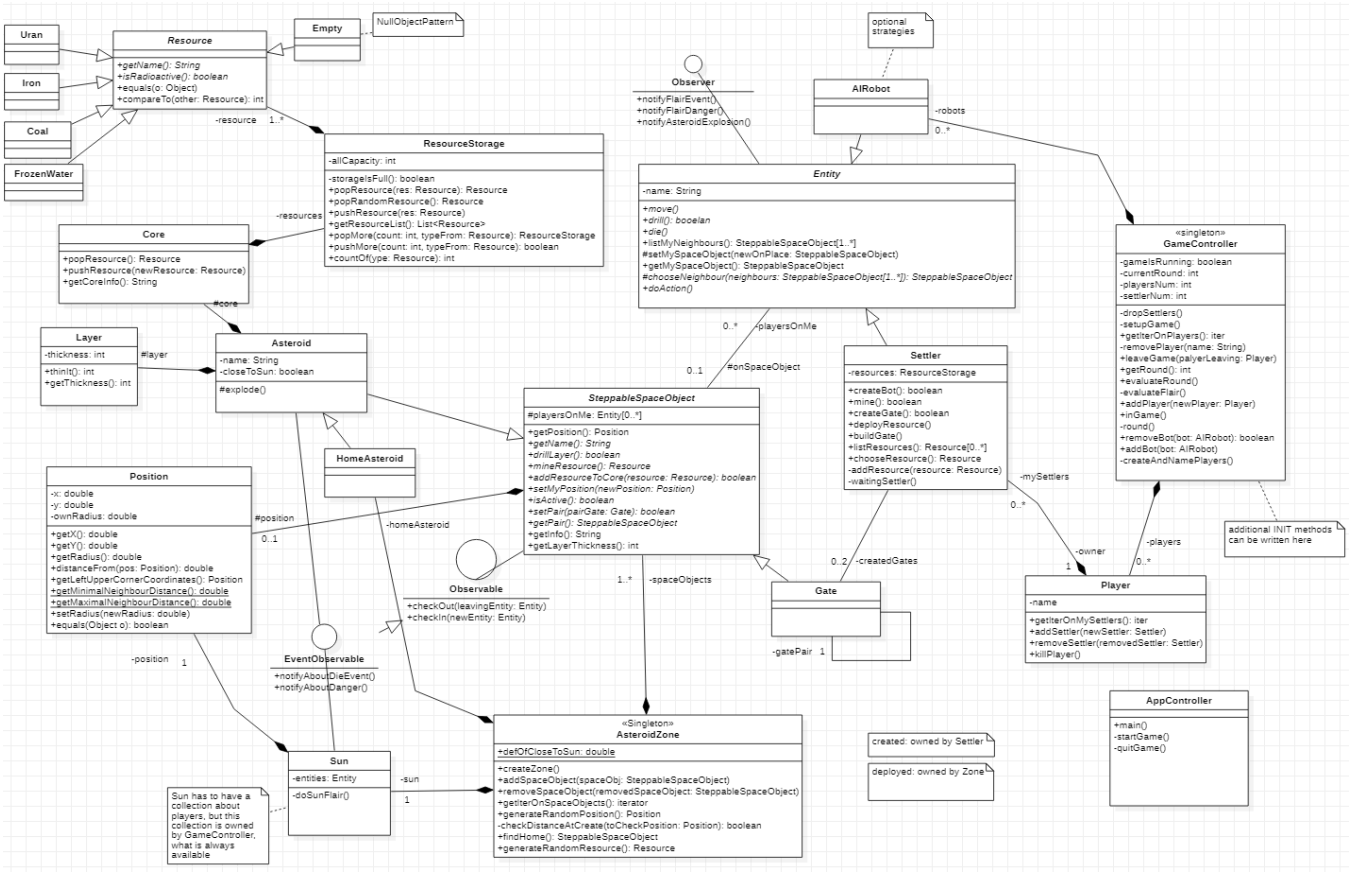
7. Konzept von Prototyp

7.0 Wirkung der Anderung auf den Modell

7.0.1 Änderungen in Klassendiagramm



7.0.2 Neue oder veränderte Methoden

7.0.2.1 Ressource
Die Klasse ist schon abstrakt nicht ein interface.

Methoden:

- *compareTo(Resource o): Vergleicht 2 Resource ob sie die Gleiche sind oder die Name der Ressourcen in alphabetische Reihenfolge kleiner oder größer ist.*
- *-equals(): Bestimmt ob 2 Ressourcen die Gleiche sind oder nicht*

7.0.2.2 Core
Methoden:
- *popResource() : nimmt ein Ressource aus dem Stack der Ressourcen*
- *pushResource(resource res): nimmt ein Resource in dem Stack rein*

7.0.2.3 ResourceStorage
Eine Hilfsklasse für die Speicherung von Ressourcen

Attribute:
-allCapacity: int - Anzahl der gespeicherten Ressourcen

Methoden:
- *storageIsFull(): Ergibt ob der Storage schon voll ist oder nicht*
- *popResource():nimmt ein Ressource aus dem Storage der Ressourcen*
- *pushResource(resource res): nimmt ein Resource in dem Storage Rein*
- *popMore(int count): nimmt mehrere Ressource aus dem Storage der Ressourcen*
- *pushMore(int count, resource res): nimmt mehr Resource in dem Storage Rein*
- *countOfType(resource res) : ergibt die Anzahl der Rohstoffe der angegebenen Rohstoff*
- *getResourceList(): gibt zurück die Liste der Rohstoffe*

7.0.2.4 HomeAsteroid
Vererbt sich von Asteroid nur die übergeschriebene Funktionen sind anders

7.0.2.5 AsteroidZone
Methoden:
- *genereateRandomResource(): Es gibt zurück eine zufällig gewählte Ressource für die einfüllung der Kern beim Generierung der Asteroiden*
- *findHome(): Es gibt zurück das StappableSpaceObject was den Basis Asteroid speichert*

7.0.2.6 SteppableSpaceObject
Methoden:
- *getLayerThickness(): Es gibt zurück ein int was die Manteldicke des Mantels in dem Asteroide speichert*

7.0.2.7 Entity
Methoden:
- *doAction(): In diesem Methode wird der Spieler auswählen welche Tätigkeit er ausführen wird z.B. Move, Drill usw.*

7.0.2.8 Settler
Attribute:
- *ResourceStorage: resources - Die Rohstoffe sind schon in einem Hilfsklasse ResourceStorage speichert nicht in einem Liste*

Methoden:
- *waitingSettler: Es macht nix, nur eine leere Runde um weitergehen zu können.*

7.0.2.9 GameController
Methoden:

- *createAndNamePlayers(): es herstellt die Spieler mit dem eingegebenen Namen und herstellt sie.*

7.0.2.10 Position
Methoden
- *equals(): Kontrolliert ob 2 Positionen die Gleiche sind oder nicht.*

7.0.3 Sequenz Diagramme

Es gibt keine nennenswerten Änderungen an den Sequenzen.

7.1 Interface Definition von Prototyp

7.1.1 Die allg. Beschreibung der Interface

Die Schnittstelle empfängt nur Befehle von der Standardeingabe und druckt alle Ausgaben an die Standardausgabe. Auf diese Weise kann es von einem Terminal aus verwendet und mithilfe des zu erstellenden Testdienstprogramms in Eingabedateien umgeleitet werden. Dies ermöglicht automatische Tests mit vorgefertigten Testfällen. Die Testfälle stammen aus der Reihe von Befehlen, die an den Prototyp gegeben werden sollen, sowie aus der korrekten Ausgabe, die für diese Reihe gegeben werden soll. Tests sind erfolgreich, wenn die tatsächliche und die beschriebene erwartete Ausgabe gleich sind.

7.1.2 Eingabesprache

jsonConfigTemplate:

Es ist eigentlich ein Testfall. Zeigt, welche Befehle ein Test mit welchen Parametern ausführen soll. Dementsprechend können Sie Testfälle manuell durchführen.

```
{
  "testname": "defaultTestName --- this test has default values, and don't do anything",
  "playerNumber": 1,
  "playerNames": ["testName"],
  "settlerNumber": 1,
  "asteroidNumber": 23,
  "range": 1000,
  "maxRound": 22,
  "sunFlairInEveryXRound": 10,
  "homeCapacity": 5,
  "settlerCapacity": 10,
  "defOfCloseToSun": 500,
  "maxNeighbourDistance": 500,
  "realCommandQueue": []
}
```

Die vom Prototyp akzeptierten Befehle sind:

start

Beschreibung: Starte das Spiel.
Optionen: -

move

Beschreibung: Schritt mit einem Siedler.
Optionen: Die Orte zu gehen.

drill

Beschreibung: Reduktion der Kortikalis des aktuellen Asteroiden.

Optionen: -

mine

Beschreibung**:** Rohstoffgewinnung aus einem Asteroiden.
Optionen**:** -

create gate

Beschreibung**:** Ein Teleport Tor machen.
Optionen**: -**

build gate

Beschreibung**:** Der Spieler legt das Teleporttor ab.
Optionen**: -**

create robot

Beschreibung**:** Ein AIRobot machen.
Optionen**: -**

list neighbours

Beschreibung**:** Liste benachbarter Asteroiden
Optionen**: -**

7.1.3 Ausgangssprache

Im Programm generierte Logos (auf Englisch).

7.2 Alle detaillierte Use-Case

Name der Use- Käse	***move***
Kurze Beschreibung	Schritt mit einem Siedler.
Aktoren	Tester
Drehbuch	Es bewegt sich zu einem der benachbarten Asteroiden.

Name der Use-Käse	***drill***
Kurze Beschreibung	Reduktion der Kortikalis des aktuellen Asteroiden.
Aktoren	Tester
Drehbuch	Es reduziert der Mantel des Asteroiden um eine Einheit.

Name der Use-Käse	***mine***
Kurze Beschreibung	Rohstoffgewinnung aus einem Asteroiden.
Aktoren	Tester
Drehbuch	Es können verschiedene Rohstoffe abgebaut werden. Wenn Sie sich auf einem radioaktiven Asteroiden in der Nähe befinden, explodiert der Siedler

Name der Use-Käse	***create gate***
Kurze Beschreibung	Ein Teleport Tor machen.
Aktoren	Tester
Drehbuch	Wenn es genug Rohmaterial gibt, um es herzustellen, kann es hergestellt werden, wenn nicht, dann nicht.

Name der Use-Käse	***build gate***
Kurze Beschreibung	Der Spieler legt das Teleporttor ab.
Aktoren	Tester
Drehbuch	Wenn es ein fertiges Teleport-Tor gibt, kann es abgelegt werden, wenn nicht, dann nein.

Name der Use-Käse	***create robot***
Kurze Beschreibung	Ein AIRobot machen.
Aktoren	Tester
Drehbuch	Wenn es genug Rohmaterial gibt, um es herzustellen, kann es hergestellt werden, wenn nicht, dann nicht.

Name der Use-Käse	***list neighbours***
Kurze Beschreibung	Liste der benachbarten Asteroiden
Aktoren	Tester
Drehbuch	Listet benachbarte Asteroiden auf

7.3 Testplan

Testen ist jetzt manuell, also man muss die Testkonfiguration angeben, und am Ende das Ergebnis manuell auswerten.
Um eine Testfall laufen lassen muss man im Testmodus eintreten, und dann die vordefinierte Testbeschreibungsdatei für Programm angeben. (z.B: testconfigs/createRobotTest.json)

Test Käse:	createRobotTest
Kurze Beschreibung:	Diese Testfall beweist, dass ein Settler kann ein Robot konstruieren.
Eingabedatei:	testconfigs/createRobotTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also ein Robot hergestellt wurde, aber die Zeitstempel sind anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	createBot
Testziel:	Eine Robot herstellen (Settler herstellt ein AIRobot)

```
      round: 1 - 2021-03-23 17:12:43 [main] TRACE  callStack
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
5
         createBot() called (Settler) - 2021-03-23 17:12:43 [main] TRACE  callStack
            AIRobot constructor called - 2021-03-23 17:12:43 [main] TRACE  callStack
         doAction() called (AIRobot) - 2021-03-23 17:12:43 [main] TRACE  callStack
            move() called (Entity) - 2021-03-23 17:12:43 [main] TRACE  callStack
               listMyNeighbours() called (Entity) - 2021-03-23 17:12:43 [main] TRACE  callStack
name: temp0, layer: 4, core: Uran, isCloseToSun: true, position: x=279.0 y=391.0
               chooseNeighbour() called (AIRobot) - 2021-03-23 17:12:43 [main] TRACE  callStack
               checkOut() called - 2021-03-23 17:12:43 [main] TRACE  callStack
               checkIn() called - 2021-03-23 17:12:43 [main] TRACE  callStack
      evaluateRound() called - 2021-03-23 17:12:43 [main] TRACE  callStack
         evaluateFlair() called - 2021-03-23 17:12:43 [main] TRACE  callStack
         listResources() called (Settler) - 2021-03-23 17:12:43 [main] TRACE  callStack
Coal
Uran
FrozenWater
Iron
Iron
```

Test Käse:	createZoneTest
Kurze Beschreibung:	Diese Testfall konstruiert ein AsteroidZone, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll.
Eingabedatei:	testconfigs/createZoneTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also das AsteroidZone ist fertig, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	createZone
Testziel:	Eine AsteroidZone herstellen nach Konfiguration.

```
------
testName
Type the number of desired settlers for each player
------
1
      createZone() called - 2021-03-23 17:15:11 [main] TRACE  callStack
         Sun constructor called - 2021-03-23 17:15:11 [main] TRACE  callStack
         Asteroid constructor called - 2021-03-23 17:15:11 [main] TRACE  callStack
         HomeAsteroid constructor called - 2021-03-23 17:15:11 [main] TRACE  callStack
         Asteroid constructor called - 2021-03-23 17:15:11 [main] TRACE  callStack
         Asteroid constructor called - 2021-03-23 17:15:11 [main] TRACE  callStack
      dropSettlers() called - 2021-03-23 17:15:11 [main] TRACE  callStack
         Settler constructor called - 2021-03-23 17:15:11 [main] TRACE  callStack
         checkIn() called - 2021-03-23 17:15:11 [main] TRACE  callStack
         checkIn() called (Sun) - 2021-03-23 17:15:11 [main] TRACE  callStack

Setup ended
   inGame() - 2021-03-23 17:15:11 [main] TRACE  callStack
```

Test Käse:	explodeTest
Kurze 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.
Eingabedatei:	testconfigs/explodeTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also der Settler ist gestorben, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	explode
Testziel:	Eine AsteroidExplosion herstellen, und durchführen.

```
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
1
         drill() called (Settler) - 2021-03-23 17:07:15 [main] TRACE  callStack
            drillLayer() called (Asteroid) - 2021-03-23 17:07:15 [main] TRACE  callStack
               thinIt() called (Layer) - 2021-03-23 17:07:15 [main] TRACE  callStack
                  The layer thickness is after the drilling: 0 - 2021-03-23 17:07:15 [main] TRACE  callStack
               popResource() called (Core) - 2021-03-23 17:07:15 [main] TRACE  callStack
               popRandomResource (ResourceStorage) called - 2021-03-23 17:07:15 [main] TRACE  callStack
                  The resource was: Uran - 2021-03-23 17:07:15 [main] TRACE  callStack
               pushResource() called (Core) - 2021-03-23 17:07:15 [main] TRACE  callStack
               notifyAboutDieEvent() called (Asteroid) - 2021-03-23 17:07:15 [main] TRACE  callStack
                  explode() called (Asteroid) - 2021-03-23 17:07:15 [main] TRACE  callStack
                     notifyAsteroidExplosion() called (Settler) - 2021-03-23 17:07:15 [main] TRACE  callStack
                        die() called (Settler) - 2021-03-23 17:07:15 [main] TRACE  callStack
                           checkOut() called - 2021-03-23 17:07:15 [main] TRACE  callStack
                           killPlayer() called - 2021-03-23 17:07:15 [main] TRACE  callStack
                           checkOut() called (Sun) - 2021-03-23 17:07:15 [main] TRACE  callStack
                     removeSpaceObject() called - 2021-03-23 17:07:15 [main] TRACE  callStack
      evaluateRound() called - 2021-03-23 17:07:15 [main] TRACE  callStack
         evaluateFlair() called - 2021-03-23 17:07:15 [main] TRACE  callStack
Game lost
```

Test Käse:	listNeighboursTest
Kurze Beschreibung:	Testen das Auszahlfunktion, in dem wir alle benachbarte Asteroide durchschauen können. Falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll.
Eingabedatei:	testconfigs/listNeighboursTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also manche Nachbarn ausgelistet sind, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	listMyNeighbours
Testziel:	Nachbarnasteroiden auflisten.

```
   inGame() - 2021-03-23 17:21:52 [main] TRACE  callStack
      round: 1 - 2021-03-23 17:21:52 [main] TRACE  callStack
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
7
         listMyNeighbours() called (Entity) - 2021-03-23 17:21:52 [main] TRACE  callStack
name: temp0, layer: 4, core: Uran, isCloseToSun: true, position: x=279.0 y=391.0
name: temp1, layer: 3, core: Iron, isCloseToSun: true, position: x=512.0 y=485.0
name: temp5, layer: 4, core: Empty, isCloseToSun: true, position: x=172.0 y=80.0
name: temp3, layer: 5, core: Coal, isCloseToSun: true, position: x=183.0 y=109.0
name: temp6, layer: 4, core: FrozenWater, isCloseToSun: false, position: x=491.0 y=710.0
      evaluateRound() called - 2021-03-23 17:21:52 [main] TRACE  callStack
         evaluateFlair() called - 2021-03-23 17:21:52 [main] TRACE  callStack
         listResources() called (Settler) - 2021-03-23 17:21:52 [main] TRACE  callStack
Coal
Coal
Uran
Uran
FrozenWater
Iron
Iron
Iron
```

Test Käse:	sunFlairTest
Kurze 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.
Eingabedatei:	testconfigs/sunFlairTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also der Settler ist gestorben auf einem anderen Asteroid, das sie/er nicht geschützt hat. Die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	SunFlair
Testziel:	Auf eine Solarflair warten, und wegen der sterben.

```
temp18
temp3
temp8
temp14
temp21
------
temp1
            checkOut() called - 2021-03-23 17:26:36 [main] TRACE  callStack
            checkIn() called - 2021-03-23 17:26:36 [main] TRACE  callStack
      evaluateRound() called - 2021-03-23 17:26:36 [main] TRACE  callStack
         evaluateFlair() called - 2021-03-23 17:26:36 [main] TRACE  callStack
      round: 2 - 2021-03-23 17:26:36 [main] TRACE  callStack
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
8
         waitingSettler() called (Settler) - 2021-03-23 17:26:36 [main] TRACE  callStack
      evaluateRound() called - 2021-03-23 17:26:36 [main] TRACE  callStack
         evaluateFlair() called - 2021-03-23 17:26:36 [main] TRACE  callStack
            notifyAboutDieEvent() called (Sun) - 2021-03-23 17:26:36 [main] TRACE  callStack
               doSunFlair() called (Sun) - 2021-03-23 17:26:36 [main] TRACE  callStack
                  notifyFlairEvent() called (Settler) - 2021-03-23 17:26:36 [main] TRACE  callStack
                     die() called (Settler) - 2021-03-23 17:26:36 [main] TRACE  callStack
                        checkOut() called - 2021-03-23 17:26:36 [main] TRACE  callStack
                        killPlayer() called - 2021-03-23 17:26:36 [main] TRACE  callStack
                        checkOut() called (Sun) - 2021-03-23 17:26:36 [main] TRACE  callStack
                  killPlayer() called - 2021-03-23 17:26:36 [main] TRACE  callStack
Game lost
```

Test Käse:	sunFlairTest2
Kurze Beschreibung:	Testen einen SunFlair, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler nichts machen, weil die auf dem Home automatisch geschützt ist, also er/sie muss nur warten.
Eingabedatei:	testconfigs/sunFlairTest2.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also der Settler noch lebend, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	SunFlair
Testziel:	Auf eine Solarflair warten, und das Flairereignis überleben.

```
Setup ended
   inGame() - 2021-03-23 17:28:08 [main] TRACE  callStack
      round: 1 - 2021-03-23 17:28:08 [main] TRACE  callStack
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
8
         waitingSettler() called (Settler) - 2021-03-23 17:28:08 [main] TRACE  callStack
      evaluateRound() called - 2021-03-23 17:28:08 [main] TRACE  callStack
         evaluateFlair() called - 2021-03-23 17:28:08 [main] TRACE  callStack
            notifyAboutDieEvent() called (Sun) - 2021-03-23 17:28:08 [main] TRACE  callStack
               doSunFlair() called (Sun) - 2021-03-23 17:28:08 [main] TRACE  callStack
                  notifyFlairEvent() called (Settler) - 2021-03-23 17:28:08 [main] TRACE  callStack
                     mineResource() called (HomeAsteroid) - 2021-03-23 17:28:08 [main] TRACE  callStack
                        popResource() called (Core) - 2021-03-23 17:28:08 [main] TRACE  callStack
                        popRandomResource (ResourceStorage) called - 2021-03-23 17:28:08 [main] TRACE  callStack
                           The resource was: Empty - 2021-03-23 17:28:08 [main] TRACE  callStack
         listResources() called (Settler) - 2021-03-23 17:28:08 [main] TRACE  callStack
Coal
Coal
Uran
Uran
FrozenWater
Iron
Iron
Iron
```

Test Käse:	drillTest
Kurze Beschreibung:	Testen das Drill Funktion, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler bohren(drill).
Eingabedatei:	testconfigs/drillTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also das Asteroid ist gebohrt, ihre Mantelgröße ist kleiner (falls noch nicht durchgebohrt), aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	drill()
Testziel:	Eine Asteroid bohren.

```
------
1
        drill() called (Settler)
            drillLayer() called (Asteroid)
                thinIt() called (Layer)
                    The layer thickness is after the drilling: 0
                popResource() called (Core)
                popRandomResource (ResourceStorage) called
                    The resource was: Coal
                pushResource() called (Core)
    evaluateRound() called
        evaluateFlair() called
            notifyAboutDanger() called (Sun)
                notifyFlairDanger() called (Settler)
    round: 7
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
1
        drill() called (Settler)
            drillLayer() called (Asteroid)
                thinIt() called (Layer)
                    The layer thickness is after the drilling: 0
                popResource() called (Core)
                popRandomResource (ResourceStorage) called
                    The resource was: Coal
                pushResource() called (Core)
```

Test Käse:	mineTest
Kurze Beschreibung:	Testen das MineFunktion, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler fördern(mine).
Eingabedatei:	testconfigs/mineTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also das Asteroid ist gebohrt und fördert, ihre Kern ist leer, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	mine()
Testziel:	Eine Asteroid fördern.

```
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
2
        mine() called (Settler)
            mineResource() called (Asteroid)
            popResource() called (Core)
            popRandomResource (ResourceStorage) called
                The resource was: Coal
        addResource called
evaluateRound() called
    evaluateFlair() called
round: 8
```

```
------
1
        drill() called (Settler)
            drillLayer() called (Asteroid)
                thinIt() called (Layer)
                    The layer thickness is after the drilling: 0
                popResource() called (Core)
                popRandomResource (ResourceStorage) called
                    The resource was: Coal
                pushResource() called (Core)
    evaluateRound() called
        evaluateFlair() called
            notifyAboutDanger() called (Sun)
                notifyFlairDanger() called (Settler)
    round: 7
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
1
        drill() called (Settler)
            drillLayer() called (Asteroid)
                thinIt() called (Layer)
                    The layer thickness is after the drilling: 0
                popResource() called (Core)
                popRandomResource (ResourceStorage) called
                    The resource was: Coal
                pushResource() called (Core)
```

Test Käse:	deployResourceTest
Kurze Beschreibung:	Testen das deployResource, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Rohstoff zurückstecken(deployResource).
Eingabedatei:	testconfigs/deployResource.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also das Asteroid ist gebohrt und fördert, ihre

	Kern ist leer, und dann ist die geförderte Rohstoff zurückgesteckt, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	deployResource()
Testziel:	In eine leere Asteroide ein Rohstoff zurückstecken

```
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
6
    deployResource() called (Settler)
        listResources() called (Settler)
Coal
Coal
Uran
Uran
FrozenWater
Iron
Iron
Iron
Coal
        chooseResource() called (Settler)
------
1
        addResourceToCore() called (Asteroid)
            popResource() called (Core)
            popRandomResource (ResourceStorage) called
                The resource was: Empty
            pushResource() called (Core)
```

```
------
1
        drill() called (Settler)
            drillLayer() called (Asteroid)
                thinIt() called (Layer)
                    The layer thickness is after the drilling: 0
                popResource() called (Core)
                popRandomResource (ResourceStorage) called
                    The resource was: Coal
                pushResource() called (Core)
    evaluateRound() called
        evaluateFlair() called
            notifyAboutDanger() called (Sun)
                notifyFlairDanger() called (Settler)
    round: 7
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
1
        drill() called (Settler)
            drillLayer() called (Asteroid)
                thinIt() called (Layer)
                    The layer thickness is after the drilling: 0
                popResource() called (Core)
                popRandomResource (ResourceStorage) called
                    The resource was: Coal
                pushResource() called (Core)
```

Test Käse:	mineTest
Kurze Beschreibung:	Testen das deployResource, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler ein Rohstoff zurückstecken(deployResource).
Eingabedatei:	testconfigs/deployResource.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also das Asteroid ist gebohrt und fördert, ihre

	Kern ist leer, und dann ist die geförderte Rohstoff zurückgesteckt, aber die Zeitstempel sind natürlich anders. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	deployResource()
Testziel:	In eine leere Asteroide ein Rohstoff zurückstecken

```
------
0
           move() called (Entity)
               listMyNeighbours() called (Entity)
name: temp20, layer: 5, core: Iron, isCloseToSun: true, position: x=1.0 y=355.0
name: temp16, layer: 4, core: Iron, isCloseToSun: false, position: x=752.0 y=668.0
name: temp17, layer: 5, core: FrozenWater, isCloseToSun: false, position: x=601.0 y=624.0
name: temp18, layer: 3, core: Uran, isCloseToSun: true, position: x=373.0 y=363.0
name: temp10, layer: 4, core: Uran, isCloseToSun: false, position: x=797.0 y=208.0
name: temp5, layer: 4, core: Empty, isCloseToSun: true, position: x=172.0 y=80.0
name: temp11, layer: 5, core: Iron, isCloseToSun: false, position: x=781.0 y=476.0
name: temp7, layer: 5, core: Iron, isCloseToSun: false, position: x=11.0 y=695.0
name: temp19, layer: 5, core: Coal, isCloseToSun: true, position: x=97.0 y=607.0
name: temp9, layer: 5, core: Empty, isCloseToSun: true, position: x=390.0 y=263.0
name: temp12, layer: 5, core: Uran, isCloseToSun: true, position: x=114.0 y=377.0
name: temp0, layer: 4, core: Uran, isCloseToSun: true, position: x=279.0 y=391.0
name: temp8, layer: 4, core: Iron, isCloseToSun: false, position: x=155.0 y=677.0
name: temp6, layer: 4, core: FrozenWater, isCloseToSun: false, position: x=491.0 y=710.0
name: temp22, layer: 4, core: FrozenWater, isCloseToSun: true, position: x=367.0 y=585.0
name: temp15, layer: 5, core: FrozenWater, isCloseToSun: true, position: x=630.0 y=201.0
name: temp3, layer: 5, core: Coal, isCloseToSun: true, position: x=183.0 y=109.0
name: temp14, layer: 5, core: Uran, isCloseToSun: false, position: x=645.0 y=884.0
name: temp1, layer: 3, core: Iron, isCloseToSun: true, position: x=512.0 y=485.0
name: temp21, layer: 5, core: Coal, isCloseToSun: true, position: x=271.0 y=349.0
               chooseNeighbour() called (Settler)
```

```
Choose destination:
temp20
temp16
temp17
temp18
temp10
temp5
temp11
temp7
temp19
temp9
temp12
temp0
temp8
temp6
temp22
temp15
temp3
temp14
temp1
temp21
------
temp11
               checkOut() called
            checkIn() called
    evaluateRound() called
        evaluateFlair() called
    round: 2
```

Test Käse:	moveTest
Kurze Beschreibung:	Testen das move, und falls alles stimmt, keine Fehlermeldung während der Ablauf ausschreiben werden soll. Dazu muss ein Settler , um zu einem anderen Asteroiden zu ziehen
Eingabedatei:	testconfigs/moveTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also der Siedler befindet sich auf dem entsprechenden Asteroiden. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	move()
Testziel:	Ein Siedler, um zu einem anderen Asteroiden zu bewegen.

Test Käse:	createGateTest
Kurze 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.
Eingabedatei:	testconfigs/createGateTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also der Siedler konnte die Teleporter erstellen. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	createGate()
Testziel:	Erstellen eines Teleporterpaares aus den erforderlichen Ressourcen.

```
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
3
        createGate() called (Settler)
            Gate constructor called
            Gate constructor called
            The 2 gates were added to the player
            setPair() called (Gate)
            setPair() called (Gate)
    evaluateRound() called
        evaluateFlair() called
    round: 3
```

Test Käse:	buildGateTest
Kurze 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.
Eingabedatei:	testconfigs/buildGateTest.json
Erwartete Ausgabe:	siehe unten
Testauswertung:	Test ist erfolgreich, falls die erwartete Ausgabe erscheint, also der Siedler konnte die Teleporter platzieren. Alle andere Fall ist unerfolgreich.
Dazugehörige Sequence:	buildGate()
Testziel:	Platzieren eines Teleporterpaares in dem Spielfeld.

```
Choose an option:
0 : move
1 : drill
2 : mine
3 : create gate
4 : build gate
5 : create robot
6 : deploy resource
7 : list neighbours
8 : wait
------
4
        buildGate() called (Settler)
            addSpaceObject() called
```

7.4 Hilfsprogramme beim Testen, und ihre Spezifikation

Zurzeit gibt's keine spezielle Programme für Testen ausserhalb von **java JVM** und **gradle** oder **IntelliJ IDEA**.
Aber wir haben von uns definierte Testkonfigurationsdateien, die die Testszenarien beschreiben. Diese Konfigurationsdateien sind im ideaProject/testconfigs/ Verzeichnis, und das Programm kann diesen einlesen, und laufen lassen. Leider die Auswertung ist noch nicht automatisch.

Template dieses Testdateien: (Bsp: jsonConfigTemplate_DO_NOT_OVERWRITE.json)

```
{
  "testname": "defaultTestName --- this test has default values, and don't do anything",
  "playerNumber": 1,
  "playerNames": ["testName"],
  "settlerNumber": 1,
  "asteroidNumber": 23,
  "range": 1000,
  "maxRound": 22,
  "sunFlairInEveryXRound": 10,
  "homeCapacity": 5,
  "settlerCapacity": 10,
  "defOfCloseToSun": 500,
  "maxNeighbourDistance": 500,
  "realCommandQueue": []
}
```

7.5 Tagebuch

Anfang	Dauer	Teilnehmer	Beschreibung

2021.03.17.	1 Stunde	alle	Allg Gespräch über die Aufgaben
2021.03.17.	2 Stunde	Pongrácz	Fehlerbehebung (Concurrent Modification) und kleinere Korrektur im move().
2021.03.17	1 Stunde	Hrotkó	Seed einstellen für das Spiel und Modifikationen
2021.03.18	3-4 Stunde	Hrotkó	Asteroide generierung optimalisieren mit kleineren radius, und slipping realisieren und bug fixes
2021.03.21	5 Stunde	Pongrácz	Testumgebung herstellen, testconfig template herstellen, manche Testreporte schreiben
2021.03.23	2 Stunde	Alle	Tests schreiben, Dokument ergänzen und Gespräch über die Aufgaben
2021.03.22.			