility**

Tell the travelling method used by Settler or the Robot, and also determine the amount of distance covered by them and their destination.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+destination:int** This variable tells the destination (neighbouring asteroid)where Settler/Robot is travelling to.

- **Methods**
  - **+void NormalTravel(Asteroid a)**: It is the slower kind of traveling. If settlers or robots travel using this, they take a lot of time.
  - **+void Teleportation(Asteroid a)**: It is the special travelling method using teleportation gates and its faster.

8.1.6 Timer

- **Responsibility**

It tells about the required time to perform different operations like drilling, mining, travelling, reviving the settler and also the duration of the sunstorm.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+healthReviveTime:int -** This variable stores the amount of time it takes to revive either Settler or Robot
  - **+sunStormTime:int -** This variable stores the amount of time for which the SunStorm hits
  - **+travelTime:int -** This variable stores the time taken for Settler/Robot to travel from one place to another (change of location)
  - **+drillTime:int -** This variable stores the time taken for drilling the asteroid core
  - **+mineTime:int -** This variable stores the time taken to mine the asteroid core

- **Methods**
  - **+int ReviveTimeCalculator(int Lives)**:This method calculates the time required to revive health based on the current health of the settler.

8.1.7 Asteroid

- **Responsibility**

Asteroid object keeps track of the type of asteroid, if it's hollow or not. It also determines which resources and the amount of the resources are present on a particular asteroid. Alongside that it knows the radioactive states of the asteroid i.e; whether it's at perihelion or aphelion. If there is a teleportation gate constructed on the asteroid or not is also present in the object. It serves as a source for the settler to collect the resources to build the space station.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **#depth:int –** This variable stores the depth of the asteroid
  - **+resourceHashMap: HashMap<Object, Integer> –** This hashmap represents the list of the resources an asteroid contains
  - **+resourceName: String –** It tells us the name of the resource that is present in the core of the particular asteroid we are on
  - **+resourceQuantity: int –** It tells us the quantity of the resource that is present in its core
  - **+resource:Resource –** This variable stores the type of resources and their amounts which can be found by the settler
  - **+isHollow:Bool –** This variable checks whether the Asteroid is hollow or not

- **+isRadioactive:Bool –** This variable tells whether the state of radioactive asteroid is at perihelion or aphelion
- **+isTeleportable: Bool –** This variable stores whether is it possible for Settler or Robot to teleport to a different location from this asteroid or not
- **+asteroidLocationX: int –** This variable stores the x coordinate of the location of the asteroid in asteroidMap
- **+asteroidLocationY: int –** This variable stores the y coordinate of the location of the asteroid in asteroidMap

- **Methods**
  - **+int getDepth():** Returns an integer variable to show the depth of the asteroid/rocks
  - **+void decreaseDepth():** If the settler mines the asteroid the depth value of that particular asteroid is changed
  - **+Bool getMined():** It shows whether the asteroid is minable or not. If radioactivePhase and checkHollow are not true its mean asteroid can be mined and it returns true value otherwise returns false
  - **+int Random(int min, int max):** It computes and randomly chooses an integer value in the provided range which in this case is min and max
  - **+int addResources():** This method assigns the resources to the asteroids at the start of the game
  - **+int decreaseResource():** If the settler mines the asteroid the amount of resources of that particular asteroid is decreased with the help of this method
  - **+Bool checkTeleportationGate():** It checks whether there is a teleportation gate constructed on the asteroid or not
  - **+String toString()**: This is a custom method which overrides the built in method toString() and help us print the Object and their corresponding values onto the console

8.1.8 AsteroidMap

- **Responsibility**

Asteroid map acts like a map and determines the total asteroids present in the asteroid belt and acts as a navigation for travelling across the asteroids. At any given time, it can be used by the player of the game to check the neighboring asteroids. It can be accessed by settlers and robots. It can also keep track of settlers and robots present in the asteroid belt.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class

- **Attributes**
  - **+totalAsteroids:ArrayList –** This variable shows all the asteroids instantiated and contained in a list

- **Methods**
  - **+Asteroid GetNeighbour(Asteroid a):** Returns the object of neighbouring asteroid for the settler
  - **+void AddSettler(Asteroid a, Settler s):** Adds the settler in the chosen asteroid
  - **+void AddRobot(Asteroid a, Robot r):** Adds the robot in the chosen asteroid to assist the settler
  - **+void RemoveSettler(Settler s):** Moves/Remove the settler from the asteroid
  - **+void RemoveRobot(Robot r):** Moves/Remove the robot from the asteroid
  - **+void RemoveAsteroid(Asteroid a):** If asteroid is radioactive this method is used to remove that asteroid to help us keep track of asteroids

8.1.9 SpaceStation

- **Responsibility**

Helps the settler to survive on the asteroid. It constitutes the main goal of the game.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class

- **Attributes**

  +CheckStateofResources:Bool –With the help of this attributes we will kep track of inventory if it's full or not means all the required resouces are present or not

- **Methods**

  +void SpaceStationCreation(Spacestaion S): This method will take the SpaceStation class object as an orgument and behalf on this, it will going to print SpaceStation has been created on the console.

8.1.10 SunStorm

- **Responsibility**

This object is responsible for the sun storms that can occur in the asteroid belt. The amount of damage done by the sunstorm to the settler can also be determined by this object. Settlers and robots can avoid being damaged by sun storms if they hide inside a hollow asteroid. In conclusion, it causes damage to both the Settler and Robot for a particular duration of time.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+time:Long –** This variable stores duration (period) of the sunstorm
  - **+damageDone:Long –** This variable tells the amount of damage caused to a Settler or Robot by decreasing their health, by which it also represents their health as well.
- **Methods**
  - **+void CollideWith(Settler s):** This method destroys the settler upon collision with the sunstorm
  - **+void CollideWith(Robot r):** This method confirms the damage of the robot in case of a collision with the sunstorm

8.1.11 RadioactiveAsteroid

- **Responsibility**

This object is responsible for the radioactive explosions that can occur on an asteroid. It keeps track of the radioactive phases (aphelion/perihelion). When the explosion happens, the settler loses all its health, but a robot can survive. This behavior is controlled by this object. In conclusion, it carries radioactive resources and explodes if the settler is mining/drilling it during the perihelion period.

- **Superclasses**

Asteroid.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **-state:Bool –** This variable stores the state of the asteroid according to its position (aphelion or perihelion)

- **Methods**
  - **+void Explode(Settler s):** This method is used for explosion of the asteroid while being drilled by Settler causing its death
  - **+void Explode(Robot r):** This method is used for explosion of the asteroid while being drilled by Robot but keeping it intact with some damage

8.1.12 Inventory

- **Responsibility**

This class is responsible for tracking items collected by the settlers. Instantiated once for each live settler.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+currentLoad:ArrayList<Resource>** - This variable is a list that stores the resources collected by the settler.
  - **+totalLoad:int** - This variable denotes the total amount of resources a settler can carry at any given time.

- **Methods**
  - **+ArrayList<Resource> GetCurrentLoad()**: This method is used to get the current content of the settler's inventory.
  - **+void addResource(Resource)**: This method receives a resource as an argument, and tries to add it to the inventory. Program raises an error if the inventory is full; otherwise, adds it as normal.

8.1.13 Resource

- **Responsibility**

This is an abstract class that defines the behaviour for generic resources that can be found throughout the asteroid map. It is a parent class to the specific resource classes, and holds resource specific attributes like radioactivity. It has an association with the Inventory class.

- **Superclasses**

There is no Superclass.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**
  - **+total: int** - This is a public variable that defines the amount of resource within an Asteroid
  - **+radioactive: Bool -** This is a public boolean variable that defines whether a resource is radioactive or not. It is set to false as default, since non-radioactive resources are the majority
  - **+type: String -** This is a public string variable that defines the type of the resource. Not defined as default so that the inheriting resource classes can define it for themselves.

- **Methods**
  - **+String TypeOfMineral()**: This is a public class that returns the type of a resource as a string variable.

8.1.14 WaterIce

- **Responsibility**

This class defines a resource that can go under sublimation at the perihelion. This class is derived from the abstract class Resource, so it has all the attributes of the base class.

- **Superclasses**

This class inherits the abstract class Resource.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

This class inherits all the attributes from the Resource class, and additionally:

- **+sublime: Bool** - This public boolean variable determines whether the waterice has gone under sublimation at the perihelion or not.

- **Methods**

This class inherits all the methods from the Resource class, and it's own constructor method. Within its constructor method, it sets the resource type to WaterIce.

8.1.15 Uranium

- **Responsibility**

This class defines a resource that can explode at the perihelion. This class is derived from the abstract class Resource, so it has all the attributes of the base class. Uranium class is instantiated within the game loop, and allocated to asteroids for the settlers to be mined. They can be used to craft various items such as robots, teleportation gates, and more importantly, space stations.

- **Superclasses**

This class inherits the abstract class Resource.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

This class inherits all the attributes from the Resource class.

- **Methods**

This class inherits all the methods from the Resource class, and it's own constructor method. Within its constructor method, it sets the resource type to Uranium, and sets the radioactivity variable to true.

8.1.16 Carbon

- **Responsibility**

This class is responsible for defining the behavior for the Carbon class; which is derived from the abstract class Resource. Carbon class is instantiated within the game loop, and allocated to asteroids for the settlers to be mined. They can be used to craft various items such as robots, teleportation gates, and more importantly, space stations.

- **Superclasses**

This class inherits the abstract class Resource.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

This class inherits all the attributes from the Resource class.

- **Methods**

This class inherits all the methods from the Resource class, and it's own constructor method. Within its constructor method, it sets the resource type to Carbon.

8.1.17 Iron

- **Responsibility**

This class is responsible for defining the behavior for the Iron class; which is derived from the abstract class Resource. Iron class is instantiated within the game loop, and allocated to asteroids for the settlers to be mined. They can be used to craft various items such as robots, teleportation gates, and more importantly, space stations.

- **Superclasses**

This class inherits the abstract class Resource.

- **Interfaces**

There are no interfaces implemented in this class.

- **Attributes**

This class inherits all the attributes from the Resource class.

- **Methods**

This class inherits all the methods from the Resource class, and it's own constructor method. Within its constructor method, it sets the resource type to Iron.

8.2 Detailed plan of testing

8.2.1 Player creation

- **Description**
  Player can choose to start the game after selecting the number of settlers. Once the game is started, the system generates asteroids in the asteroid belt.

- **Unit of functionality to be tested, possible failures**
  Functionality is tested by correct initialization of the class Settler.
  Failure: Wrong input was entered. Try again.

- **Input**
  Add Settler<2>

- **Output**
  Successfully created <2> settlers.

8.2.2 Mining a resource

- **Description**
  The user is asked to mine the resources from the asteroid. The mining is only successful if the asteroid has been drilled to its core.

- **Unit of functionality to be tested, possible failures**
  Function: Mine
  Failure: The asteroid hasn't been drilled.
  Failure: Sorry, a robot can not mine, please choose a settler.

- **Input**
  Mine
- **Output**
  Do you want to Mine because your Robot cannot mine? Yes or No?
- **Input**
  'Y/y' or 'N/n'
- **Output**
  Resource was mined successfully.

8.2.3 Crafting a robot

- **Description**
  If the player has enough resources he/she can create/build a robot to help him out in drilling. The crafting is only successful if the settler has enough resources.

- **Unit of functionality to be tested, possible failures**
  Functionality is tested if the Robot object is created successfully.

Function: BuildRobot
Class: Settler
Failure: Sorry you do not have enough resources.

- **Input**
  Build Robot< >

- **Output**
  The robot has been built.

8.2.4 Crafting a teleportation gate

- **Description**

If the player has enough resources he/she can create/build a teleportation gate. The crafting is only successful if the settler has enough resources.

- **Unit of functionality to be tested, possible failures**
  Functions: BuildTeleportationGate
  Class: Settler
  Failure: Sorry you do not have enough resources.

- **Input**
  Build Teleportation Gate<>

- **Output**
  The teleportation gate has been built.

8.2.5 Building a space station

- **Description**

If the player has enough resources he/she can create/build the space station and win the game. The crafting is only successful if the settler has enough resources.

- **Unit of functionality to be tested, possible failures**
  Functions: BuildSpaceStation
  Class: Settler, SpaceStation
  Failure: Sorry you do not have enough resources.

- **Input**
  Build Teleportation Gate<>

- **Output**
  The teleportation gate has been built.

8.2.6 Activating a pair of teleportation gate

- **Description**

If player wants to use teleportation gate for traveling, he must activate the pair of the teleportation gate before he can use it. But before that he needs to make sure that pair of teleportation gates has been built and deployed successfully.

- **Unit of functionality to be tested, possible failures**
  Functionality to be tested is to check whether the gates are deployed or not.
  Classes: Asteroid
  Failure: Teleportation gates are not activated.
- **Input**
  Teleportation<>
- **Output**
  Teleportation gates are activated successfully

8.2.7 Inactivate teleportation gate

- **Description**

When player has reached to a specified destination(Asteroid) and want o get out of the teleportation, he needs to inactivate the gates.

- **Unit of functionality to be tested, possible failures**
- **Input**
  Teleportation<>
- **Output**
  The gates have inactivated

8.2.8 Sun storm deals damage to a traveller

- **Description**
  When sunstorm occurs both the Robot and Settler get impacted by sunstorm.

Eventually resulting in killing the Settler but not the Robot rather Robot gets a health damage.

- **Unit of functionality to be tested, possible failures**
  Functions: collideWith(Settler s), collidWith(Robot r)
  Class: Settler, Robot, SunStorm

- **Input**
  <Timer Machine Showing>
  Timer is showing to the Settler,when this sunstorm is coming(Timing Machine)

- **Output**
  Within the time limits Settlers can survive and perform any required operations

8.2.9 Perihelion explosion

- **Description**

If the radioactive Asteroid is in perihelion state then the explosion occurs. Settler can be killed due to explosion but robot survive.

- **Unit of functionality to be tested, possible failures**
  Functions: explode(Settler s), explode(Robot r)
  Class: Settler, Robot, RadioactiveAsteriod
  Failure: Sorry you do not have enough resources.
- **Input**
  checkTheStateOfAsteroid<>

- **Output**
  Not Perihelion than you are safe to go.

8.2.10 Drop resource to a hollow asteroid

- **Description**

After mining the resources, settler has to collect the resources on an asteroid which must be hollow. So that he can further use these resources for further tasks.

- **Unit of functionality to be tested, possible failures**
  Functionality to be tested is to check the state of the asteriod.
- **Input**
  checkHollow<>
- **Output**
  resources deposited.

8.2.11 Hiding in asteroid

- **Description**

If the mantle has been drilled through a settler can hide in the core of that asteroid to be able to survive a sun storm.

- **Unit of functionality to be tested, possible failures**

Functionality to be tested is to check if the asteroid is well drilled before the settler can hide.

- **Input**
  depth<>
- **Output**
  The depth is decreased

8.2.12 All lives are lost

- **Description**
  If all the settlers die, the players lose.
- **Unit of functionality to be tested, possible failures**
  Functionality is tested by the number of settlers that are left.
- **Input**
  Settler Number
- **Output**
  Settler is added

8.2.13 Performing operations

- **Description**
  There are several operations that can be done by a settler
- **Unit of functionality to be tested, possible failures**
  The test is to check if only one operation is performed at a single move. For example mine
- **Input**

The user is asked to mine the resources from the asteroid.

- **Output**
  Resources are added

8.2.14 Checking the core of an Asteroid

- **Description**
  The core of an asteroid is always homogeneous; it is always made of a single kind of a material.
- **Unit of functionality to be tested, possible failures**
  The test of the kind of resources found in the Asteroid. For example Uranium class.
- **Input**
  The user is asked whether he/she wants the asteroid being instantiated
- **Output**
  Asteroid is created

8.2.15 Mining Uranium

- **Description**
  Uranium is found at the core of the asteroid. Uranium is highly radioactive.
- **Unit of functionality to be tested, possible failures**
  getMined
- **Input**
  radioactivePhase and checkHollow
- **Output**
  Yes, it is minable

8.3 Plans of the supporting programs

So far, we have developed a utility program which is responsible for instantiating the Asteroid Map and Asteroids and other elements included like resources and their quantities, locations, hollow and radioactive states etc. It allows us to run different test cases and check the printed output on the console.

For the future, we have planned to use JUnit testing for our application. It will allow us to validate code results and states for state testing, and can also execute sequence of events for behavior testing. These unit tests will target a small unit of code only, so we may decide in the future if we need to text external dependencies by replacing the current dependency with test implementation or some object generated by an external test framework. These decisions will take place later during the development of our project.

8.4 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
8.04.2021 18:00	4 hours	Team	Meeting: discussion of the task assigned, discussing new requirements, structure of the system, and future implementation details. Wali Ullah 8.1 & 8.2 + 8.3 Ibrar Shakoor 8.1 & 8.2+ 8.3 Ashiq 8.1 & 8.2 Asad Idrees 8.1 & 8.2 Orgil, Vincent 8.1 & 8.2
9.04.2021 08:00	3.0 hours	Ibrar Shakoor	Completed 8.1.1-8.1.6 ,8.2.1-8.2.5 Completed:8.3 Completed: Documentation
9.04.2021 015:00	3.0 hours	Wali Ullah	Completed 8.1.1-8.1.6 ,8.2.1-8.2.5 Completed:8.3
10.04.2021 08:00	2.0 hours	Asad Idrees Razak	Completed 8.1.7 - 8.1.11, 8.2.6 - 8.2.10
10.04.2021 08:00	2.0 hours	Muhammad Ashiq	Completed 8.1.7 - 8.1.11, 8.2.6 - 8.2.10
10.04.2021 08:00	2.0 hours	Orgil Luis Macha, Igbineweka Vincent	Completed 8.1.12 - 8.1.17, 8.2.11 - 8.2.15

10. Prototype program

10.1 Deployment guide

10.1.1 List of files

File name	Size	Date	Content
Game.java	7KB	18-04-2021	Contains the “Game” class with its attributes and methods.
Settler.java	6KB	18-04-2021	Contains the “Settler” class with its attributes and methods.
Robot.java	1KB	18-04-2021	Contains the “Robot” class with its attributes and methods.
TravelMethod.java	1KB	18-04-2021	Contains the “TravelMethod” class with its attributes and methods.
Timer.java	1KB	18-04-2021	Contains the “Timer” class with its attributes and methods.
Asteroid.java	4KB	18-04-2021	Contains the “Asteroid” class with its attributes and methods.
AsteroidMap.java	7KB	18-04-2021	Contains the “AsteroidMap” class with its attributes and methods.
RadioactiveAsteroid.java	1KB	18-04-2021	Contains the “RadioactiveAsteroid” class with its attributes and methods.
SunStorm.java	1KB	18-04-2021	Contains the “SunStorm” class with its attributes and methods.
Resource.java	1KB	18-04-2021	Contains the abstract “Resource” class with its attributes and methods.
Carbon.java	1KB	18-04-2021	Contains the “Carbon” class with its attributes and methods.
Iron.java	1KB	18-04-2021	Contains the “Iron” class with its attributes and methods.
Uranium.java	1KB	18-04-2021	Contains the “Uranium” class with its attributes and methods.

WaterIce.java	1KB	18-04-2021	Contains the “WaterIce” class with its attributes and methods.
Inventory.java	1KB	18-04-2021	Contains the “Inventory” class with its attributes and methods.
SpaceStation.java	1KB	18-04-2021	Contains the “`SpaceStation” class with its attributes and methods.
Location	1KB	18-04-2021	Contains the “`Location” class with its attributes and methods.

10.1.2 Compilation

The main requirements for successful compilation are as following:

- Operating System: Windows
- IDE: IntelliJ IDEA
- JDK 8 or higher

10.1.3 Run

In order to run the application, follow the steps listed below:

1. Open the project files in Intellij IDEA, and navigate to “Game” class.

2. Navigate to the main method “public static void main(String []args)” and run using the shortcut: Shift + F10.
   Alternatively, you can press the green run button on the left side of the code. Its right next to the line numbers.

3. You will be presented with a welcome screen in console, and a choice to either start the game or exit. Follow the on-screen instructions to continue the game.

10.2 Test protocols

10.2.1 Player Creation

Name of the tester	Ibrar Shakoor
Date & time of test	18-04-2021, 10:00

10.2.2 Mining a Resource

Name of the tester	Ibrar Shakoor
Date & time of test	18-04-2021, 10:10

Name of the tester	Ibrar Shakoor
Date & time of test	18-04-2021, 10:20
Result (failure)	Could not mine resources from the asteroid.
Possible causes	Asteroid was hollow
Correction	Tested again using a non-hollow asteroid. Mined successfully.

10.2.3 Building a Robot

Name of the tester	Ibrar Shakoor
Date & time of test	18-04-2021, 10:30

Name of the tester	Ibrar Shakoor
Date & time of test	18-04-2021, 10:40
Result (failure)	Could not craft a robot successfully.
Possible causes	Lack of resources to build a robot.
Correction	Added extra resources to inventory.

10.2.4 Building a Teleportation Gate

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 10:10

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 10:25
Result (failure)	Could not craft teleportation gates successfully.
Possible causes	Lack of resources to build a teleportation gate.
Correction	Added extra resources to inventory.

10.2.5 Building a Space Station

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 10:35

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 10:40
Result (failure)	Could not craft a space station successfully.
Possible causes	Settler is at a non-hollow asteroid.
Correction	No correction, since this is an expected result.

10.2.6 Activating a pair of Teleportation Gate

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 10:50

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 10:55
Result (failure)	The pair counterpart gate has not been deployed.
Possible causes	Settler still has the other pair of the gate in his inventory.
Correction	No correction, since this is an expected result.

10.2.7 Inactivate Teleportation Gate

Name of the tester	Ashiq Muhammad
Date & time of test	18-04-2021, 11:25

10.2.8 Sun-Storm damages a Traveller

Name of the tester	Wali Ullah
Date & time of test	18-04-2021, 15:00

Name of the tester	Wali Ullah
Date & time of test	18-04-2021, 15:05

10.2.9 Radioactive Perihelion Explosion

Name of the tester	Wali Ullah
Date & time of test	18-04-2021, 15:25

Name of the tester	Wali Ullah
Date & time of test	18-04-2021, 15:35
Result (failure)	Settler has lost a life.
Possible causes	Sunstorm damages the settler.
Correction	No correction needed since it’s an expected outcome.

10.2.10 Drop resources in a Hollow Asteroid

Name of the tester	Wali Ullah
Date & time of test	18-04-2021, 15:40

Name of the tester	Wali Ullah
Date & time of test	18-04-2021, 15:50

10.2.11 Hiding in Hollow Asteroid

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:10

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:20
Result (failure)	Unable to hide in asteroid
Possible causes	Asteroid was not drilled all the way through
Correction	Tested again on an asteroid with 0 depth. Worked as expected.

10.2.12 All Lives Are Lost

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:20

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:25
Result (failure)	Game still running
Possible causes	Logic error in code.
Correction	Debugged the code, worked as expected.

10.2.13 Setting Radioactive Cores of Asteroids

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:30

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:40

10.2.14 Mining Uranium

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 12:50

Name of the tester	Asad Idrees Razak
Date & time of test	18-04-2021, 13:00

10.2.15 Creating Asteroids in Asteroid Belt

Name of the tester	Igbineweka Vincent
Date & time of test	18-04-2021, 13:15

Name of the tester	Igbineweka Vincent
Date & time of test	18-04-2021, 13:25
Result (failure)	Asteroid could not be added.
Possible causes	Mismatched parameter list.
Correction	Passed the correct parameters.

10.2.16 Travelling to a different Asteroid

Name of the tester	Igbineweka Vincent
Date & time of test	18-04-2021, 13:35

Name of the tester	Igbineweka Vincent
Date & time of test	18-04-2021, 13:40
Result (failure)	Settler has travelled from A0 to A1.
Possible causes	Expected.
Correction	None.

10.2.17 Drilling

Name of the tester	Igbineweka Vincent
Date & time of test	18-04-2021, 13:45

Name of the tester	Igbineweka Vincent
Date & time of test	18-04-2021, 13:50

10.2.18 Mining

Name of the tester	Orgil Luis Macha
Date & time of test	18-04-2021, 20:00

Name of the tester	Orgil Luis Macha
Date & time of test	18-04-2021, 20:10
Result (failure)	Carbon added to inventory
Possible causes	Expected result.
Correction	None.

10.2.19 Water Sublimation

Name of the tester	Orgil Luis Macha
Date & time of test	18-04-2021, 20:20

Name of the tester	Orgil Luis Macha
Date & time of test	18-04-2021, 20:30
Result (failure)	Asteroid previously containing WaterIce is hollow after perihelion contact.
Possible causes	Expected result.
Correction	None.

10.2.20 End Game

Name of the tester	Orgil Luis Macha
Date & time of test	18-04-2021, 20:40

Name of the tester	Orgil Luis Macha
Date & time of test	18-04-2021, 20:50
Result (failure)	Game state has been set to END
Possible causes	Expected result.
Correction	None.

10.3 Evaluation

Name of the team member	Participation (%)
Wali Ullah	16.66%
Ibrar Shakoor	16.66%
Idrees Razak	16.66%
Ashiq Muhammad	16.66%
Macha Orgil Luis	16.66%
Igbineweka Vincent Ifeanyi	16.66%

10.4 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
14.04.2021 12:15	1 Hour	Team	Meeting: Discussion and Brainstorming about the next tasks. Ibrar and Wali are responsible for: Game, Settler, Robot, Timer, TravelMethod Idrees and Ashiq are responsible for: Asteroid, AsteroidMap, RadioactiveAsteroid, SunStorm, Resource Luis and Vincent are responsible for: Location, Inventory, SpaceStation, Iron, WaterIce, Carbon, Uranium
15.04.2021 18:00	1+3	Wali Ullah Ibrar Shakoor	Implementation of Classes, Attributes, and Methods for: Game, Settler, Robot, Timer, TravelMethod
16.04.2021 10:00	1	Wali Ullah Ibrar Shakoor	Updating implemented methods + adding more comments
17.04.2021 12:15	0.5	Wali Ullah Ibrar Shakoor	Completed Artifact 10.1.1: List of Files
17.04.2021 13:15	0.5	Wali Ullah Ibrar Shakoor	Completed Artifact 10.1.2: Compilation
17.04.2021 12:15	0.5	Orgil Vincent	Completed Artifact 10.1.3: Run
17.04.2021 12:15	4	Wali Ullah Ibrar Shakoor Orgil Vincent Ashiq Muhammad Idrees Razak	Completed Artifact 10.2: Test Protocols

18.04.2021 09:15	3	Ashiq Muhammad Idrees Razak	Implementation of Classes: Asteroid, AsteroidMap, RadioactiveAsteroid, SunStorm, Resource
18.04.2021 12:15	1	Ashiq Muhammad Idrees Razak	Review, updating and adding comments
14.04.2021 12:15	1	Orgil Luis Macha Vincent	Implementation of Classes: Resource, Inventory, Iron, Uranium, WaterIce, Carbon

NR#11 User Interface Specification

4 – Six Stars

Supervisor:
Dr. Balla Katalin

Members:

Name	Neptun	Email
Ashiq Muhammad	EDOXAM	muhammadashiqse10@gmail.com
Ibrar Shakoor	FS4J6L	ibrarraja11222@gmai.com
Wali Ullah	WTW2SV	wali6985@gmail.com
Asad Idrees Razak	XJJUK4	idreesshaikh6300@gmail.com
Igbineweka Vincent	TOM9S6	ifyvinz@gmail.com
Orgil Luis Macha	VH0JMU	orgilluismacha@gmail.com

11. User interface specification

11.1 Graphical User Interface

Figure 1. Title screen.

Player will see this screen when the game starts, and clicking the LAUNCH button will start the game. After pressing the launch button a graphical representation of an Asteroid will going to present to the user.

Figure 2. Drilling an asteroid.

Player can see the progress of the drilling action from the depth indicator on its right. On the bottom left of the screen, the inventory slots are visible. Current load: 2 Water, 1 Uranium, 1 Carbon, 1 Iron. All the shapes which are on the left bottom of the above photo are representing the necesary resources on the screen.

Figure 3. Mining resources

Once fully drilled through, a pathway to the core will be visible.This is also a graphical representation for the user that he/she has Mined through the core an Asteroid.

Figure 4. Hiding in a hollow asteroid.

After the successful mining, Settler leaves the ship and hides inside the hollow asteroid. On the left side of the operations button,map indicates the location of the settler in the asteroid belt. We can also keep track of the Settler activity on the Asteroid with the help of this Graphical Map representation.

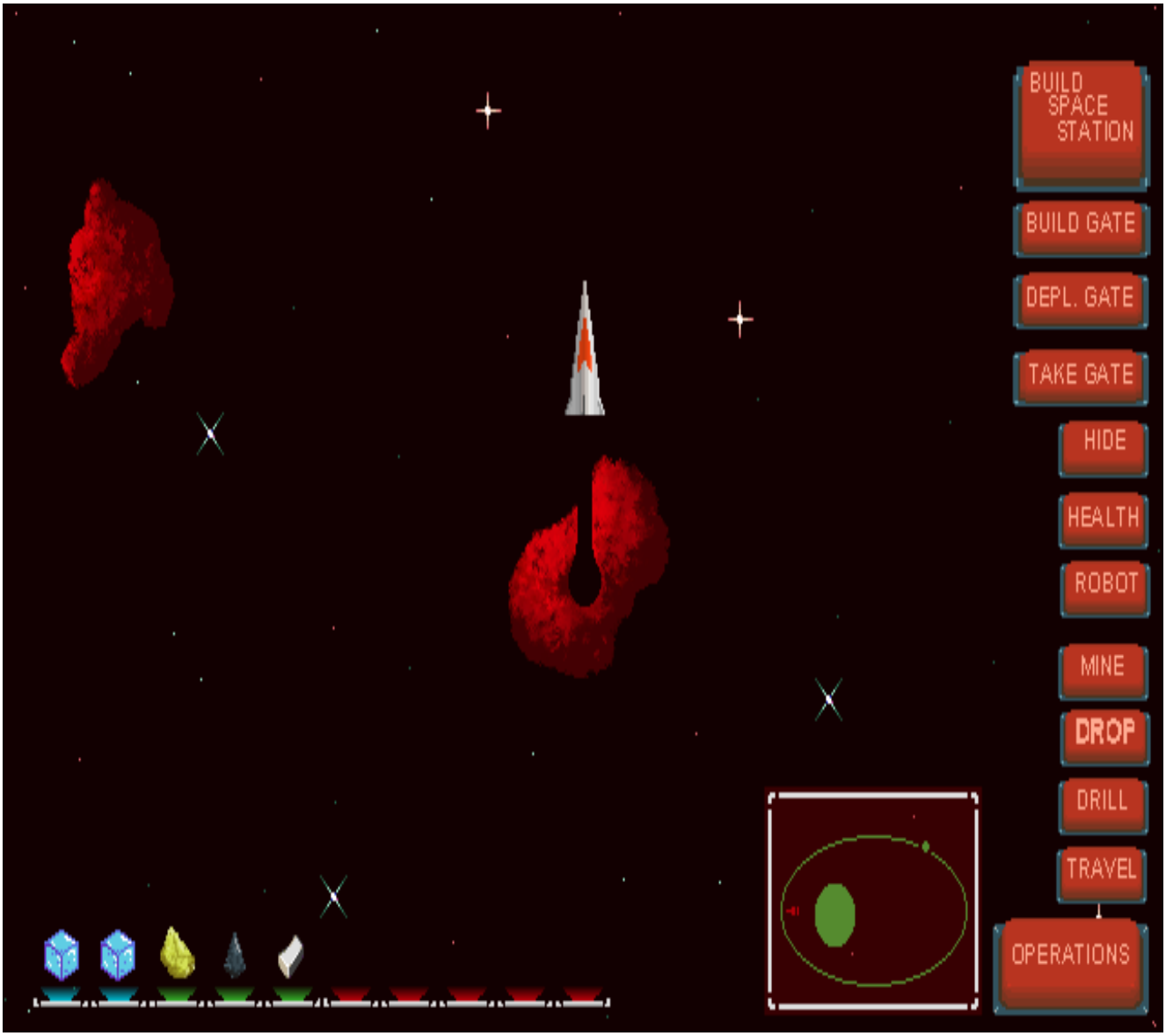


Figure 5. Expanded OPERATIONS menu.

When the player clicks on the operations button, it will expand a collapsible list of buttons. And accoring to the need he/she can chose any operation towards their goal.Clicking besides these operation buttons will going to popup menu saying that you haven't chose the right operation.

11.2 Architecture of the graphical system

11.2.1 Principles of the GUI

Model:

The Model contains only the pure application data, it contains no logic describing how to present the data to a user.

In our program, the model represents the following classes:

- Game
- Settler
- Robot
- Travel Method
- Time
- Asteroid
- Asteroid Map
- Radioactive Asteroid
- SunStorm
- Resource
- Carbon
- Iron
- WaterIce
- Uranium
- Inventory
- Space Station
- Location

Some of these classes act as a part of the controller as well since they implement some part of the behavior such as, Settler class uses travel( ), mine( ), and drill( ) functions and SunStorm class has a damageDone( ) method.

Controller:

The Controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events.

The Controller part is implemented in the Game class, this is the class where all the game logic is stored and realized. It has important methods which are responsible for populating the asteroids, determining the amount of resources, the conditions to win/lose the game etc.

View:

The View presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.

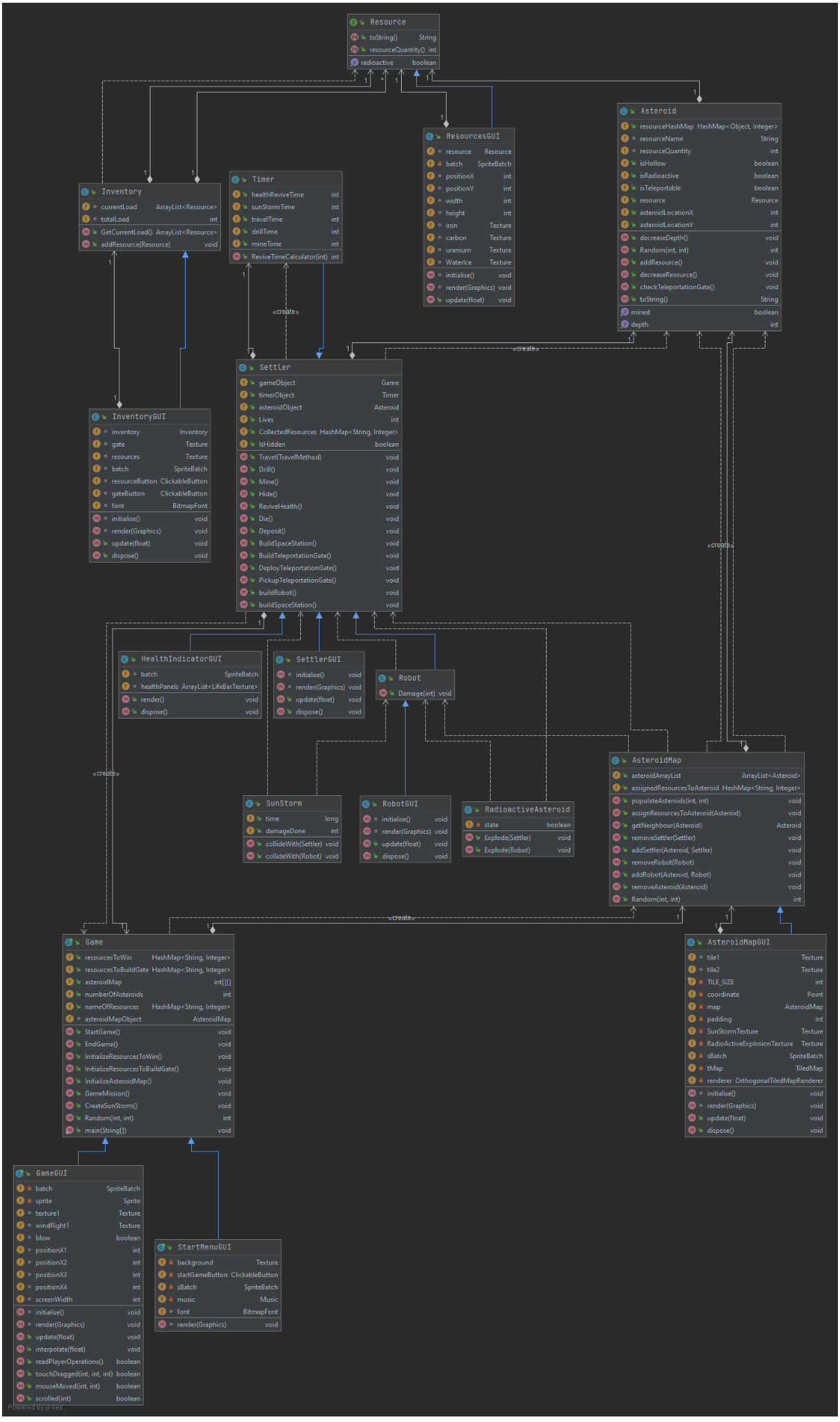
The View will be implemented by our GUI classes. We introduce supplementary classes for the view section like SettlerGUI, RobotGUI and InventoryGUI. These classes are designed similarly, for example, SettlerGUI it will store an attribute- settler and will implement the functions for rendering the settler behavior, while RobotGUI will do the same but for robot behavior.

Our final program will ensure the Push model: the user input like mouse click events would be handled in the Game class. It will take the input and process based on the current state of the Model after which, the GUI would be notified to change its state. This notification would be done by calling the required method from the GUI classes.

Example: the player presses the OPERATIONS button by Left Mouse Click -> the Game takes the input and calls the ShowOperationsMenu from GameUI class. The ShowOperationsMenu in GameUI updates the view by showing the list of all operations available to the player.

11.2.2 GUI Structure diagram

Note: All GUI Classes and their dependencies are shown in the diagram below. To keep the size small, and ensure visibility, we have chosen to omit unnecessary classes such as Iron, Carbon, WaterIce, Uranium etc.


Resource
toString() String
resourceQuantity() int
radioactive boolean
Asteroid
resourceHashMap HashMap<Object, Integer>
resourceName String
resourceQuantity int
isHollow boolean
isRadioactive boolean
isTeleportable boolean
resource Resource
asteroidLocationX int
asteroidLocationY int
decreaseDepth() void
Random(int, int) int
addResource() void
decreaseResource() void
checkTeleportationGate() void
toString() String
mined boolean
depth int
ResourcesGUI
resource Resource
batch SpriteBatch
positionX int
positionY int
width int
height int
iron Texture
carbon Texture
uranium Texture
WaterIce Texture
initialise() void
render(Graphics) void
update(float) void
Inventory
currentLoad ArrayList<Resource>
totalLoad int
GetCurrentLoad() ArrayList<Resource>
addResource(Resource) void
Timer
healthReviveTime int
sunStormTime int
travelTime int
drillTime int
mineTime int
ReviveTimeCalculator(int) int
«create»
Settler
gameObject Game
timerObject Timer
asteroidObject Asteroid
Lives int
CollectedResources HashMap<String, Integer>
isHidden boolean
Travel(TravelMethod) void
Drill() void
Mine() void
Hide() void
ReviveHealth() void
Die() void
Deposit() void
BuildSpaceStation() void
BuildTeleportationGate() void
DeployTeleportationGate() void
PickupTeleportationGate() void
buildRobot() void
buildSpaceStation() void
InventoryGUI
inventory Inventory
gate Texture
resources Texture
batch SpriteBatch
resourceButton ClickableButton
gateButton ClickableButton
font BitmapFont
initialise() void
render(Graphics) void
update(float) void
dispose() void
HealthIndicatorGUI
batch SpriteBatch
healthPanels ArrayList<LifeBarTexture>
render() void
dispose() void
SettlerGUI
initialise() void
render(Graphics) void
update(float) void
dispose() void
Robot
Damage(int) void
«create»
SunStorm
time long
damageDone int
collideWith(Settler) void
collideWith(Robot) void
RobotGUI
initialise() void
render(Graphics) void
update(float) void
dispose() void
RadioactiveAsteroid
state boolean
Explode(Settler) void
Explode(Robot) void
AsteroidMap
asteroidArrayList ArrayList<Asteroid>
assignedResourcesToAsteroid HashMap<String, Integer>
populateAsteroids(int, int) void
assignResourcesToAsteroid(Asteroid) void
getNeighbour(Asteroid) Asteroid
removeSettler(Settler) void
addSettler(Asteroid, Settler) void
removeRobot(Robot) void
addRobot(Asteroid, Robot) void
removeAsteroid(Asteroid) void
Random(int, int) int
Game
resourcesToWin HashMap<String, Integer>
resourcesToBuildGate HashMap<String, Integer>
asteroidMap int[][]
numberOfAsteroids int
nameOfResources HashMap<String, Integer>
asteroidMapObject AsteroidMap
StartGame() void
EndGame() void
InitializeResourcesToWin() void
InitializeResourcesToBuildGate() void
InitializeAsteroidMap() void
GameMission() void
CreateSunStorm() void
Random(int, int) int
main(String[]) void
AsteroidMapGUI
tile1 Texture
tile2 Texture
TILE_SIZE int
coordinate Point
map AsteroidMap
padding int
SunStormTexture Texture
RadioActiveExplosionTexture Texture
sBatch SpriteBatch
tMap TiledMap
renderer OrthogonalTiledMapRenderer
initialise() void
render(Graphics) void
update(float) void
dispose() void
GameGUI
batch SpriteBatch
sprite Sprite
texture1 Texture
windRight1 Texture
blow boolean
positionX1 int
positionX2 int
positionX3 int
positionX4 int
screenWidth int
initialise() void
render(Graphics) void
update(float) void
interpolate(float) void
readPlayerOperations() boolean
touchDragged(int, int, int) boolean
mouseMoved(int, int) boolean
scrolled(int) boolean
StartMenuGUI
background Texture
startGameButton ClickableButton
sBatch SpriteBatch
music Music
font BitmapFont
render(Graphics) void

11.3 GUI Classes

11.3.1 GameGUI

- **Responsibility**

The responsibility of this class is to define the view of the game according to the screen's width and height. This is where the game starts and prints out the introductory screen. The game class controls every other class because it's the backbone and also the one that triggers the start of the game.

- **Superclasses**

There is no super

- **Interfaces**

There is no interface to this class

- **Attributes**
  - **-batch**: **SpriteBatch-** used for the drawing text strings and sprites
  - **-sprite**: **Sprite-** It represent the image of different elements in the game like asteroid, settler, robot etc
  - **+texture1: Texture-**The texture (image) that will be used for this item representation
  - **+windRight1: Texture-** textures that display the wind from the left and from the right
  - **+positionX1: int-** This is the position place holder coordinate
  - **+positionX2: int-** another position placeholder coordinate
  - **+positionX3: int-** another position placeholder coordinate regarded as X3
  - **+positionX4: int-** another position placeholder coordinate regarded as X4
  - **+screenWidth: int-** Width of the screen
  - **+blow: boolean -** variable that saves the state and helps to reset the position of sprites to their initial position

- **Methods**
  - **+void initialise()**: Method called to initialize the map tiles with its textures
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void update(delta: float):** Method called every time there happens any event regarding the players
  - **+void interpolate(alpha: float):**
  - **+boolean readPlayerOperations():** Check weather player perform any operation or not

11.3.2 AsteroidMapGUI

- **Responsibility**

This shows the view of the asteroid map onto the screen and the respective arrangement of asteroids in the asteroid belt which is completely random. However, it prints out every asteroid texture and the appearance of sunstorm texture too.

- **Superclasses**

There is no super class

- **Interfaces**

There is no interface to this class

- **Attributes**
  - **+tile1**: **Texture-** texture (image) of tile1 in the 2D grid
  - **+tile2**: **Texture -** texture (image) of tile1 in the 2D grid
  - **-tileSize: int-** Defines the size of each tile in pixels
  - **-coordinate: Point-** Represent the specific spot on the pixel
  - **-map: AsteroidMap-** Map that must be represented in view
  - **-padding: int -** it is the space from the borders making the ui more aesthetically pleasing
  - **-sunStormTexture: Texture-** Defines the sun storm represented by its texture
  - **-radioActiveExplosionTexture: Texture-** Defines the explosion(happen in the radioactive asteroid) represented by its texture
  - **-sBatch: SpriteBatch-** used for the drawing
  - **-tMap: TiledMap-** Shows the whole representation of tiles in the image
  - **-renderer: OrthogonalTiledMapRenderer-** for rendering purpose of the tiles

- **Methods**
  - **+void initialise()**: Method called to initialize the map tiles with its textures
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void update(delta: float):** Method called every time there happens any event regarding the players
  - **+void dispose():** Dispose of the batch

11.3.3 InventoryGUI

- **Responsibility**

To display the number of items in the inventory and the goal (collected resources and gate). The inventory is rendered on the side of the screen so the player can quickly glance at the item available and know his collected amount. This class creates ease as it will give visual representation to the resources a player already has.

- **Superclasses**

There is no super class

- **Interfaces**

There is no interface to this class

- **Attributes**
  - **+inventory**: **Inventory-** Inventory which contains all the items
  - **+gate**: **Texture-** Teleportation gate represented by its texture
  - **+resources: Texture-** Each resource represented by specific texture
  - **+batch: SpriteBatch-** used for drawing purpose
  - **+resourceButton: ClickableButton-** Button to control the resources on the screen
  - **+gateButton: ClickableButton-** Button to control the teleportation gate on the screen

- **Methods**
  - **+void initialise()**: Method called to initialize the map tiles with its textures
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void update(delta: float):** Method called every time there happens any event regarding the players
  - **+void dispose():** Dispose of the batch

11.3.4 SettlerGUI

- **Responsibility**

Represent the settler or number of settlers printed onto the screen. This class renders and prints every Settler texture there is in the asteroid belt. A settler may be hidden in an asteroid core however he/she is still visible due to this class.

- **Superclasses**

There is no super class

- **Interfaces**

There is no interface to this class

- **Attributes**

- **Methods**
  - **+void initialise()**: Method called to initialize the map tiles with its textures
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void update(delta: float):** Method called every time there happens any event regarding the players
  - **+void dispose():** Dispose of the batch
  -

11.3.5 RobotGUI

- **Responsibility**

It represents a Robot or number of Robot sprites printed on the screen. A robot is very similar in terms of class UI design/responsibility ;however, it cannot mine therefore it won't be able to hide inside the asteroid core. This class renders and prints every robot texture there is in the asteroid belt.

- **Superclasses**

There is no super class

- **Interfaces**

There is no interface to this class

- **Attributes**

- **Methods**
  - **+void initialise()**: Method called to initialize the map tiles with its textures
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void update(delta: float):** Method called every time there happens any event regarding the players
  - **+void dispose():** Dispose of the batch

11.3.6 ResourcesGUI

- **Responsibility**

It displays the amount of resources onto the screen where the setter or robot can find them. The resources are widespread randomly in the asteroid cores. It also displays the resources a settler or robot has at a current time as well as the texture of resources itself. Resources are the collected items so each one is randomly distributes and rendered

- **Superclasses**

There is no super class

- **Interfaces**

There is no interface to this class

- **Attributes**
  - **-batch**: **SpriteBatch -** used for the drawing
  - **+positionX**: **int-** the position (x coordinate) where the resources are placed.
  - **+positionY: int-** the position (y coordinate) where the resources are placed.
  - **+width: int -** the width of resources onto the screen (the dimension of resources)
  - **+height: int-** the height of resources onto the screen (the dimension of resources)
  - **+iron: Texture -** one of the unique resource image of Iron
  - **+carbon: Texture -** one of the unique resource image of Carbon
  - **+uranium: Texture -** one of the unique resource image of Uranium
  - **+waterIce: Texture -** one of the unique resource image of WaterIce

- **Methods**
  - **+void initialise()**: Method called to initialize the map tiles with its textures
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void update(delta: float):** Method called every time there happens any event regarding the players
  - **+void dispose():** Dispose of the batch

11.3.7 StartMenuGUI

- **Responsibility**

This represents the very initial screen in the beginning of the game with multiple options provided in the form of buttons. The choices selected there triggers the start of the game. The game can be exited outrightly or played continuously. The player makes the choice here.

- **Superclasses**

There are no superclasses.

- **Interfaces**

There are no interfaces associated with this class.

- **Attributes**
  - **-background**: **Texture-** The graphical texture provided onto the screen in the background
  - **-startGameButton**: **ClickableButton-** It's a clickable button to trigger the start of the game
  - **-sBatch: SpriteBatch** for the drawing
  - **-music: Music-** The music being played in the Main Menu (start screen)
  - **+font: BitMapFont-** The textual font used to change the look and feel of text onto the screen.

- **Methods**
  - **+void render(g: org.mini2Dx.core.graphics.Graphics):** Method called by the game loop from the application every time rendering of the map should be performed

11.3.8 HealthIndicatorGUI

- **Responsibility**

The responsibility of this class is to show the health (life) of the Settler played by the player. The textures used are the heart icons representing the health primarily of Settlers and robots as well. As the game goes the health textures has a size change presented as the change in remaining life of the player.

- **Superclasses**

There is no super

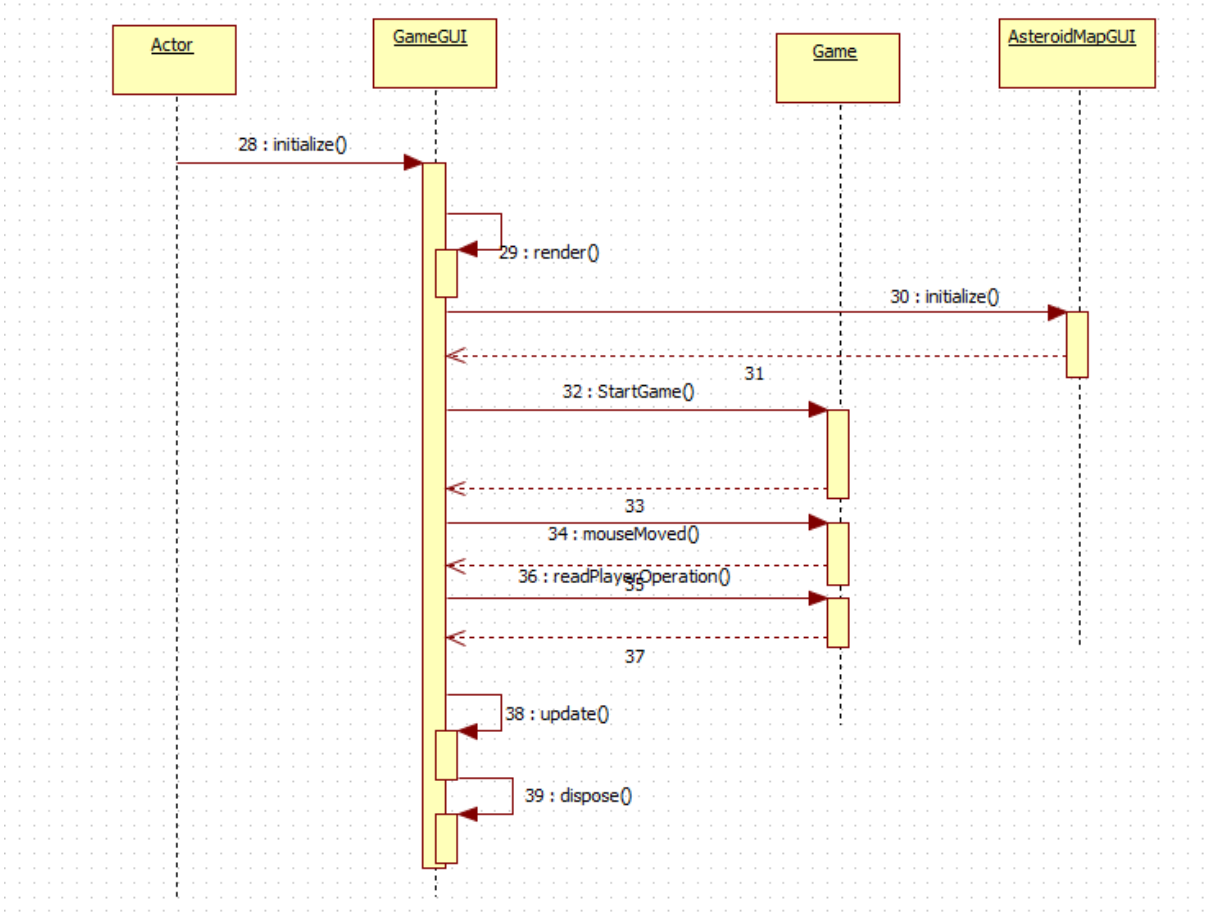
- **Interfaces**

There are no interfaces to this class.

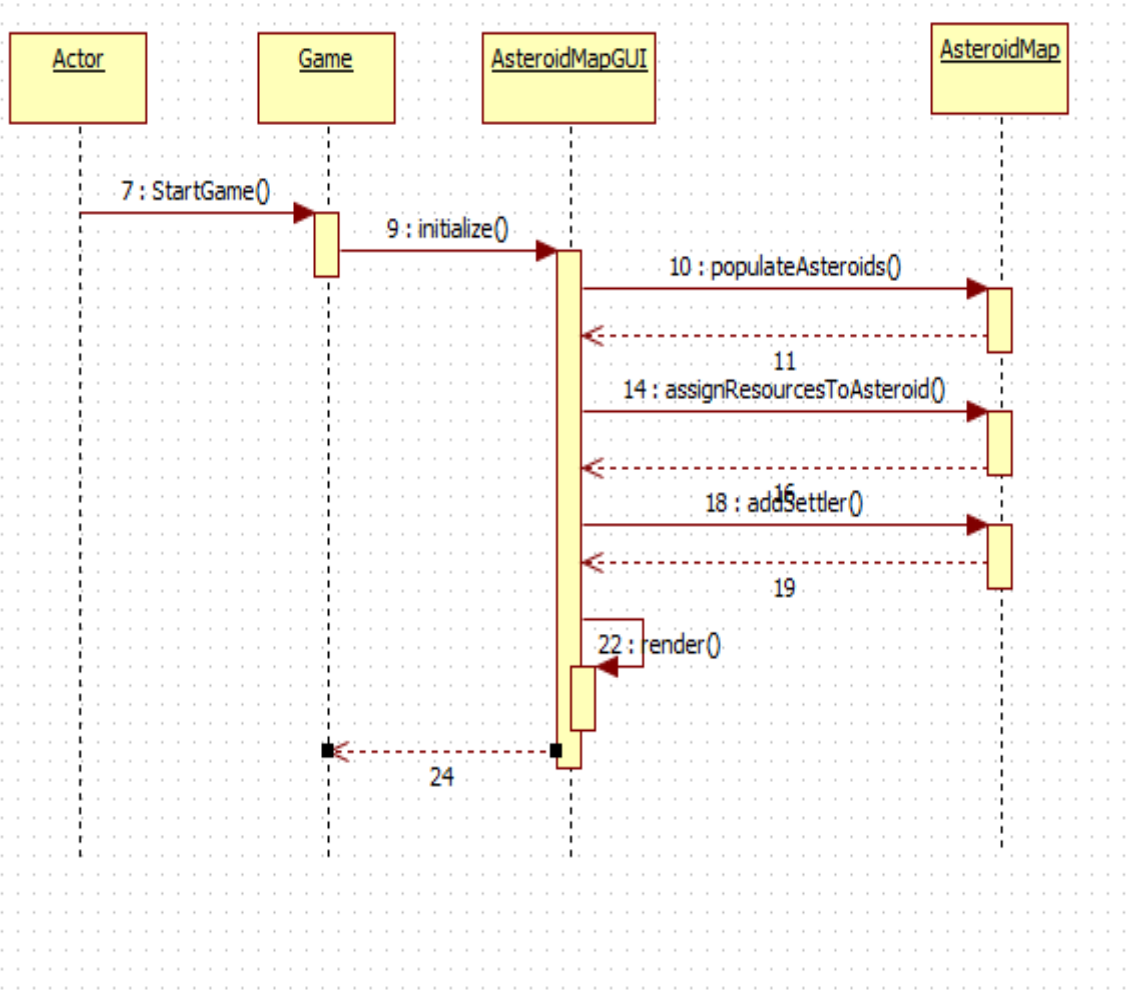
- **Attributes**
  - **+healthPanels**: **ArrayList<LifeBarTexture> -** arraylist of the textures (image) that will be used for the life representation
  - **+batch**: **SpriteBatch -** for the drawing

- **Methods**
  - **+void render(g: Graphics)**: Method called by the game loop from the application every time rendering of the map should be performed
  - **+void dispose():** Dispose of the batch

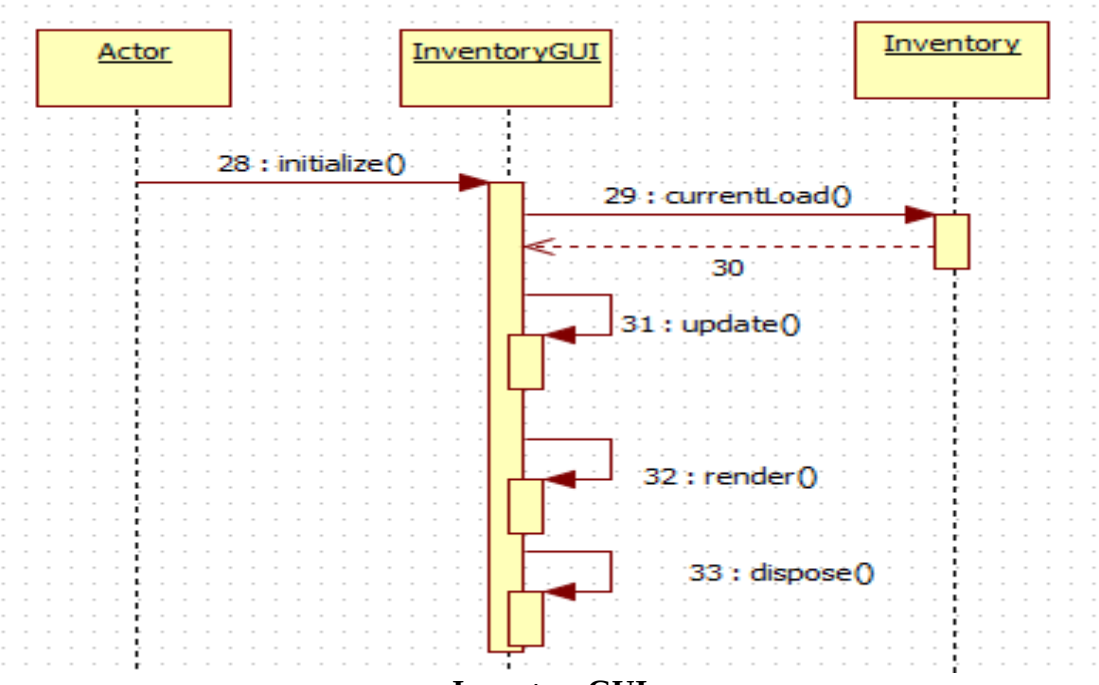
11.4 Dynamic connection between the model and the GUI



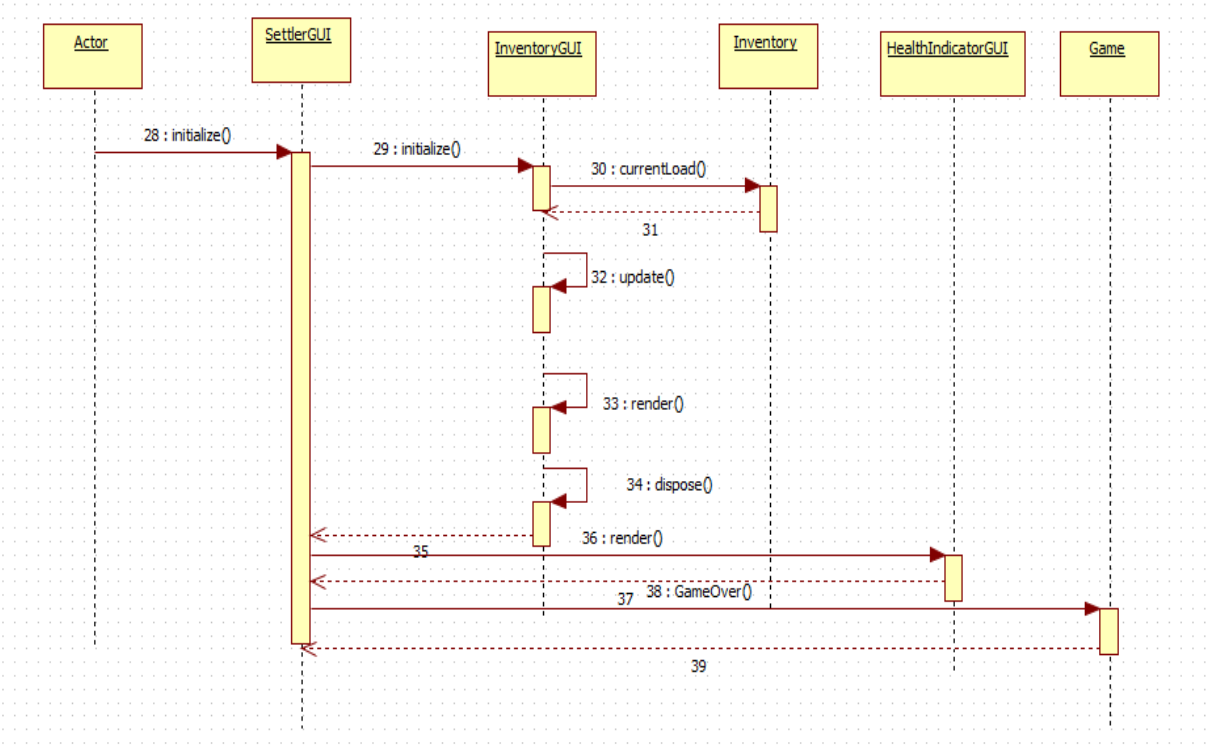
GameGUI Rendering



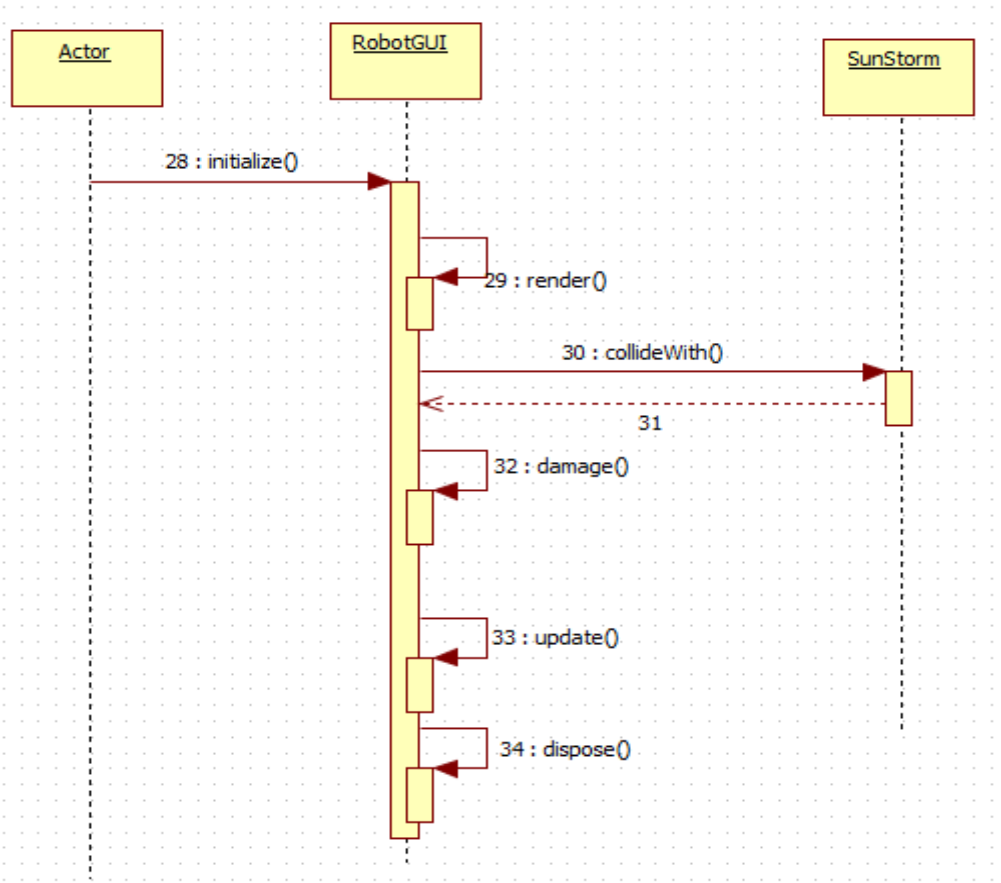
AsteroidMapGUI initialization



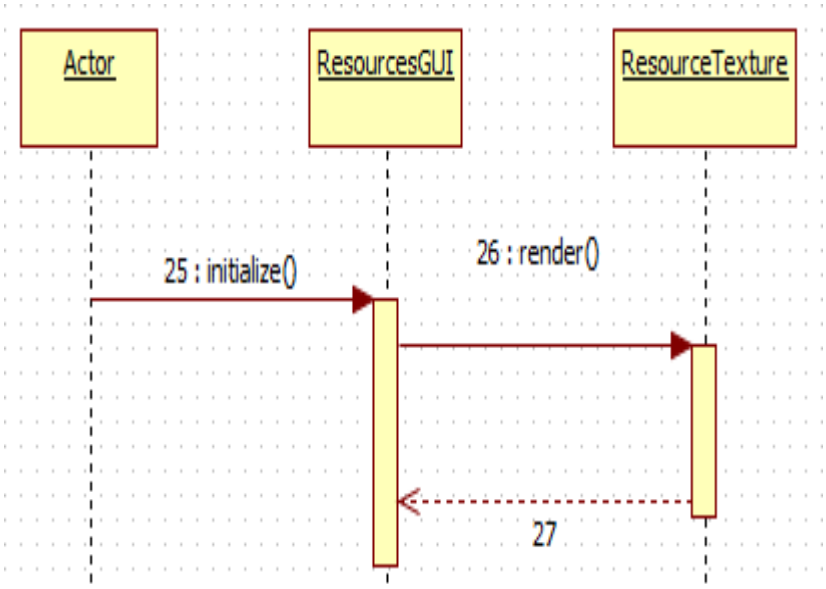
InventoryGUI



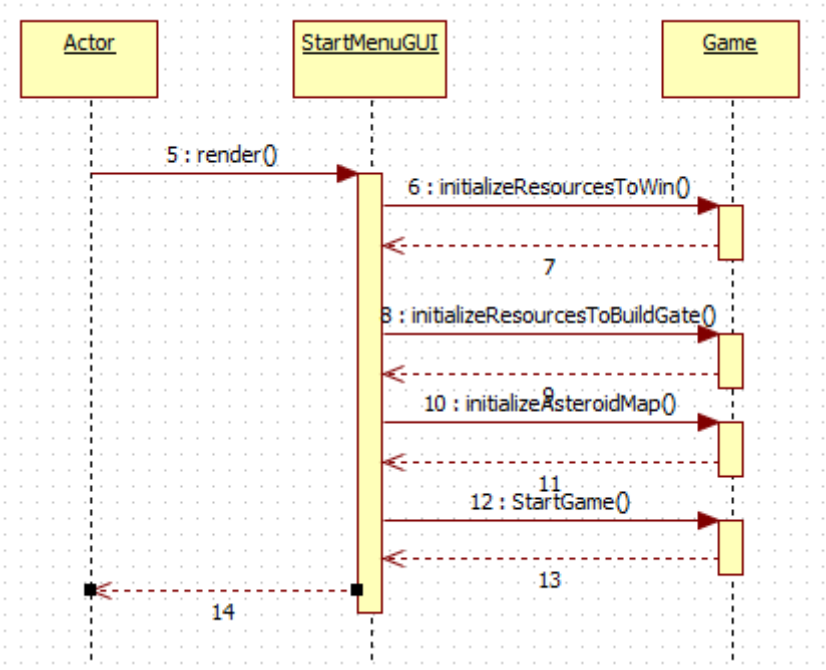
SettlerGUI



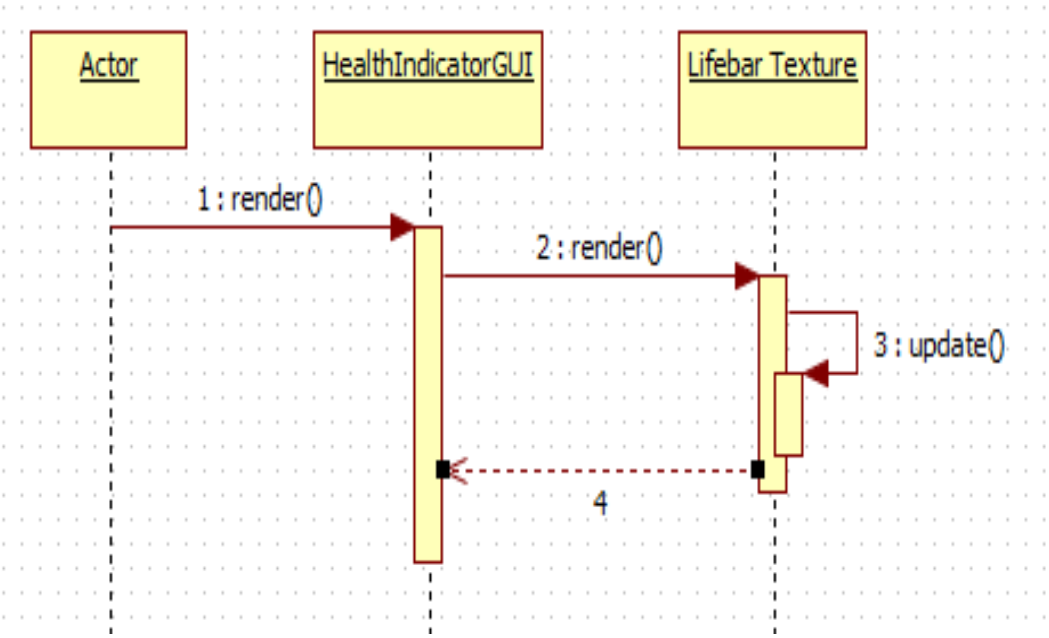
RobotGUI



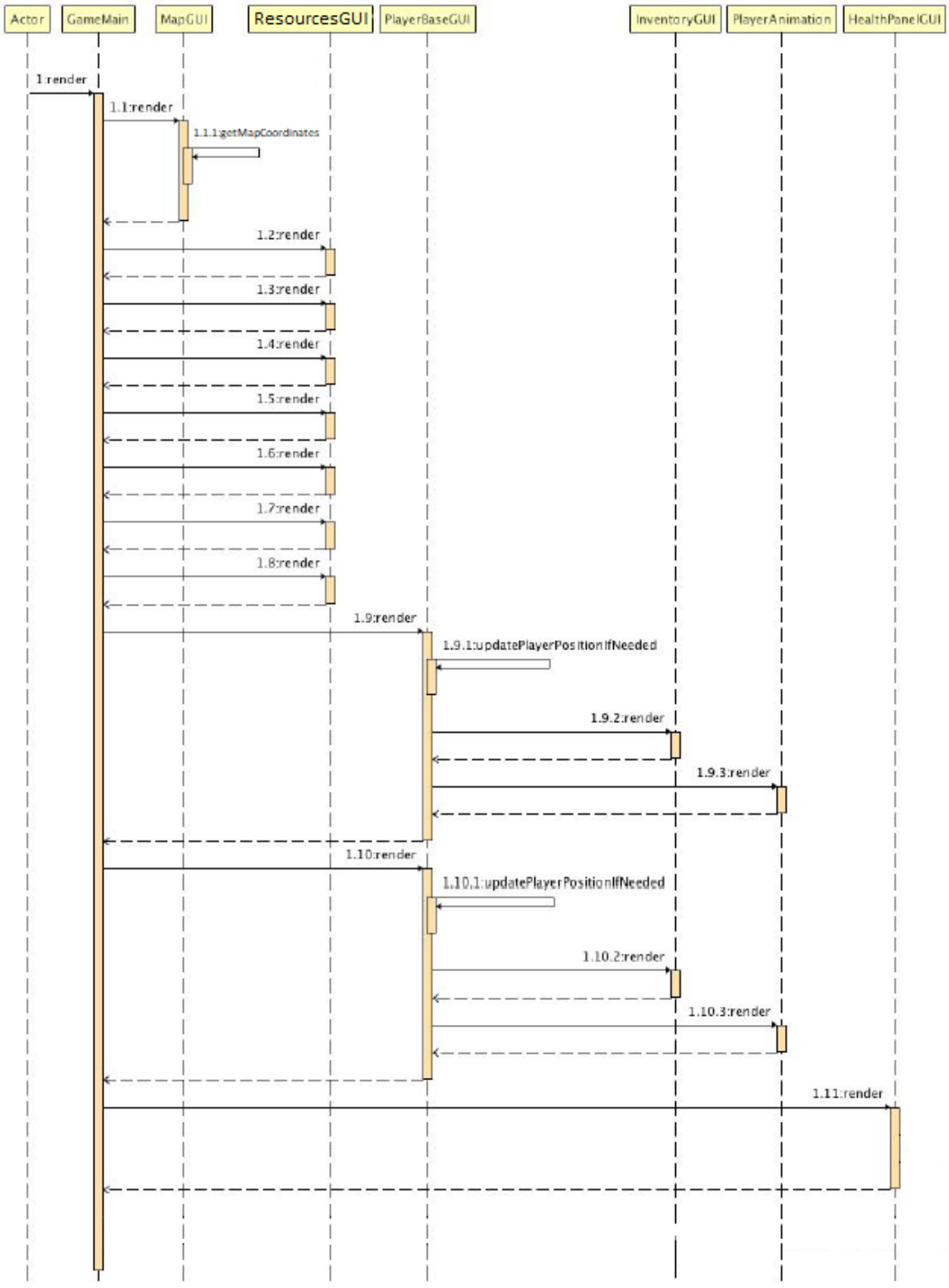
Rendering ResourceGUI Texture



StartMenuGUI Initialization



Health PanelGUI



General GUI Renderi

11.5 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
22.04.2021 12:15	1 Hour	Team	Meeting: Discussion and Brainstorming about the upcoming tasks. Writing down the GUI classes. Integrating the GUI classes into the current code. Making of the GUI sprites and textures Dividing the writing of User Interface Specification Doc Tasks
24.04.2021 09:00	2 hours	Asad Idrees Razak	Completed Artifact 11.3
24.04.2021 09:00	2 hours	Muhammad Ashiq	Completed Artifact 11.3
24.04.2021 09:00	2 hours	Wali Ullah	Completed Artifact 11.2
24.04.2021 09:00	2 hours	Ibrar Shakoor	Completed Artifact 11.2 and 11.4 and Documentation
23.04.2021 09:00	3 hours	Macha Orgil Luis	Completed Artifact 11.1 and 11.4
23.04.2021 09:00	2 hours	Vincent	Completed Artifact 11.4

13. Complete program

13.1 Deployment guide

13.1.1 List of files

File name	Size	Date	Content
Game.java	7KB	13-05-2021	Contains the “Game” class with its attributes and methods.
Settler.java	6KB	13-05-2021	Contains the “Settler” class with its attributes and methods.
Robot.java	1KB	13-05-2021	Contains the “Robot” class with its attributes and methods.
TravelMethod.java	1KB	13-05-2021	Contains the “TravelMethod” class with its attributes and methods.
Timer.java	1KB	13-05-2021	Contains the “Timer” class with its attributes and methods.
Asteroid.java	4KB	13-05-2021	Contains the “Asteroid” class with its attributes and methods.
AsteroidMap.java	7KB	13-05-2021	Contains the “AsteroidMap” class with its attributes and methods.
RadioactiveAsteroid.java	1KB	13-05-2021	Contains the “RadioactiveAsteroid” class with its attributes and methods.
SunStorm.java	1KB	13-05-2021	Contains the “SunStorm” class with its attributes and methods.
Resource.java	1KB	13-05-2021	Contains the abstract “Resource” class with its attributes and methods.
Carbon.java	1KB	13-05-2021	Contains the “Carbon” class with its attributes and methods.
Iron.java	1KB	13-05-2021	Contains the “Iron” class with its attributes and methods.
Uranium.java	1KB	13-05-2021	Contains the “Uranium” class with its attributes and methods.

WaterIce.java	1KB	13-05-2021	Contains the “WaterIce” class with its attributes and methods.
Inventory.java	1KB	13-05-2021	Contains the “Inventory” class with its attributes and methods.
SpaceStation.java	1KB	13-05-2021	Contains the “`SpaceStation” class with its attributes and methods.
Location	1KB	13-05-2021	Contains the “`Location” class with its attributes and methods.
TitlescreenGUI.java	3KB	13-05-2021	Handles the textures, and event listeners for the title screen.
AsteroidMapGUI.java	2KB	13-05-2021	Handles the textures, and event listeners for the asteroid map.
InventoryGUI.java	2KB	13-05-2021	Handles the textures, and event listeners for the inventory HUD
TravelllerGUI.java	3KB	13-05-2021	Handles the textures, and event listeners for the travellers
DesktopLauncher.java	1KB	13-05-2021	Application window configuration.

13.1.2 Compilation

The main requirements for successful compilation are as following:

- Operating System: Windows
- IDE: IntelliJ IDEA
- JDK 8 or higher

13.1.3 Run

In order to be able to run the executable program, please follow the steps carefully:

- Set the configuration in your IntelliJ IDEA: DesktopLauncer in GameMain
- Class path of module: AsteroidMining.desktop.main
- Project Structure > Modules > Assets > Mark as “Resources”
- Make sure your gradle is setup properly for project build

Once you have successfully completed all these steps, you can build the project and run it with DesktopLauncer config that you set up earlier. After you press the start button, the game will start.

13.2 Evaluation

Name of the team member	Participation (%)
Wali Ullah	16.66%
Ibrar Shakoor	16.66%
Idrees Razak	16.66%
Ashiq Muhammad	16.66%
Macha Orgil Luis	16.66%
Igbineweka Vincent Ifeanyi	16.66%

13.3 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description
11.05.2021 16.00	3	Team	Meeting: Discussion and Brainstorming about the next tasks, and finalizing last week's work.
13.05.2021 12.00	1	Wali	Completed Artifact 13.1.1: List of Files
13.05.2021 13.00	1	Ibrar	Completed Artifact 13.1.2: Compilation
13.05.2021 15.00	1	Asad and Ashiq	Completed Artifact 13.1.3: Run
13.05.2021 20.00	1	Luis	Modified Artifact 13.1.1: List of Files Added GUI files with modified sizes
13.05.2021 22.00	0.5	Wali	Completed Artifact 13.2: Evaluation