● Detaillierte Pläne

8.1. Entwürfe von Klassen und Methoden

8.1.1 AIRobot

- **Verantwortung**

  Der AIRobot bohrt, bewegt sich und hilft den Spielern, ihr Ziel zu erreichen.

- **Basisklasse**

  Entity → AIRobot

8.1.2 AppController

- **Verantwortung**

  Diese Abteilung ist für die Ausführung des Spiels verantwortlich. Startet, stoppt oder pausiert das Spiel.

- **Methoden**
  - **+main()**: Ein Programm muss eine globale Funktion namens main enthalten, die den festgelegten Start des Programms darstellt.
  - **-startGame():** Wird aufgerufen, wenn das Spiel beginnt.
  - **-quitGame():** Wird aufgerufen, wenn das Spiel beendet ist.

8.1.3 Asteroid

- **Verantwortung**

  Der Asteroid speichert Rohmaterial in seinem Kern. Er kann Astronauten schützen, wenn sie sich in seinem Kern verstecken. Das Asteroid kann explodieren. Das Rohmaterial des Kerns kann abgebaut werden..

- **Ősosztályok**

  SteppableSpaceObject → Asteroid

- **Schnittstellen / Interface**
  - EventObservable
  - Observable

- **Attribute**
  - **String name**: Der Name des Asteroiden.
  - **boolean closeToSun**: Ist der Asteroid in der Nähe des Sonne.

- **Methoden**
  - noch überschreibende Methoden von der abstrakten Basisklasse (SteppableSpaceObject)
  - **explode():** innere Method um Explosion zu behandeln

8.1.4 HomeAsteroid

- **Verantwortung**

  Diese Asteroid speichert gleichzeitig mehrere Rohmaterial in seinem Kern. Er kann Astronauten schützen, wenn sie sich in seinem Kern verstecken. Dieser Asteroid kann nicht explodieren, er hat andere Verhalten aus diese Sichtpunkt.

- **Basisklassen**

  SteppableSpaceObject, Asteroid → HomeAsteroid

- **Schnittstellen / Interface**
  - EventObservable
  - Observable

- **Attribute**
  - **String name**: Der Name des Asteroiden. (“Home”)
  - **boolean closeToSun**: Ist der Asteroid in der Nähe des Sonne.

- **Methoden**
  - noch überschreibende Methoden von der abstrakten Basisklasse (SteppableSpaceObject)
  - **explode():** innere Method um Explosion zu behandeln.

8.1.5 AsteroidZone

- **Verantwortung**

  Verwaltet das Spielfeld.

- **Attribute**
  - **double defOfCloseToSun**: Wenn Sie diese Zahl mit einem SteppableSpaceObject vergleichen, können Sie feststellen, ob es sich in der Nähe der Sonne befindet

- **Methoden**
  - **+createZone()**: Es schafft das Spielfeld vor dem Spielstart.
  - **+addSpaceObject(spaceObj: SteppableSpaceObject)**: Fügt der Asteroidenzone einen SteppableSpaceObject hinzu.
  - **+removeSpaceObject(onPosition: Position)**: Entfernt ein Objekt in einer Asteroidenzone von einer bestimmten Position.
  - **+iterator getIterOnSpaceObjects()**: Wenn ich die Nachbarn auflisten möchte, gibt diese Funktion eine Liste aller SpaceObjects und ich kann basierend darauf weiter filtern.
  - **+Position generateRandomPosition()**: Erstellt Positionen durch Generieren von Zufallszahlen. Erforderlich, um die Strecke zu bauen.
  - **-boolean checkDistanceAtCreate(toCheckPosition: Position):** Hier wird geprüft, ob um eine bestimmte Position herum genügend Platz vorhanden ist, um ein anderes SpaceObject abzulegen.
  - **+SteppableSpaceObject findHome():** Das Haus kehrt mit einem Asteroiden zurück.
  - **+Ressource generateRandomRessource():** Er kehrt mit einem zufälligen Rohstoff zurück.

8.1.6 Core

- **Verantwortung**

  Es ist verantwortlich für die Speicherung der Rohstoffe innerhalb des Asteroiden

- **Attribute**
  - **int capacity**: Gibt an, wie viele Rohstoffe sich im Kern befinden können.

- **Methoden**
  - **Resource[1…*] popResource**: Es kehrt mit dem Rohmaterial des Kerns zurück.
  - **pushResource(newResource: Resource)**: Setzt den Kernrohstoff auf eine bestimmte Ressource
  - **+getCoreInfo():** Gibt den Namen des Rohmaterials zurück.

8.1.7 Entity

- **Verantwortung**

  Abstrakte Basisklasse für Roboter und Siedler.

- **Schnittstellen / Interface**
  - Observer → Entity

- **Attribute**
  - **-String name**: Name des Entity.

- **Methoden**
  - **+*move()***: Durch Aufrufen dieser Funktion werden die Entitäten verschoben.
  - **+*drill()*:** Durch Aufrufen dieser Funktion werden die Entitäten bohren.
  - **+*die()*:** Durch Aufrufen dieser Funktion werden Entitäten beendet.
  - **+SteppableSpaceObject[] listMyNeighbours()**: Gibt die Nachbarn des Asteroiden zurück, auf dem sich die Entität befindet.
  - **+SteppableSpaceObject getMySpaceObject():**
  - **+doAction():**
  - **#setMySpaceObject(newOnPlace: SteppableSpaceObject):**
  - **#SteppableSpaceObject chooseNeighbour(neighbours: SteppableSpacObject[]):**

8.1.8 EventObservable

- **Verantwortung**

  Wenn ein Asteroid explodiert oder eine Sonneneruption gibt, benachrichtigt SteppableSpaceObject.

- **Methoden**
  - **notifyAboutDanger():** Informiert alle aufgeschriebene Entität über die Gefahr von Solarflair.
  - **notifyAboutDieEvent():** Informiert alle aufgeschriebene Entität über den Tod, und das Grund darauf.

8.1.9 GameController

- **Verantwortung**

  Verwaltet den Verlauf des Spiels. Vom Start zum Ende.

- **Attribute**
  - **-boolean gameIsRunning**: Sagt dir, ob das Spiel läuft.
  - **-int currentRound**: Die Nummer der aktuellen Runde des Spiels.
  - **+int playersNum**: Anzahl der Spieler. Etwas, das während des Setups konfiguriert werden kann.
  - **-int settlerNum**: Anzahl der Siedler. Etwas, das während des Setups konfiguriert werden kann.

- **Methoden**
  - **-removePlayer(name: String)**: Spieler aus dem Spiel entfernen.
  - **leaveGame(playerLeaving: Player)**: Wenn ein Spieler aufhören möchte, gibt er auf, tötet im Wesentlichen alle seine Siedler und löscht sie auch aus den Spielern.
  - **+int getRound():** Gibt das Anzahl der aktuelle Runde im Spiel zurück.
  - **+evaluateRound():** Wertet den gegebenen Rund aus. Überprüfen Sie, ob alle Siedler des Spielers gestorben sind, und ob die Spieler gewonnen oder verloren haben.
  - **+addPlayer(newPlayer: Player):** Fügt dem Spiel einen neuen Spieler hinzu.
  - **+inGame():** Eine fast unendliche Schleife, in dieser Funktion läuft das Spiel und das Spiel kreist als Endlosschleife. Wir rufen diese Funktion auf, indem wir das Spiel starten und am Ende des Spiels beenden.
  - **+boolean removeBot(bot: AIRobot):** Entfernt den Roboter. Der Boolesche Wert gibt an, ob Sie ihn entfernt haben.
  - **+addBot(bot: AIRobot):** Fügt dem Spiel einen neuen AIRoboter hinzu.
  - **-dropSettlers():** Macht einen Siedler für den Spieler.
  - **-setUpGame():** Rufen Sie die Funktionen auf, mit denen das Spiel eingestellt wird.
  - **+iter getterOnPlayer():** Kehrt mit den Spielern zurück, die den ganzen Weg iteriert wurden.
  - **-removePlayer(name: String):** Entfernt einen bestimmten Spieler von den Spielern.
  - **-evaluateFlair():** Bewertet, dass es diesen Sonnenwind im gegebenen Kreis gibt.
  - **-round():**Diese Funktion geht abwechselnd durch die Spieler und Siedler, wenn sie etwas tun.
  - **-createAndNamePlayers():** Legt den Namen des Spielers fest.

4.3.9 Gate

- **Verantwortung**

  Kann von Siedlern erstellt und gespeichert werden. Ermöglicht das Reisen zwischen einem Paar von ihnen.

- **Basisklasse**

  SteppableSpaceObject → Gate

- **Attribute**
  - **Gate gatePair:** Der Paar eines Portals.

- **Methoden**

  nur überschreibende Methoden von der abstrakten Basisklasse.

8.1.10 Layer

- **Verantwortung**

  Es stellt die Kortikalis/Kruste/Layer des Asteroiden dar, weiß, wie dick er ist und kann ihn reduzieren.

- **Attribute**
  - **int thickness**: Die Dicke der Kruste des Asteroiden.

- **Methoden**
  - **int thinIt()**: Es verdünnt sich. Der DrillLayer ruft Sie an.
  - **int getThickness()**: Der Asteroid kehrt mit der Dicke seiner Kortikalis zurück.

8.1.11 Player

- **Verantwortung**

  Hilft bei der Verwaltung der Entitäten eines Spielers, repräsentiert ein Spieler. (nicht ein Astronaut/Settler)

- **Attribute**
  - **-String name:** Der Name des Spielers.
  - **-Settler[1..n] mySettlers:** Die Kollektion der Siedler, die zu einem Spieler gehören.

- **Methoden**
  - **+Iterator getIterOnMyEntities()**: Erzeugt einen Iterator für die Kollektion der Siedler eines Spielers
  - **+addEntity(newEntity: Entity):** Weist einem Spieler eine neu Entität zu.
  - **+removeEntity(removedEntity: Entity):** Hebt die Zuordnung einer Entität zu einem Spieler auf
  - **+setName(name: String)**: Setter für Name.
  - **+String getName():** Getter für Name.
  - **+killPlayer()**: Töten/eliminiert ein Spieler mit ihre Settlern aus dem Spiel.

4.3.12 Position

- **Verantwortung**

  Ordnet Objekten im Feld eine bestimmte Position zu.

- **Schnittstellen / Interface**

  Comparable

- **Attribute**
  - **double x:** Koordinate auf der x-Achse.
  - **double y:** Koordinate auf der y-Achse
  - **double ownRadius:** Hilft überlappende Objekte zu vermeiden.

- **Methoden**
  - **+double getX():** Gibt die x-Koordinate zurück.
  - **+double getY():** Gibt die y-Koordinate zurück.
  - **+double getRadius():** Gibt den Radius des zugehörenden Objekts zurück.
  - **+double distanceFrom(pos: Position):** Kalkuliert die Distanz zweier Positionen.
  - **+double[2] getLeftUpperCornerCoordinate():** Hilft mit dem korrekten Platzieren der Objekte.
  - **+double getMinimalNeighbourDistance():** Untere Schranke für die Zufallsdistanz, in der die Objekte benachbart sind.
  - **+double getMaximalNeighbourDistance():** Obere Schranke…
  - **+setRadius(newRadius: double):** Legt den Radius der Position fest.
  - **+boolean equals(Object o:):** Sagt, dass die Positionen der beiden Objekte gleich e sind.

8.1.13 Settler

- **Verantwortung**

  Siedler sind die Entitäten, die die Spieler bewegen und über sie mit dem Spielfeld interagieren können.

- **Basisklasse**

  Entity → Settler

- **Attribute**
  - **Resources[0...10] resources:** Speichert die abgebauten Ressourcen eines Siedlers.
  - **Player owner:** Der Spieler, dem die Siedler gehören.
  - **Gate[0...2] createdGates:** Die Kollektion der erstellten Portale.

- **Methoden**
  - **+createBot():** Erschafft einen AI Roboter.
  - **+mine():** Entfernt die Ressource aus dem Asteroidenkern, und fügt es seiner eigenen Kollektion von Ressourcen zu.
  - **+createGate():** Erstellt ein Paar Portale, und fügt sie seiner eigenen Kollektion von Portalen zu.
  - **+deployResource():** Setzt eine Ressource wieder in einen Asteroiden ein.
  - **+buildGate():** Platziert eines der getragenen Portale.
  - **+listResources():** Listet die Ressourcen auf, die der Siedler besitzt.
  - **+Resource chooseResource():** Der Siedler wählt eine Ressource aus seinem Inventar aus.
  - **-addResource(resource: Resource):** Fügt Ressource zu der Kollektion .
  - **-waitingSettler():** Der Siedler tut nichts und wartet auf die nächste Runde.

8.1.14 SteppableSpaceObject

- **Verantwortung**

  Diese sind Objekte, auf die die Spieler mit ihren Siedlern treten können.

- **Schnittstellen / Interface**

  Observable

- **Attribute**
  - **Entity[0…*] playersOnMe:** Speichert die Entitäten, die auf einem solchen Objekt stehen.

- **Methoden**
  - **Position getPosition():** Gibt die Position des Objektes zurück.
  - **String getName()**: Gibt den Namen des SteppableSpaceObjectes zurück, falls es existiert.
  - **boolean drillLayer()**: Falls es eine Asteroid ist, reduziert die Größe der Kortikalis des Asteroiden um eins. Übergibt die Bohraufgabe an das Layer.
  - **Ressource mineResource()**: Falls es eine Asteroid ist, baut es das Innere des Asteroiden ab und setzt die Art des Rohmaterials auf leer. Leitet die Mining-Aufgabe an das Core weiter.
  - **boolean addResourceToCore (resource: Resource)**: Übergibt die Aufgabe der Rohstoffzugabe an das Core, falls es eine solche Objekt ist, welches ein Core (Kern) hat.
  - **setMyPosition(newPosition: Position):** Setzt die Position eines Portals, falls solche Operation interpretierbar ist.
  - **boolean isActive():** Gibt an, ob ein Portal, oder Objekt aktiv ist, also ob es wirklich funktioniert.
  - **boolean setPair(pairGate: Gate):** Bindet das aktuelle Objekt mit einem Anderen zu.
  - **SteppableSpaceObject getPair():** Gibt das Paar eines Portals/SteppableSpaceObject zurück.
  - **String getInfo()**: manche Information über das Objekt. Später wird es nützlich, beim GUI.
  - **int getLayerThickness()**: Gibt die Manteldicke einer SpaceObject zurück.

8.1.15 Sun

- **Verantwortung**

  Startet Sunflairs, was gefährlich für Siedler und Roboter sind.

- **Schnittstellen / Interface**

  EventObservable

- **Attribute**
  - **Position position:** Position der Sonne auf dem Spielfeld.

- **Methoden**
  - **doSunFlair():** Jede Entität auf dem Feld wird überprüft, ob sie in Gefahr ist, und wenn ja, stirbt sie.

8.1.16 Observable

- **Verantwortung**

  Diese Schnittstelle hat Operationen, dadurch Entitäten (Siedler und Roboter) zur Kollektion von SteppableSpaceObjects hinzugefügt werden können.

- **Methoden**
  - **checkOut(leavingEntity: Entity):** Das SteppableSpaceObject entfernt die Entität von sich.
  - **checkIn(newEntity: Entity):** Das SteppableSpaceObject registriert eine ankommende Entität an sich.

8.1.17 Observer

- **Verantwortung**

  Diese Schnittstelle hat das Aufgabe, dass die Entitäten auf SolarFlair, oder Asteroidexplosion reagieren. Diese Methoden der Interface hat die Kenntnisse, wie das aktuelle Entität das Ereignisse behandelt.

- **Methoden**
  - **notifyFlairEvent():** Benachrichtigt die Entität (und Spieler auch) über einem FlairEreignis, und reagiert darauf mit Methodenaufrufe der konkrete Objekt.
  - **notifyFlairDanger():** Benachrichtigt die Entität (und Spieler auch) über einem bevorstehende FlairEreignis, und reagiert darauf mit Methodenaufrufe der konkrete Objekt. Often nur ein Message auf dem Bildschirm, über die zukünftige Solarflair.
  - **notifyAsteroidExplosion():** Benachrichtigt die Entität (und Spieler auch), dass ihre Asteroid explodiert hat, und soll sie (die Entität) dieses Ereignis irgendwie abhängig von ihrem Typ behandeln.

8.1.18 Resource

- **Verantwortung**

  Diese abstrakte Klasse wird von den anderen konkrete Ressourcen implementiert, die sich im Kern der Asteroiden (im ResourceStorage) befinden. Sie hat Operationen, dadurch die Name der Ressourcen abgefragt werden kann. Das kann man auch bestimmen, ob es radioaktiv ist.

- **Methoden**
  - **String getName():** Gibt den Namen (Typ) der Ressource zurück.
  - **boolean isRadioactive():** Sagt, ob die Ressource radioaktiv ist.
  - **boolean equals(o: Object):** Vergleicht zwei Ressourcen. Die sind gleich, wenn ihre Namen sind gleich.
  - **int compareTo():** Vergleicht zwei Ressourcen

8.1.19 ResourceStorage

- **Verantwortung**

  Eigene Klasse für einfachere Ressourcespeicherung mit einstellbare Kapazität.

  Often wird als Container referenziert in der nachkommenden Beschreibung.

- **Methoden**
  - **boolean storageIsFull():** Gibt das Containerzustand zurück, ob es voll ist. (ob man weitere Ressourcen einpacken kann)
  - **Resource popResource(res:Resource):** Probiert ein bestimmte Ressource aus dem Container ausheben.
  - **Resource popRandomResource(res:Resource):** Probiert ein zufälliges Ressource aus dem Container ausheben. Falls nur eine Ressource im Container ist, gibt diese einzige zuruck.
  - **boolean pushResource(res: Resource):** Probiert ein bestimmte Ressource im Container speichern, falls es noch nicht voll ist.
  - **List<Resource> getResourceList():** Gibt das aktuelle Inhalt der Container zurück.
  - **ResourceStorage popMore(count: int, res: Resource):** Gibt ein RessourceStorage mit gesuchte Menge der Resourcen aus dem originelle Container, falls es möglich ist.
  - **boolean pushMore(res: Resource):** Probiert die bestimmte Menge der Ressourcen im Container speichern, falls es noch möglich ist.
  - **int countOf(type: Resource):** Gibt das Ressourcemenge der angefragte Ressource aus dem Container.
  - **int getAllCapacity():** Gibt das Containerkapazität zurück.
  - **setAllCapacity(allCapacity: int):** Stellt das Kapazitätsgrenze des Containers ein.

- **Attributen**

  -resource: List<Resource>: Um Ressourcen zu speichern.

  -allCapacity: int: Obere Schrank für Listenelemente.

8.1.20 Coal

- **Verantwortung**

  Ressource vom Typ Coal.

- **Schnittstellen / Interface**

  Resource

- **Methoden**
  - **String getName():** Gibt “Coal” zurück.
  - **boolean isRadioactive():** Gibt false zurück.

8.1.21 Empty

- **Verantwortung**

  Ressource vom Typ Empty.

- **Schnittstellen / Interface**

  Resource

- **Methoden**
  - **String getName():** Gibt “Empty” zurück
  - **boolean isRadioactive():** Gibt false zurück.

8.1.22 FrozenWater

- **Verantwortung**

  Ressource vom Typ FrozenWater.

- **Schnittstellen / Interface**

  Resource

- **Methoden**
  - **String getName():** Gibt “FrozenWater” zurück
  - **boolean isRadioactive():** Gibt false zurück.

8.1.23 Iron

- **Verantwortung**

  Ressource vom Typ Iron

- **Schnittstellen / Interface**

  Resource

- **Methoden**
  - **String getName():** Gibt “Iron” zurück
  - **boolean isRadioactive():** Gibt false zurück.

8.1.24 Uran

- **Verantwortung**

  Ressource vom Typ Uran

- **Schnittstellen / Interface**

  Resource

- **Methoden**
  - **String getName():** Gibt “Uran” zurück
  - **boolean isRadioactive():** Gibt true zurück.

8.2 Die Pläne der Tests und ihre Beschreibung mit der Testsprache

8.2.1 CreateGateTest

- **Beschreibung:**
  Testen das createGate, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Teleportpaar aus seinen Ressourcen machen.
- **Inspektierte Funktionalität: CreateGate**
- **Eingabe:** createGateTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : createGateTest
--------------------------------------
GameController - setup ended : pattern found
Gate - New gate created : pattern found
Gate - New gate created : pattern found
--------------------------------------
createGateTest : PASSED
--------------------------------------
```

8.2.2 BuildGateTest

- **Beschreibung:**
  Testen das buildGate, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Teleportpaar in dem Asteroidenfeld platzieren.
- **Inspektierte Funktionalität: BuildGate**
- **Eingabe:** buildGateTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : buildGateTest
--------------------------------------
GameController - setup ended : pattern found
Gate - New gate created : pattern found
Gate - New gate created : pattern found
Settler - BuildGate called : pattern found
Settler - Gate placed at : pattern found
--------------------------------------
buildGateTest : PASSED
--------------------------------------
```

8.2.3 SunFlairTest

- **Beschreibung:**
  Testen einen SunFlair Ereignis, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Im Test wird das Settler auf einem benachbarten Asteroide bewegen, dort auf dem SunFlair warten, und endlich gestorben.
- **Inspektierte Funktionalität: SunFlair**
- **Eingabe:** sunFlairTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : sunFlairTest
---------------------------------------
GameController - setup ended : pattern found
Entity - move called : pattern found
ConsoleUI - Used automatic command: temp1 : pattern found
GameController - evaluateFlair called : pattern found
GameController - evaluateFlair called : pattern found
Settler - Die method of player testName : pattern found
---------------------------------------
sunFlairTest : PASSED
---------------------------------------
```

8.2.4 ListNeighbourTest

- **Beschreibung:**
  Testen das Auszahlfunktion, in dem wir alle benachbarte Asteroide durchschauen können. Falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll.
- **Inspektierte Funktionalität: ListNeighbour**
- **Eingabe:** listNeighboursTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : listNeighboursTest
---------------------------------------
GameController - setup ended : pattern found
Entity - listMyNeighbours called : pattern found
Entity - Possible neighbour: : pattern found
---------------------------------------
listNeighboursTest : PASSED
---------------------------------------
```

8.2.5 ExplodeTest

- **Beschreibung:**
  Testen einen Asteroidexplosion, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Asteroid völlig bohren, und das Asteroid soll im Sonnennahe mit radioaktive Material im Kern sein.
- **Inspektierte Funktionalität: Explode**
- **Eingabe:** explodeTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : explodeTest
---------------------------------------
GameController - setup ended : pattern found
Entity - move called : pattern found
ConsoleUI - Used automatic command: temp0 : pattern found
Settler - Settler tried to drill an object : pattern found
Asteroid - drillLayer called, before drill was the layer: 1 : pattern found
Asteroid - Asteroid has radioactive core _and_ it's close to Sun --> EXPLODE : pattern found
Asteroid - explode called : pattern found
Settler - Die method of player : pattern found
---------------------------------------
explodeTest : PASSED
---------------------------------------
```

8.2.6 DrillTest

- **Beschreibung**

Testen das Drill Funktion, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler bohren(drill).

- **Inspektierte Funktionalität: drill**
- **Eingabe**: drillTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : drillTest
---------------------------------------
GameController - setup ended : pattern found
Settler - Drill called : pattern found
Settler - Settler tried to drill an object : pattern found
Asteroid - drillLayer called, before drill was the layer: : pattern found
Settler - Drill was successful : pattern found
---------------------------------------
drillTest : PASSED
---------------------------------------
```

8.2.7 MineTest

- **Beschreibung**

Testen das MineFunktion, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler fördern(mine).

- **Inspektierte Funktionalität: mine**
- **Eingabe**: mineTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : mineTest
----------------------------------------
GameController - setup ended : pattern found
Settler - Mine called : pattern found
Asteroid - mineResource called : pattern found
Settler - addResource called : pattern found
Settler - Coal added to settler successfully : pattern found
----------------------------------------
mineTest : PASSED
----------------------------------------
```

8.2.8 DeployResource Test

- **Beschreibung**

Testen das deployResource, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Rohstoff zurückstecken(deployResource).

- **Inspektierte Funktionalität:** deployResource
- **Eingabe**: deployResource.json
- **Erwartete Ausgabe:**

```
EVAL TEST : mineTest
----------------------------------------
Settler - DeployResource called : pattern found
Settler - ListResources called : pattern found
Settler - chooseResource called : pattern found
Settler - Write the number of the resource you would like to choose : 1 - Coal, 2 - FrozenWater, 3 - Iron, 4 - Uran : pattern found
Settler - The selected resource can be chosen : pattern found
Asteroid - addResourceToCore called : pattern found
----------------------------------------
mineTest : PASSED
----------------------------------------
```

8.2.9 Move Test

- **Beschreibung**

Testen das move, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Rohstoff zurückstecken(move).

- **Inspektierte Funktionalität:** moveTest
- **Eingabe**: moveTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : move
-------------------------------------
SteppableSpaceObject - checkIn called : pattern found
Settler - doAction called : pattern found
Entity - move called : pattern found
Entity - listMyNeighbours called : pattern found
Entity - Possible neighbour: exampleInfo : pattern found
Settler - ChooseNeighbour called : pattern found
SteppableSpaceObject - checkOut called : pattern found
SteppableSpaceObject - checkIn called : pattern found
-------------------------------------
move : PASSED
-------------------------------------
```

8.2.10 Create Zone Test

- **Beschreibung**

Testen das createZone, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Rohstoff zurückstecken(createZone).

- **Inspektierte Funktionalität:** createZone
- **Eingabe**: createZoneTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : createZoneTest
-------------------------------------
GameController - createAndNamePlayers called : pattern found
Sun - Sun constructor called : pattern found
SteppableSpaceObject - SteppableSpaceObject constructor called : pattern found
SteppableSpaceObject - SteppableSpaceObject constructor called : pattern found
Asteroid - Asteroid constructor called : pattern found
Asteroid - Asteroid constructor called : pattern found
Sun - getPosition called : pattern found
Sun - getPosition called : pattern found
Asteroid - Asteroid constructor called : pattern found
Sun - getPosition called : pattern found
GameController - dropSettlers called : pattern found
SteppableSpaceObject - checkIn called : pattern found
Sun - checkIn called : pattern found
-------------------------------------
createZoneTest : PASSED
-------------------------------------
```

8.2.11 Create Robot Test

- **Beschreibung**

Testen das createRobot, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Rohstoff zurückstecken(createRobot).

- **Inspektierte Funktionalität:** createRobot
- **Eingabe**: createRobotTest.json
- **Erwartete Ausgabe:**

```
EVAL TEST : createRobotTest
-------------------------------------
GameController - getInstance called : pattern found
GameController - getInstance called : pattern found
SteppableSpaceObject - checkIn called : pattern found
Settler - Bot created at Home asteroid : pattern found
GameController - Iterate on robots, they do things: : pattern found
AIRobot - doAction called : pattern found
AIRobot - strat_1 called: AIRobot on strat1 : pattern found
Entity - move called : pattern found
Entity - listMyNeighbours called : pattern found
Entity - Possible neighbour: exampleInfo : pattern found
AIRobot - chooseNeighbour called : pattern found
SteppableSpaceObject - checkOut called : pattern found
SteppableSpaceObject - checkIn called : pattern found
-------------------------------------
createRobotTest : PASSED
-------------------------------------
```

8.3 Napló

Anfang	Zeitdauer	Teilnehmern	Beschreibung
2021.03.29.	2,5 óra	alle	Meeting: Aufgaben teilen, Problemen besprechen
2021.03.29.	4 óra	Pongrácz	Testvalidation implementieren
2021.03.29	1-2 Stunde	Hrotkó	Modifikationen implementieren
2021.03.30	3 Stunde	alle	Tests schreiben, Dokumentation schreiben
2021.03.29.	4 Stunde	Domokos	Ufo class, Dokumentation (Entwürfe von Klassen und Methoden)