3. Ausarbeitung des Analysenmodells 1

31 – marci_und_die_mitbewohner

Konsulent:

Kovács Márton

Mitglieder

Seben Domonkos András	(ETBCNP)	domi.seben@gmail.com
Szapula László	(DJQOM9)	szapula.laszlo.99@gmail.com
Filip Krisztina	(QE4L0M)	fkriszta997@gmail.com
Golej Márton Marcell	(V1BYVS)	golejmarci@gmail.com
Visy Tamás	(CTSJ3H)	tamas.visy@gmail.com

3. Ausarbeitung des Analysemodells 1

3.1 Objektkatalog

3.1.1 Controller

Kontrolliert den Ablauf des Spiels, sorgt für die wichtigere, weniger spezifische Aufgaben. Solche Aufgaben sind zum Beispiel die Behandlung der Schneestürme, die Bewegung der Charaktere, und die Steuerung ihrer Arbeit (also die Tritte). Bewartet den Zusammenhang der Eisschollen und Charakters. Weiß von den Dingen, die in den Eisschollen gefroren sind.

3.1.2 Eisscholle

Die elementare Einheit des Eisfeldes. Diese Bausteine repräsentieren den Zustand und ihre Änderung von den kleineren Einheiten des Eisfeldes. Beispiel dafür kann die Speicherung der Schneeschichten, der Aufbau oder Verfallen von Iglus, der Stand der Eisschollen, (gut/umgedreht) beziehungsweise ihre Arten (stabil/instabil/Loch) und Tragfähigkeit sein.

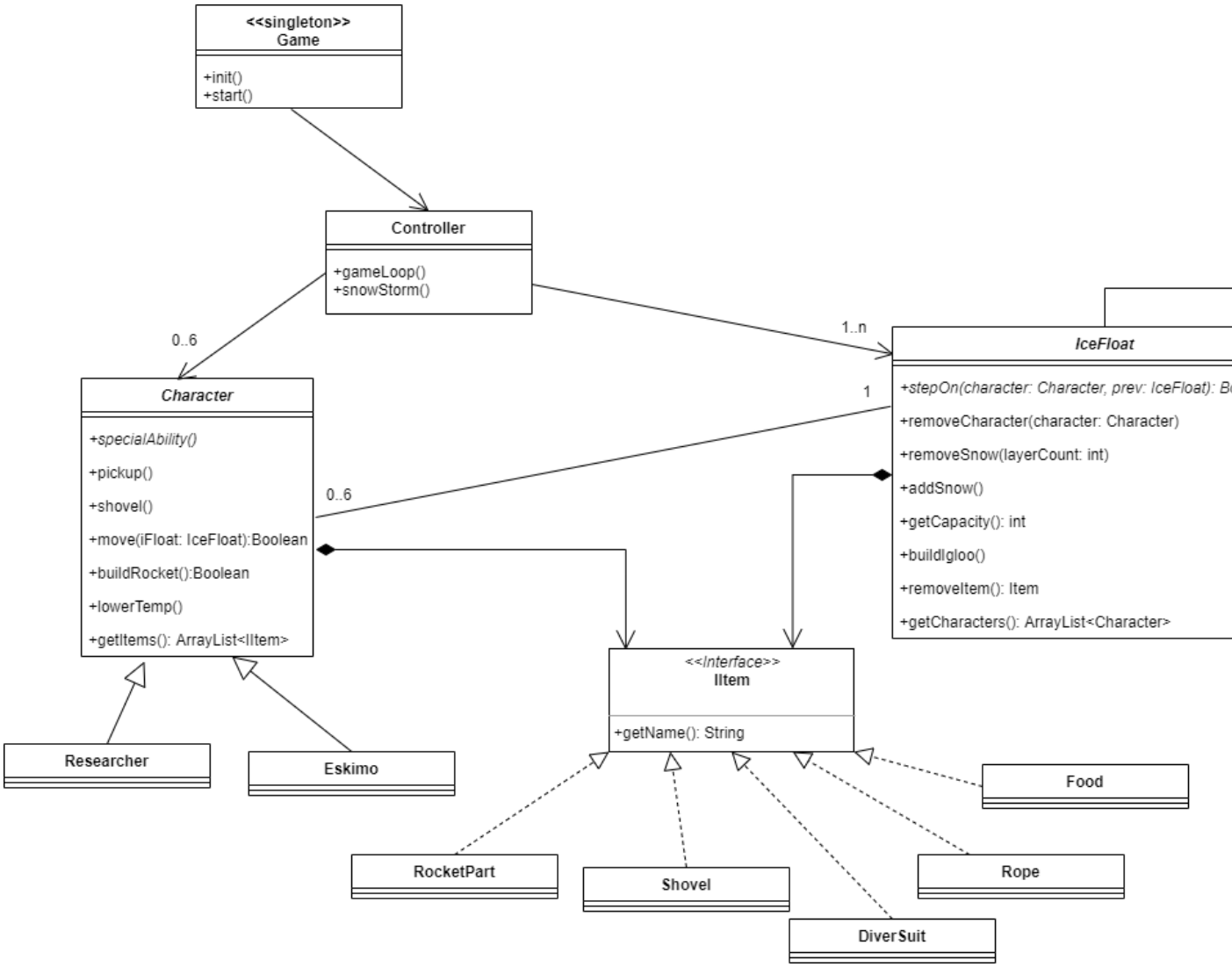
3.1.3 Character (Spieler)

Repräsentiert die Spieler. Es wird zwei Typen geben, ein Eskimo und ein Forscher. Ihr Ziel ist die Wartung der Körpertemperatur mit Essen und Arbeit. Besitzt und benutzt Gegenstände beziehungsweise schafft sie an.

3.1.4 Gegenstand

Die Gegenstände, die in den Eisschollen gefroren sind. Die Spieler schaffen sie an und verlagern sie sie. Es gibt 7 Typen davon: Schaufel, Seil, Taucheranzug, Lebensmittel und die 3 Bauteile der Leuchtpistole.

3.2 Statische Strukturdiagramme



3.3 Beschreibungen der Klassen

3.3.1 Game

- **Verantwortung**

Das Spiel zu initialisieren und starten.

- **Attribute**
  - **- controller**: Referenz zur Klasse “Controller”

- **Methoden**
  - **+ void init():** Diese Funktion initialisiert das Spiel. Also herstellt sie alle benötigte Klassen.
  - **+ void start():** Diese Funktion startet das Spiel. Es ruft also die Funktion “void gameLoop()” von der Klasse “Controller” an.

3.3.2 Controller

- **Verantwortung**

Das nacheinander Kommen von Spielern zu organisieren. Beim Wechseln zwischen Spieler ist diese Klasse auch verantwortlich für die Schneestürme.

- **Attribute**
  - **- icefloats**: Referenzlist zu den Eisschollen.
  - **- characters**: Referenzlist zu den Spielern.

- **Methoden**
  - **+ void gameLoop():** Diese Funktion ist verantwortlich für das nacheinander Kommen von Spielern, und beim Wechseln zwischen Spieler ruft die Funktion “*snowStorm()*” an.
  - **- void snowStorm():** Diese Funktion wählt einige Schneeschollen aus, und liegt Schnee darauf. Falls ein Spieler auf einer solchen Scholle steht, dann verliert er eine Einheit von Körpertemperatur außer des Falles, wenn dort ein Iglu ist. In diesem Fall verliert der Spieler keine Körpertemperatur, aber der Iglu wird zerstört.

3.3.3 Character

- **Verantwortung**

Die Durchführung von den möglichen Arbeiten. Die Klasse ermöglicht die Nachverfolgung und Änderung der Körpertemperatur des Characters.

- **Attribute**
  - **#icefloat**: Die Eisscholle, auf der der Character sich befindet.
  - **#items**: Referenzlist auf den Gegenständen, die der Character besitzt.
  - **-temp:** Repräsentiert die Körpertemperatur von dem Character.

- **Methoden**
  - **+abstract void specialAbility()**: Die einzige Funktion, die von den Subklassen implementiert werden muss. Diese Funktion führt die für den Charakter-Typ spezifische Arbeit durch.

- **+void pickup()**: Hier wird ein Gegenstand von der entsprechenden Eisscholle übernommen.
- **+void shovel()**: Ruft die Methode *removeSnow()* von der aktuellen Eisscholle und es nimmt in Betracht, ob der Character eine Schaufel hat.
- **+bool move(IceFloat float)**: Versucht auf die Eisscholle *float* zu treten. Falls der Versuch erfolgreich war, tritt die Methode mit *true* zurück.
- **+bool buildRocket()**: Das entspricht der Arbeit, mit der das Spiel gewonnen wird.
- **+void removeItem(Item i)**: Damit wird ein Gegenstand von *items* entfernt.
- **+void lowerTemp()**: Sinkt die Körpertemperatur des Characters um 1. Das wird typischerweise beim Schneesturm angerufen.

3.3.4 Researcher

- **Verantwortung**

Einen solchen Character zu realisieren, der herausfinden kann, wie viele Characters die benachbarten Eisschollen ertragen können.

- **Basisklasse**

Character

- **Methoden**
  - **+void specialAbility()**: Ermittelt die Kapazität einer nebeneinanderliegenden Eisscholle.

3.3.5 Eskimo

- **Verantwortung**

Einen solchen Character zu realisieren, der Iglus aufbauen kann.

- **Basisklassen**

Character

- **Methoden**
  - **+void specialAbility()**: Baut ein Iglu auf die aktuelle Eisscholle auf.

3.3.6 IItem

- **Verantwortung**

Das soll für einen abstrakten Gegenstand entsprechen, das einen Namen hat und als Objekt gespeichert werden kann.

- **Methoden**
  - **+String getName()**: Teilt den Namen des Gegenstandes mit.

3.3.7 IceFloat

- **Verantwortung**

Das ist eine abstrakte Klasse, das irgendwie regulieren muss, wie die Spieler auf sie treten. Ansonsten ist sie verantwortlich für die Schneeschichten und die Iglus, die sie charakterisieren.

- **Attribute**
  - **#neighbour:** eine Kollektion der benachbarten Eisschollen
  - **#item:** der Gegenstand, der in der Eisscholle gefroren ist (der gibt es nicht immer)
  - **#characters:** Die Characters, die auf dieser Eisscholle stehen.
  - **#iglu:** Durch diesem Attribut kann man sagen, ob es auf der Eisscholle ein Iglu gibt.
  - **#snowLevel:** Repräsentiert die Menge von Schnee auf der Eisscholle.
  - **#capacity:** Durch diesem Attribut kann man beobachten, wie viel Character der Eisscholle fördern kann.
- **Methoden**
  - **+abstract boolean stepOn(Character ch, IceFloat prev)**: Die Methode behandelt die Anfrage eines Characters, der auf diese Eisscholle treten möchte.
  - **+void removeCharacter(Character ch)**: Das entfernt der Character von einer Eisscholle.
  - **+void removeSnow(int layerCount)**: Die bestimmte Anzahl von Schichten werden von der Eisscholle entfernt.
  - **+void addSnow()**: Eine Schicht Schnee wird auf die Eisscholle gelegt. Sie wird von dem Schneesturm gerufen.
  - **+int getCapacity()**: Die Methode gibt ihre Kapazität bekannt.
  - **+void buildIglu()**: Die Methode erschafft ein Iglu.
  - **+Item removeItem()**: Durch dieser Methode wird das Gegenstand von der Eisscholle zu dem Charakter übernommen.

3.3.8 RocketPart

- **Verantwortung**

Die Bestandteile der Leuchtpistole zu identifizieren und es sagen können, ob die Bestandteile aufgenommen wurden.

- **Schnittstellen**

IItem

- **Methoden**
  - **+String getName()**:Liefert den Namen des Bestandteiles

3.3.10 DiverSuit

- **Verantwortung**

Es soll vermöglichen, dass ein Spieler überlebt einen Fall in ein Loch.

- **Schnittstellen**

IItem

- **Methoden**
  - **+String getName()**:Liefert den Namen des Gegenstandes

3.3.11 StableIceFloat

- **Verantwortung**

Zu wissen, wer die benachbarten Eisschollen sind und was für Zustände sie charakterisieren (z.B. Iglu, Schneeschichten).

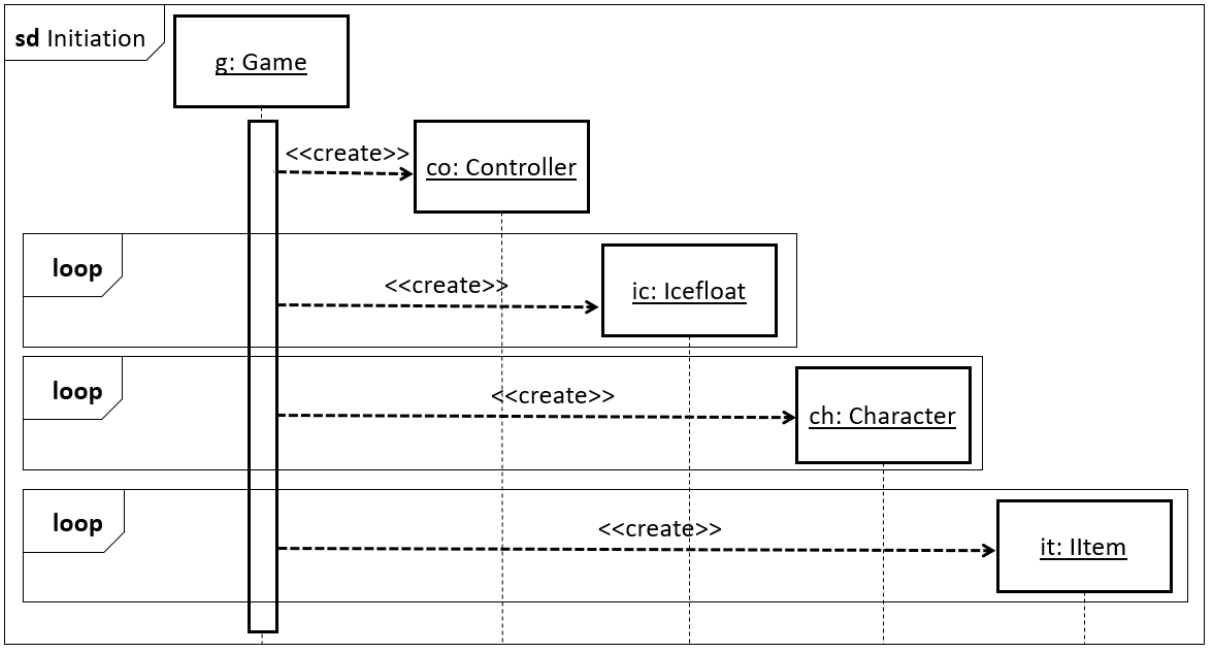
- **Basisklasse**

IceFloat

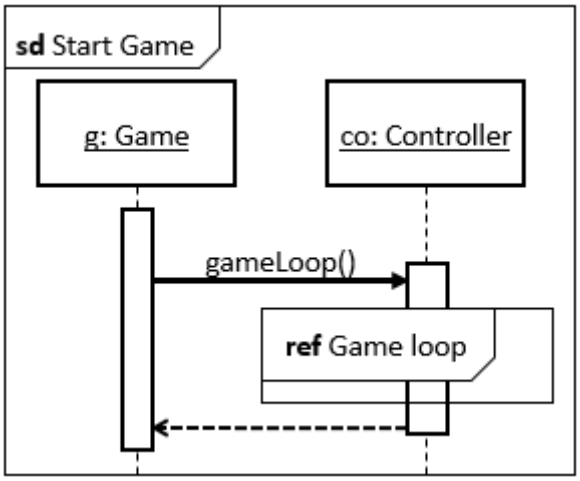
- **Methoden**
  - **+bool stepOn(Character character, IceFloat prev)**: Das akzeptiert den Character auf diese Eisscholle

3.4 Sequenzdiagramme

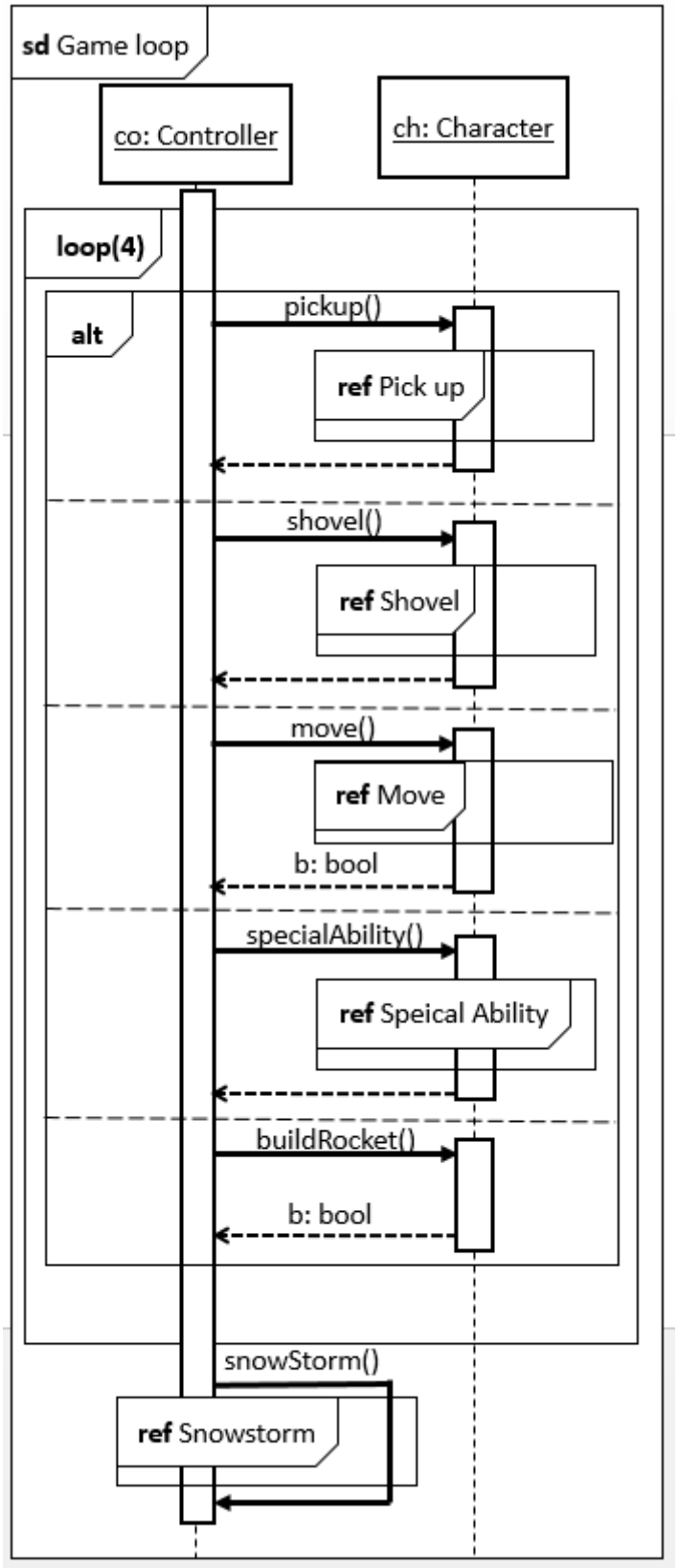
3.4.1 Initiation



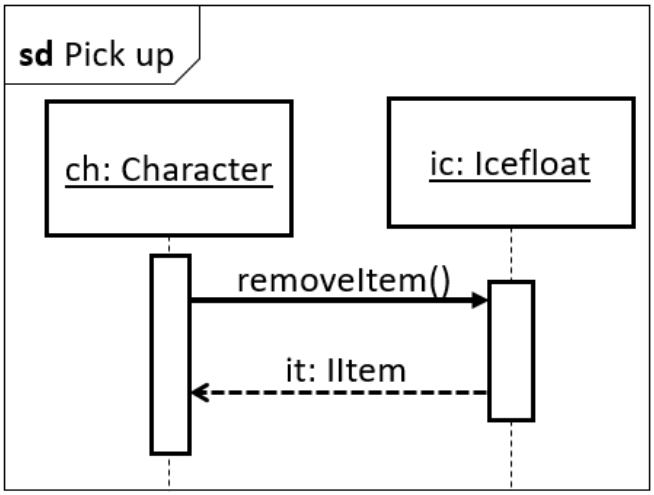
3.4.2 Start Game



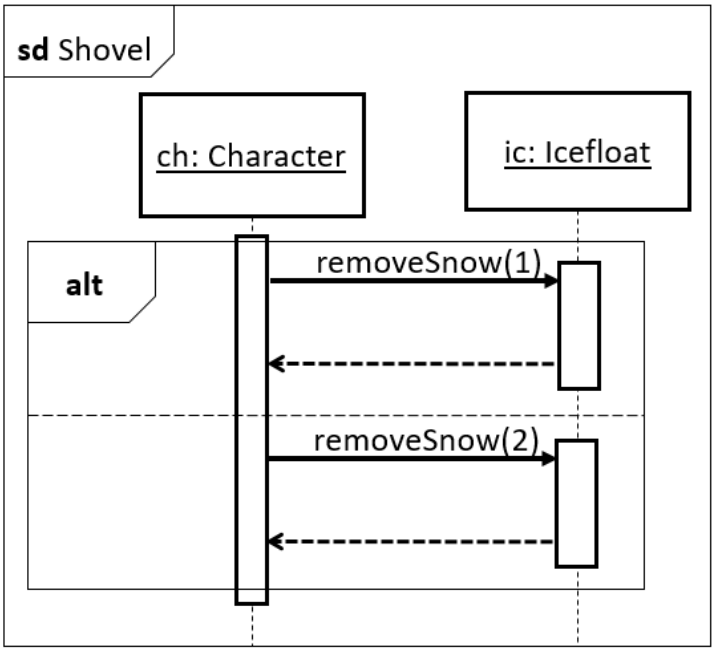
3.4.3 Game loop



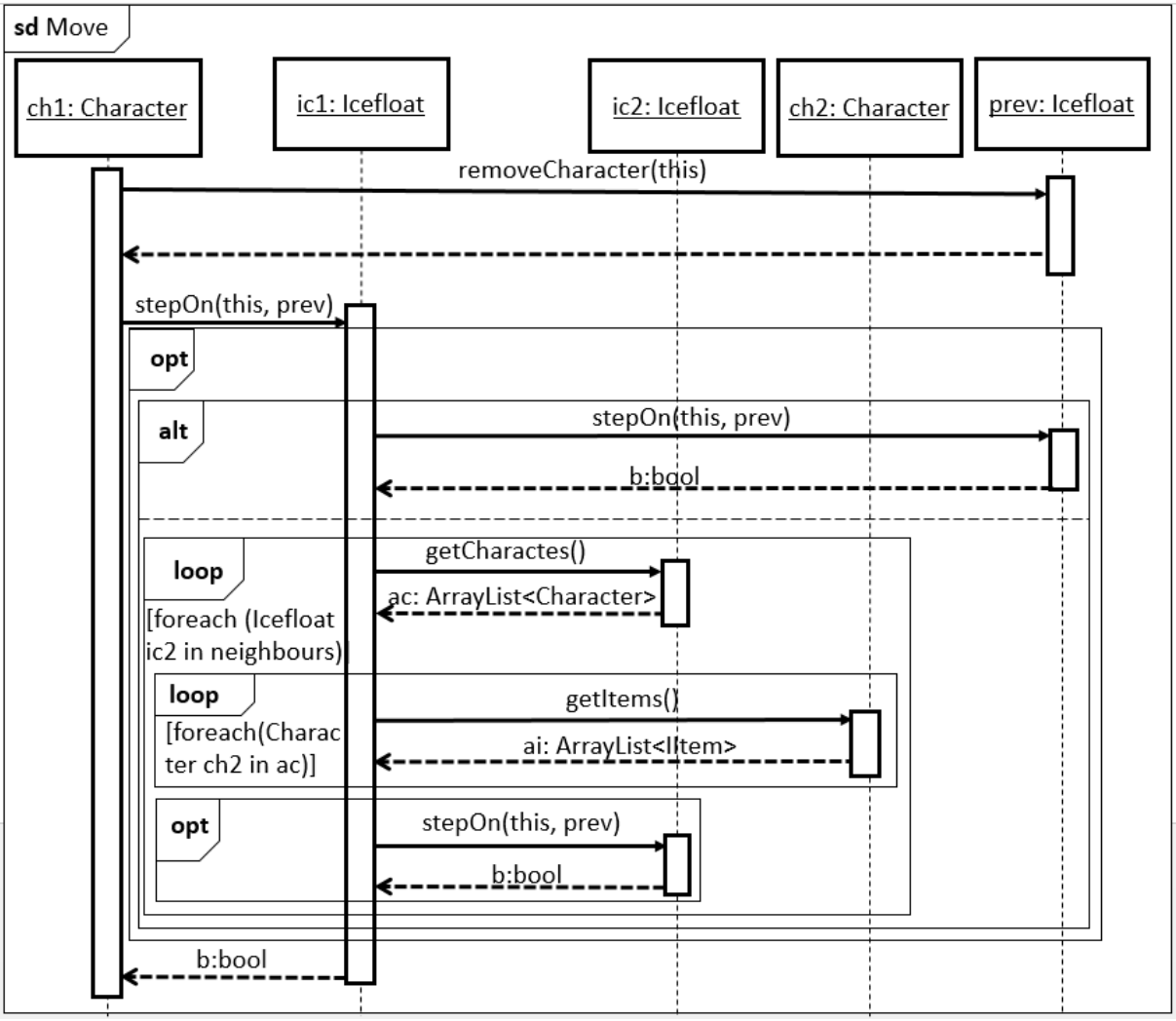
3.4.4 Pick up



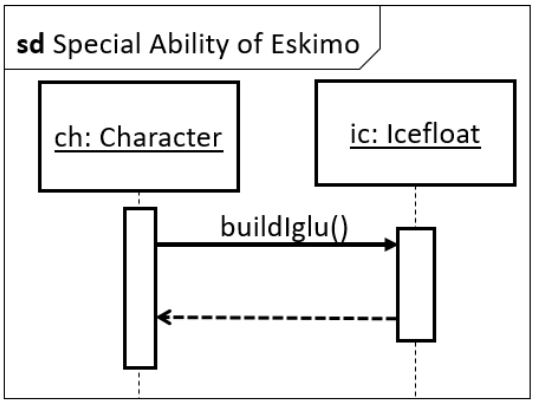
3.4.5 Shovel



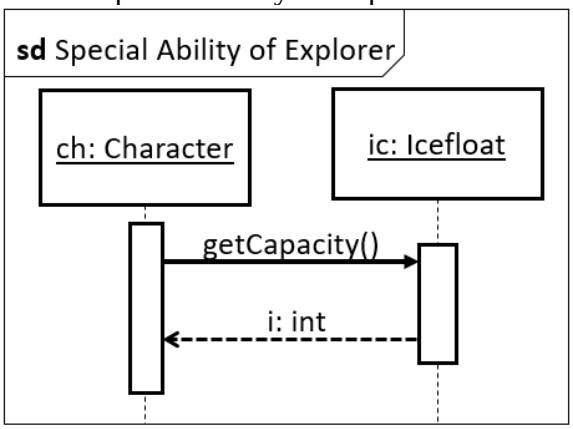
3.4.6 Move



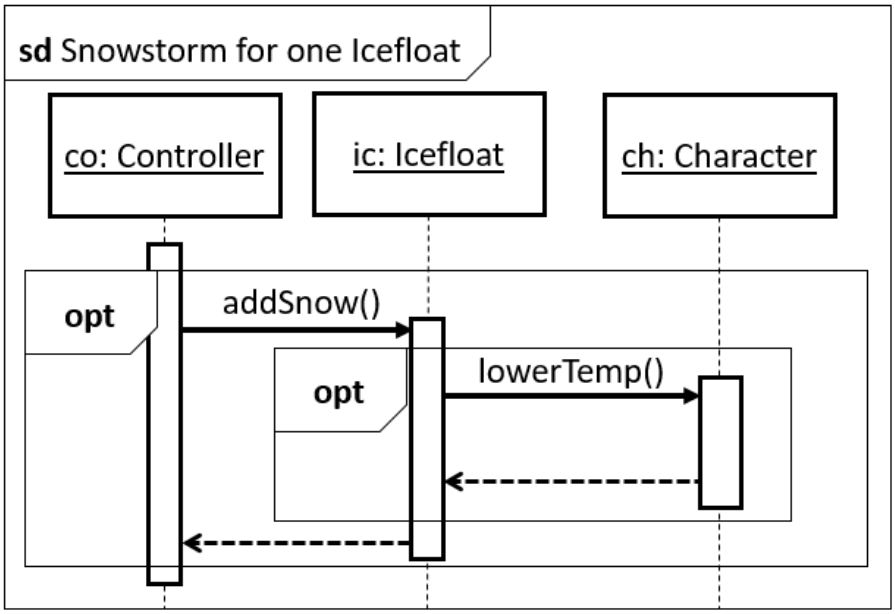
3.4.7 Special Ability of Eskimo



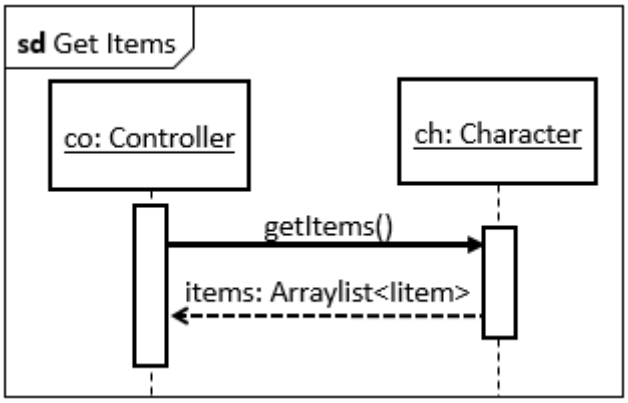
3.4.8 Special Ability of Explorer



3.4.9 Snowstorm



3.4.10 Get Items



3.5 Statechart Diagramme

Die Gruppe ist der Meinung, dass keine solche Funktionalität in der Aufgabe vorkommt, wann es sich lohnen würde, sie zu modellieren.