8. Detaillierte Pläne

31 – marci_und_die_mitbewohner

Konsulent:

Kovács Márton

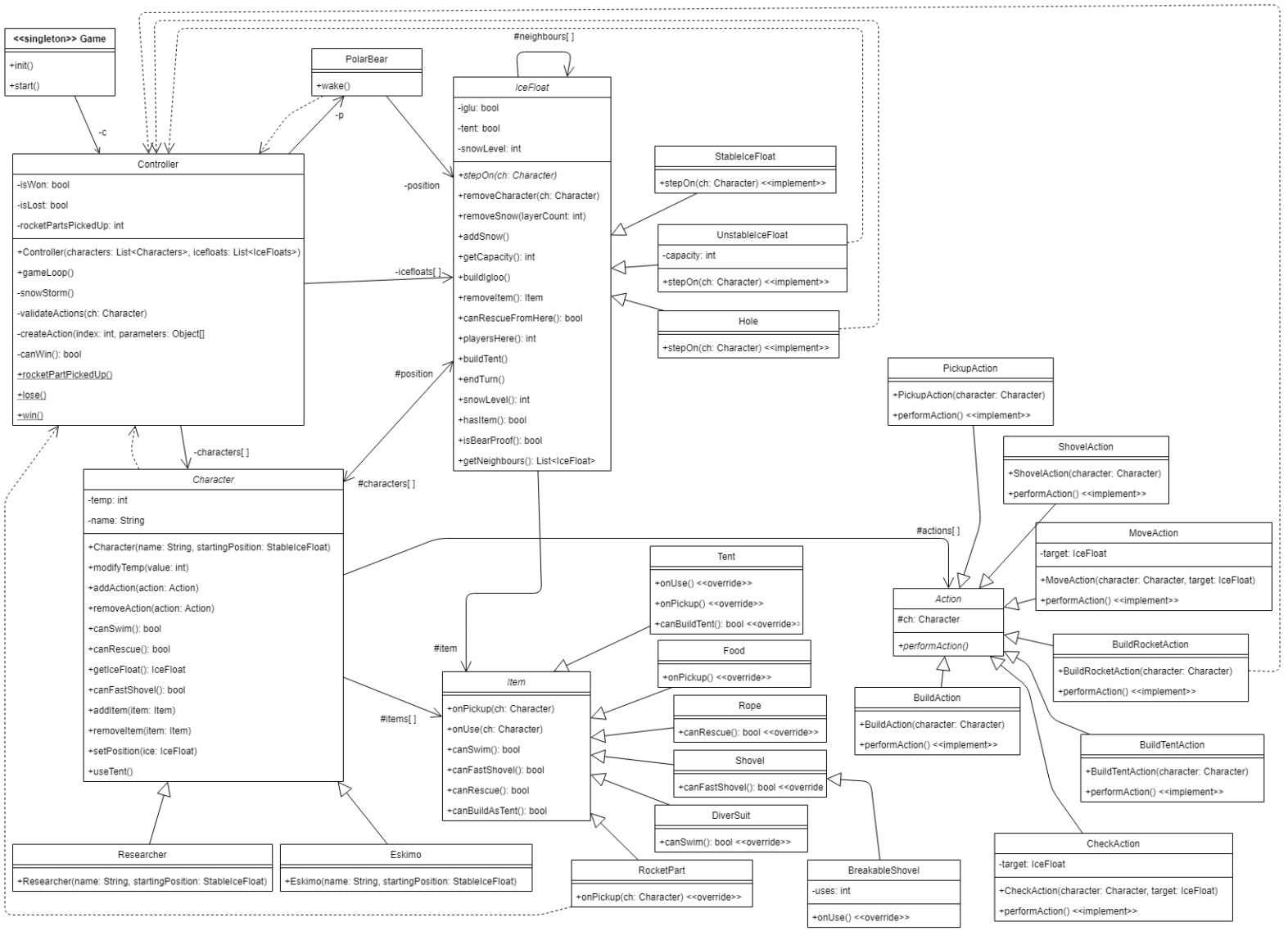
Mitglieder

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

8. Detaillierte Pläne

Während wir diese Sektion gemacht haben und die einzigen Klassen, ihre Funktionalitäten gründlich übergedacht haben, haben wir einige, kleinere Änderungen durchgeführt.
Die statische Funktionen **win()** und **lose()** gehören von hier an zur Klasse **Controller**.
Sie hat eine andere neue Methode auch, **rocketPartPickedUp()**. Damit kann sie die Anzahl der aufgenommenen Bestandteilen folgen und damit entscheiden, ob es eine gültige Aktion ist, die Rakete aufzubauen.
Was die Klasse **Character** betrifft, sie hat eine neue Methode **useTent()**, die auf ihre Schaufel **onUse()** anruft.
Die Klasse **IceFloat** hat eine Methode **stepOn**, die früher ein inkonsistentes Header hatte. Jetzt sieht sie so aus: **stepOn(character: Character): void**. Die Eisscholle, wo der Charakter vor dem Schritt gestanden ist, muss man nicht übernehmen. Der Rückwert ist nicht mehr boolean wegen der Änderung in dem Schritt-Mechanismus. Sie hat einige einfache neue public Methoden, die für die Bestimmung der gültigen Aktionen notwendig sind. (**hasItem(), playersHere(), snowLevel()**).
Über die Action-Klassen: Ihre Konstruktor-headers sind ab jetzt spezifiziert. Früher hat **BuildRocketAction** die Anzahl der aufgenommenen Bestandteilen verwaltet, dafür ist aber jetzt die **Controller** Klasse verantwortlich.

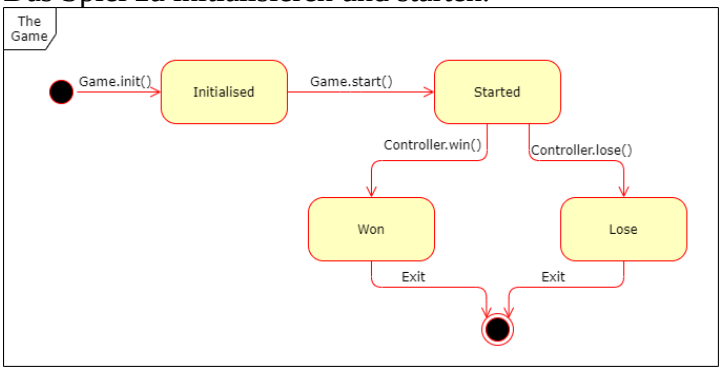
8.1 Pläne der Klassen und Methode



8.1.1 Game

- **Verantwortung**

Das Spiel zu initialisieren und starten.



- **Attribute**
  - **-controller**: Referenz zur Klasse “Controller”
- **Methoden**
  - **+init():** Diese Funktion initialisiert das Spiel. Also herstellt sie alle benötigte Klassen.

    create items
    create icefloats with items
    sets neighbours of icefloats
    create characters asking GUI for their names
    create controller with everything above

  - **+start():** Diese Funktion startet das Spiel. Es ruft also die Funktion “void gameLoop()” von der Klasse “Controller” an.

8.1.2 Controller

- **Verantwortung**

Das nacheinander Kommen von Spielern zu organisieren. Beim Wechseln zwischen Spieler ist diese Klasse auch verantwortlich für die Schneestürme. Daneben soll er das Ergebnis des Spieles zu verwalten.

- **Attribute**
  - **- icefloats: List<IceFloat>**: Referenzlist zu den Eisschollen.
  - **- characters: List<Character>:** Referenzlist zu den Spielern.
  - **- isWon: bool**: Flag, um der Sieg des Spieles zu melden.
  - **- isLost: bool**: Flag, um das Verlieren des Spieles zu melden.
  - **- rocketPartsPickedUp: int** Die Anzahl der ausgegraben Teile der Leuchtpistole.
  - **- p: PolarBear** Der Eisbär im Eisfeld

- **Methoden**
  - **+Controller(characters: List<Characters>, icefloats: List<IceFloats>)**: Generiert eine Instanz, die die schon erzeugten Charakters und Eisschollen übernimmt.
  - **+gameLoop():** Diese Funktion ist verantwortlich für die Organisation der Zügen von Spielern, und nach jeder Runde ruft einige Funktionen, wie “snowStorm()” oder weckt den Eisbär.

```
while !isWon and !isLost do
    for character in characters do
        for i:=1 to 4 do
             validateActions(character)
             read actionId, parameters
             action = createAction(actionId, parameters)
             action.performAction()
        end
    end
    p.wake()
    snowStorm()
    for icefloat in icefloats do
        icefloat.endTurn()
    end
end
```

  - **-snowStorm():** Diese Funktion wählt einige Eisschollen 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.

```
for icefloat in icefloats do
if random%10 > 5 then icefloat.addSnow()
end
```

- **-validateActions(ch: Character):** Diese Funktion überprüft, welche Aktionen der Spieler in seinem Zug machen kann.

```
if ch.getIceFloat().snowLevel() > 0 then
                ch.AddAction(ShovelAction.class)
end
if ch.getIceFloat().getNeighbours().size() > 0 then
               ch.AddAction(MoveAction.class)
end
if canWin() then
               ch.AddAction(BuildRocketAction.class)
end
if ch.getIceFloat().hasItem() then
               ch.AddAction(PickupAction.class)
end
```

- **-createAction(index: int, parameters: Object[]): Action** Der GUI liefert uns den Index der ausgewählten Aktion und die Parameter, die dazu nötig sind. In dieser Methode erstellen wir die Aktion, die wir möchten.
- **-canWin(): bool** Überprüft, ob alle Spieler auf einer Eisscholle stehen, und dass die Anzahl der ausgegrabenen Teilen größer als 3 ist.

```
for icefloat in icefloats do
     if icefloat.playersHere() < characters.size() then
             return false
     else if  icefloat.playersHere() = characters.size()
          then
     if rocketPartsPickedUp >=3 then
                 return true
end
```

- **+rocketPartPickedUp():** Inkrementiert den Zähler.
- **+lose():** Wegen des Funktionsaufrufes wird das Spiel beendet und bekannt gegeben, dass die Spieler verloren haben.
- **+win():** Wegen des Funktionsaufrufes wird das Spiel beendet und bekannt gegeben, dass die Spieler gewonnen haben.

8.1.3 Character

- **Verantwortung**

Repräsentiert den Spieler im Spiel. Enthaltet die Arbeiten, die der Charakter leisten kann. Diese Klasse ermöglicht auch die Nachverfolgung und Änderung der Körpertemperatur des Charakters. Diese Klasse enthält auch die Gegenstände, die der Spieler aufgenommen hat, und hat einige Funktionen um zu bestimmen, ob der Charakter dazu fähig ist, bestimmte Tätigkeiten durchzuführen. Die Klasse ist abstrakt, es macht nur Sinn, eine konkrete Charakterentität, z.B. ein Eskimo zu instanzieren.

- **Attribute**
  - **#position: IceFloat**: Die Eisscholle, auf der der Character sich befindet.
  - **#items**: **List<Item>** Referenzlist auf den Gegenständen, die der Character besitzt.
  - **#actions: List<Action>** Eine Collection von Aktionen, die der Character benutzen kann.
  - **-temp: int** Repräsentiert die Körpertemperatur von dem Character.
  - **-name: String:** Ein beliebiger Name, um den Spieler zu identifizieren.

- **Methoden**
  - +Character(name: String, startPosition: StableIceFloat): Erstellt eine Instanz der Klasse.
  - **+modifyTemp(value: int)**: Modifiziert die Körpertemperatur des Characters um "value". Das wird typischerweise beim Schneesturm und bei der Aufnahme von Essen angerufen. Überprüft, ob der Spieler noch leben kann, ansonsten ruft Controller.lose() an.

```
temp := temp + value
if temp = 0 then
        Controller.lose()
```

  - **+addAction(action: Action)**: Die Aktion im Parameter wird zum Character gegeben, falls es aber schon in der Liste ist und es nicht eine BuildTentAction ist, dann nicht.

```
if action is instanceof(TentAction) then
     actions.Add(action)
else
     if actions.Contains(instanceof(action)) then
            return
     else
            actions.Add(action)
```

  - +**removeAction(action: Action):** Die Aktion im Parameter wird vom Character weggenommen nach dem Aufruf der Aktion.
  - **+canSwim(): bool** Gibt bekannt, ob der Character in einem Loch schwimmen kann.
  - **+canRescue(): bool** Gibt bekannt, ob der Spieler andere Spieler retten kann, die in ein benachbartes Loch gefallen sind.
  - **+getIceFloat(): IceFloat** Gibt den Eisscholle zurück, auf dem der Character steht.
  - **+canFastShovel(): bool** Gibt bekannt, ob der Spieler eine Schaufel hat.

```
for all item in items do
      if item.canFastShovel() then
            return true
end
return false
```

  - **+addItem(item: Item)**: Durch diese Funktion können Gegenstände zum Lager des Charakters hinzugefügt werden.

- **+removeItem(item: Item)**: Durch diese Funktion können Gegenstände vom Lager des Charakters entnommen werden.
- **+setPosition(ice: IceFloat)**: Bewegt den Character zur Eisscholle. (Bis jetzt war der Name moveTo(), aber dieser neue Name gilt besser zur Funktionalität, die nicht geändert wurde.)
- **+buildTent():** Der Charakter benutzt ein Item der als ein Zelt aufgebaut werden kann.

8.1.4 Researcher

- **Verantwortung**

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

- **Basisklasse**

Character

- **Methoden**
  - **+Character(name: String, startingPosition: StableIceFloat)** Erstellt eine Instanz der Klasse, aber nimmt auch eine CheckAction zu den Aktionen auf.

8.1.5 Eskimo

- **Verantwortung**

Einen solchen Charakter zu realisieren, der Iglus aufbauen kann.

- **Basisklassen**

Character

- **Methoden**
  - **+Character(name: String, startingPosition: StableIceFloat)** Erstellt eine Instanz der Klasse, aber nimmt auch eine BuildAction zu den Aktionen auf.

8.1.6 PolarBear

- **Verantwortung**

Eine solche Entität zu modellieren, die auf Eisschollen treten kann und wenn er Spieler trifft, sie tötet.

- **Attribute**
  - **-position: IceFloat** Die Eisscholle des Eisbären
- **Methoden**
  - **+wake():** Am Ende jeder Runde erwacht der Eisbär und bewegt sich in eine zufällige Richtung. Falls er mit Spielern auf einer Eisscholle landet, und sie nicht von einem Iglu geschützt sind, frisst er sie auf, und die Spieler sterben. Das Spiel wird verloren.

```
nextPos := random() % count(neighbours)
position := neighbours[nextPos]
if position.characters.size() > 0 then
          if !position.isBearProof() then
               Controller.lose()
```

8.1.7 Item

- **Verantwortung**

Das soll für einen abstrakten Gegenstand entsprechen, das die verschiedene Eigenschaften der einzelne Gegenstände beschreibt.

- **Methoden**
  - **+onPickup(ch: Character):** Sagt, was passiert, wenn ein Character diesen Gegenstand aufnimmt. Es gibt es dem Spieler weiter.
  - **+onUse(ch: Character):** Sagt, was passiert, wenn ein Character diesen Gegenstand benutzt. Nichts passiert.
  - **+canSwim(): bool** Sagt, ob der Character, der dieser Gegenstand hat, aus einen Loch schwimmen kann. Kehrt mit falsch zurück.
  - **+canFastShovel(): bool** Sagt, ob der Character mehr Schnee mit einem Aktion schaufeln kann. Kehrt mit falsch zurück.
  - **+canRescue(): bool** Sagt, ob der Character einen anderen Character retten kann, der in eine Loch eingefallen ist. Kehrt mit falsch zurück.
  - **+canBuildAsTent(): bool** Sagt, ob der Character ein Zelt auf seiner Eisscholle mit Hilfe von diesen Item aufstellen kann. Kehrt mit falsch zurück.

8.1.8 RocketPart

- **Verantwortung**

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

- **Basisklasse**

Item

- **Methoden**
- **+onPickup(ch: Character):** Sagt, was passiert, wenn ein Character diesen Gegenstand aufnimmt. Er signalisiert den Controller, dass ein Teil ausgegraben wurde.

8.1.9 DiverSuit

- **Verantwortung**

Es soll ermöglichen, dass ein Spieler ein Fall in ein Loch überlebt.

- **Basisklasse**

Item

- **Methoden**
  - **+canSwim(): bool** Kehrt mit wahr zurück, so wissen wir, dass wenn man dieses Item besitzt, man Schwimmen kann.

8.1.10 Shovel

- **Verantwortung**

Falls ein Character diesen Gegenstand besitzt, dann kann er 2 Schnee mit einer Action schaufeln.

- **Basisklasse**

Item

- **Methoden**
  - **+canFastShovel(): bool** Sagt, ob der Character mehr Schnee mit einem Aktion schaufeln kann. Kehrt mit wahr zurück.

8.1.11 BreakableShovel

- **Verantwortung**

Falls ein Character diesen Gegenstand enthält, dann kann er 2 Schnee mit einem Action schaufeln. Dieser Gegenstand geht nach 3 Benutzung kaputt.

- **Basisklasse**

Shovel

- **Methoden**
  - **+onUse(ch: Character)**: Die Schaufel wird einmal benutzt. Falls es schon dreimal benutzt wurde, entfernt er sich von dem Charakter der es benutzt.

- **Attribute**
  - **-uses: int** wie oft man die Schaufel noch benutzen kann. Es wird mit 3 initialisiert.

8.1.12 Rope

- **Verantwortung**

Falls ein Charakter diesen Gegenstand enthält, dann kann er einen anderen Character aus einem benachbarten Loch retten.

- **Basisklasse**

Item

- **Methoden**
  - **+canRescue(): bool** Kehrt mit wahr zurück.

8.1.13 Food

- **Verantwortung**

Beim Aufnehmen dieses Gegenstandes erhöht sich die Körpertemperatur des Spielers um 1.

- **Basisklasse**

Item

- **Methoden**
  - **+onPickup(ch: Character):** Inkrementiert den Körpertemperatur mit 1. Dann wird das Objekt zerstört, also nicht an den Charakter weitergegeben.

8.1.14 Tent

- **Verantwortung**

Dieser Gegenstand kann benutzt werden, um ein Zelt auf einer Eisscholle zu bauen.

- **Basisklasse**

Item

- **Methoden**
  - **+onUse(ch:Character):** Entfernt sich von dem Charakter, der in benutzt. Er entfernt auch eine BuildTentAction von dem Charakter. Der Charakter selbst kann dann ein Zelt auf seiner Eisscholle erbauen.
  - **+onPickup(ch:Character):** Gibt dem Charakter eine BuildTentAction hinzu und addiert dem Spieler sich selbst.
  - +canBuildAsTent(): bool: Kehrt mit wahr zurück.

8.1.15 IceFloat

- **Verantwortung**

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

- **Attribute**
  - **#neighbours: List<IceFloat>** Eine Sammlung der benachbarten Eisschollen
  - **#item: Item** der Gegenstand, der in der Eisscholle gefroren ist. Es kann “leer” sein, schon von Anfang an oder falls ein Spieler ihn schon aufgenommen hat.
  - **#characters: List<Character>** Die Menge der Charaktere, die auf dieser Eisscholle stehen.
  - **#iglu: bool** Durch dieses Attribut kann man sagen, ob es auf der Eisscholle ein Iglu gibt.
  - **#tent: bool** Durch dieses Attribut kann man sagen, ob es auf der Eisscholle ein Zelt gibt.
  - **#snowLevel: int** Speichert die Menge von Schnee auf der Eisscholle.

- **Methoden**
  - **+ *stepOn(ch: Character)*:** Die Methode behandelt die Anfrage eines Charakters, der auf diese Eisscholle treten möchte.
  - **+ removeCharacter(ch: Character):** Das entfernt der Charakter von einer Eisscholle.
  - **+ removeSnow(layerCount: int):** Die bestimmte Anzahl von Schichten werden von der Eisscholle entfernt, aber die Menge des Schnees wird nie negativ.

- **+ addSnow():** Eine Schicht Schnee wird auf die Eisscholle gelegt. Sie wird von dem Schneesturm gerufen. Wenn ein Zelt oder ein Iglu auf der Eisscholle ist, wird es zerstört, sonst verlieren alle Spieler, die hier stehen eine Einheit von Körpertemperatur.

```
snowLevel = snowLevel + 1
if !iglu and !tent then
           for character in characters do
                  character.modifyTemp(-1)
           end
end
```

- **+ getCapacity(): int:** Die Methode gibt ihre Kapazität bekannt.
- **+ buildIgloo():** Die Methode erschafft ein Iglu.
- **+ removeItem(): Item**: Durch dieser Methode wird der Gegenstand von der Eisscholle zu dem Charakter übernommen.
- **+ canRescueFromHere(): bool:** Es bestimmt, ob ein Charakter von einem Loch mit einem Seil aufgezogen werden kann.

```
for character in characters do
           if character.canRescue() then
                        return true
end
return false
```

- **+ playersHere(): int:** Es gibt bekannt, wie viele Spieler auf der Eisscholle stehen.
- **+ buildTent():** Baut ein Zelt auf der Eisscholle auf.
- **+ endTurn():** Am Ende jeder Runde wird es aufgerufen, und dann wird die Zelt zerstört.
- **+snowLevel(): int** Kehrt mit der Menge des Schnees auf der Eisscholle zurück.
- **+hasItem(): bool**: Kehrt mit wahr zurück, falls die Eisscholle einen Gegenstand hat, mit falsch sonst.
- **+isBearProof(): bool:** Wenn ein Eisbär auf die Eisscholle tritt, hängt davon ab, ob die Spieler das Spiel verlieren.
- **+getNeighbours(): List<IceFloat>:** Kehrt mit den Nachbarn der Eisscholle zurück.

8.1.16 StableIceFloat

- **Verantwortung**

Die allgemeine Eisscholle, der alle Characters halten kann, ohne umzukippen.

- **Basisklasse**

IceFloat

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

8.1.17 UnstableIceFloat

- **Verantwortung**

Nach einem bestimmten Anzahl von Character kippt es um.

- **Basisklasse**

IceFloat

- **Attribute**
  - **-capacity: int**: Durch diesem Attribut kann man beobachten, wie viel Charakter der Eisscholle fördern kann.

- **Methoden**
  - **+ stepOn(character: Character)**: Das akzeptiert den Character auf diese Eisscholle oder falls schon zu viele Spieler auf der Eisscholle stehen, kippt es um und die Spieler verlieren

```
IceFloat prev := character.getIceFloat()
prev.removeCharacter(character)
characters.add(character)
character.setPosition(this)
if  playersHere() > capacity then
            Controller.lose()
end
```

8.1.18 Hole

- **Verantwortung**

Falls man darauf tritt, fällt er ins Wasser.

- **Basisklasse**

IceFloat

- **Methoden**
  - **+ stepOn(character: Character)**: Der Charakter fällt ins Wasser. Falls er schwimmen kann, kehrt er zurück auf die vorige Eisscholle. Sonst muss er gerettet werden, dann bewegt er sich zu seinem Retter. Falls er auch nicht gerettet werden kann, ertrinkt er und das Spiel endet mit einer Niederlage.

```
if character.canSwim()
          return
for ice in neighbours do
              if ice.canRescueFromHere() then
                          ice.stepOn(character)
                          return
              end
end
IceFloat prev := character.getIceFloat()
prev.removeCharacter(character)
characters.add(character)
character.setPosition(this)
Controller.lose()
```

8.1.19 Action

- **Verantwortung**

Abstrakte Klasse, ihre Kinder realisieren die einzelne verschiedene Aktionen (Arbeiten) der Spieler.

- **Attribute**
  - **# ch**: Speichert ein Charakter, zu wem die gegebene Aktion gehört.
- **Methoden**
  - **+ *performAction()*:** Durchführt den Action.

8.1.20 ShovelAction

- **Verantwortung**

Schaufelt Schnee auf dem gegebenen Eisscholle. Kann ausgeführt werden, falls auf der Eisscholle des Spielers mindestens eine Einheit Schnee liegt.

- **Basisklasse**

Action

- **Methoden**
  - **+ ShovelAction(character: Character)**: Instanziert eine neue Schaufelaktion.
  - **+ performAction()**: Durchführt den Action.

```
if ch.canFastShovel then
              ch.position.removeSnow(2)
else
              ch.position.removeSnow(1)
```

8.1.21 MoveAction

- **Verantwortung**

Bewegt den Character auf einen anderen Eisscholle. Kann ausgeführt werden, falls die Eisscholle des Spielers mindestens einen Nachbar hat.

- **Basisklasse**

Action

- **Attribute**
  - **-target: IceFloat** wohin sich der Charakter bewegen möchte
- **Methoden**
  - + MoveAction(character: Character, ice: IceFloat): Instanziert eine neue Bewegungsaktion.
  - **+ performAction()**: Durchführt den Action.

8.1.22 BuildRocketAction

- **Verantwortung**

Baut die Rakete zusammen, falls alle Teile auf einem Scholle sind. Kann ausgeführt werden, falls alle Spieler auf einer Eisscholle sind und alle Teile ausgegraben wurden.

- **Basisklasse**

Action

- **Methoden**
  - **+BuildRocketAction(character: Character):** Instanziert eine neue Raketebauaktion.
  - **+ performAction():** Durchführt den Action.

8.1.23 PickupAction

- **Verantwortung**

Nimmt den Gegenstand von dem gegebenen Eisscholle auf. Kann ausgeführt werden, falls auf der Eisscholle des Spielers kein Schnee liegt und es ein Gegenstand beinhaltet.

- **Basisklasse**

Action

- **Methoden**
  - **+ PickupAction(character: Character):** Instanziert eine neue Aufnahmeaktion.
  - **+ performAction():** Durchführt den Action.

8.1.24 BuildAction

- **Verantwortung**

Baut einen Iglu auf den gegebenen Eisscholle. Kann immer ausgeführt werden, die Möglichkeit darauf wird nie von einem Charakter entfernt.

- **Basisklasse**

Action

- **Methoden**
  - + BuildAction(character: Character): Instanziert eine neue Bauaktion.
  - **+ performAction():** Durchführt den Action.

8.1.25 BuildTentAction

- **Verantwortung**

Baut ein Zelt auf der gegebenen Eisscholle. Für die Verfügbarkeit sind die einzelnen Zelte verantwortlich.

- **Basisklasse**

Action

- **Methoden**
  - **+ BuildTentAction(character: Character):** Instanziert eine neue Zeltbauaktion.
  - **+ performAction():** Durchführt den Action.

8.1.26 CheckAction

- **Verantwortung**

Überprüft den Kapazität einer Eisscholle. Kann immer ausgeführt werden, die Möglichkeit darauf wird nie von einem Charakter entfernt.

- **Basisklasse**

Action

- **Attribute**
  - **-target: IceFloat** deren Kapazität ermittelt wird
- **Methoden**
  - + CheckAction(character: Character, target: IceFloat): Instanziert eine neue Überprüfungsaktion.
  - **+ performAction():** Ermittelt die Kapazität der Eisscholle, der im Parameter des Konstruktors übergegeben war.

8.2 Die detaillierte Pläne der Testfälle und ihre Beschreibung auf der Sprache des Testes

8.2.1 Player steps on stable Ice-float

- **Beschreibung**

Die erste Test der Bewegungsarbeit von einem Spieler ist durch diesen Test realisiert. Der Fall wird getestet, in dem ein Spieler auf eine stabile Eisscholle zu treten probiert.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Die Bewegung soll erfolgreich sein. Fehler kann bei der Bewegung der Spieler vorkommen.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
</config>
<test>
<moveaction character=”testBunny” icefloat=”1” />
<saveOutput filename=”test1Out.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
```

```
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

8.2.2 Player steps on unstable Ice-float

- **Beschreibung**

Diese Test realisiert die Bewegungsarbeit eines Spielers, wenn er auf eine instabile Eisscholle probiert zu treten.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

In diesem Test wird beide Fälle simuliert. Falls die Eisscholle umkippt, soll das Spiel beendet werden, andernfalls soll die Bewegung erfolgreich sein. Am wahrscheinlichsten wird Fehler bei der umkippen von der instabilen Eisscholle.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>unstable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
```

```
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>2</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
</config>
<test>
<moveaction character=”testBunny1” icefloat=”1” />
<saveOutput filename=”test2Out1.xml” />
<moveaction character=”testBunny2” icefloat=”1” />
<saveOutput filename=”test2Out2.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

Die erste Ausgangsdatei soll so aussehen:

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>unstable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
```

```
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>2</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

Die zweite Ausgangsdatei soll so aussehen:

```
<icefield state=”lost”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>unstable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

8.2.3 Player steps on Hole

- **Beschreibung**

Die dritte Art der Bewegungsarbeit soll durch diesen Test passieren. Diese Test simuliert die Ereignis wenn ein Spieler in einem Loch tritt.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Drei Sachen können passieren, alle drei werden getestet. Erste ist das der Spieler ertrunken hat, in diesem Fall soll der Spiel beendet werden. Zweitens kommt die Möglichkeit, wenn der Spieler ein Taucheranzug hat. Am letzten kommt der Fall, wenn ihn ein anderer Spieler mit einem Seil rettet. In dem zweiten und dritten Fall soll der Spieler überleben. Erwartete Platzt der Fehler befindet sich in dem Fall wenn der Spieler gerettet wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
	<icefloat>
		<type>stable</type>
		<capacity></capacity>
		<item></item>
		<snowcount>0</snowcount>
		<igloo>false</igloo>
		<tent>false</tent>
	</icefloat>
	<icefloat>
		<type>hole</type>
		<capacity>0</capacity>
		<item></item>
		<snowcount>0</snowcount>
		<igloo>false</igloo>
		<tent>false</tent>
	</icefloat>
	<icefloat>
		<type>stable</type>
		<capacity></capacity>
		<item></item>
		<snowcount>0</snowcount>
		<igloo>false</igloo>
		<tent>false</tent>
	</icefloat>
	<icefloat>
		<type>stable</type>
		<capacity></capacity>
		<item></item>
		<snowcount>0</snowcount>
		<igloo>false</igloo>
		<tent>false</tent>
	</icefloat>
</icefloats>
<characters>
	<character>
		<type>Eskimo</type>
		<name>testBunny1</name>
		<position>0</position>
		<bodytemp>5</bodytemp>
	</character>
<character>
		<type>Eskimo</type>
		<name>testBunny2</name>
```

```
            <position>3</position>
            <bodytemp>5</bodytemp>
<items>
                  <item  type=”DiverSuit”/>
<item  type=”Rope”/>
            </items>
      </character>
</characters>
</config>
<test>
<moveaction character=”testBunny2” icefloat=”1” />
<saveOutput filename=”test3Out1.xml” />
<moveaction character=”testBunny1” icefloat=”1” />
<saveOutput filename=”test3Out2.xml” />
<moveaction character=”testBunny2” icefloat=”2” />
<moveaction character=”testBunny1” icefloat=”1” />
<saveOutput filename=”test3Out3.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

Die erste Ausgangsdatei soll so aussehen:

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>hole</type>
            <capacity>0</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
```

```
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>3</position>
            <bodytemp>5</bodytemp>
<items>
                  <item  type=”DiverSuit”/>
<item  type=”Rope”/>
            </items>
      </character>
</characters>
</config>
</icefield>
```

Die zweite Ausgangsdatei soll so aussehen:

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>hole</type>
            <capacity>0</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
```

```
            <name>testBunny1</name>
            <position>3</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>3</position>
            <bodytemp>5</bodytemp>
<items>
                  <item  type=”DiverSuit”/>
<item  type=”Rope”/>
            </items>
      </character>
</characters>
</config>
</icefield>
```

Die dritte Ausgangsdatei soll so aussehen:

```
<icefield state=”lost”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>hole</type>
            <capacity>0</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>1</position>
```

```
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>2</position>
            <bodytemp>5</bodytemp>
<items>
                  <item  type="DiverSuit"/>
<item  type="Rope"/>
            </items>
      </character>
</characters>
</config>
</icefield>
```

8.2.4 Player picks up food

- **Beschreibung**

Der Spieler hebt ein Lebensmittel auf.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Der Temperatur des Spielers soll mit Eins inkrementiert werden. Ein Fehler kann sein, dass der Temperatur nicht erhöht wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item>Food</item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<pickupAction character="testBunny"/>
<saveOutput filename="test4Out.xml" />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state="ongoing">
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
```

```
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>3</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

8.2.5 Player picks up item other than food

- **Beschreibung**

Der Spieler hebt ein Gegenstand außer Lebensmittel auf.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Der Gegenstand soll in dem Lager des Spielers gehen. Ein Fehler könnte sein, wenn der Gegenstand in dem Lager nicht erscheint.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item>Rope</item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<pickupAction character=”testBunny”/>
<saveOutput filename=”test5Out.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
```

```
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
<items>
<item  type="Rope"/>
            </items>
      </character>
</characters>
</config>
</icefield>
```

8.2.6 Player shovels snow

- **Beschreibung**

Der Spieler entfernt Schnee von Eisscholle.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Beide Fällen sollen getestet werden. Falls der Spieler ein Schaufel hat, dann soll er die Menge der Schnee mit Zwei dekrementieren, andernfalls soll sie mit eins dekrementiert werden. Ein Möglichkeit des schlechten Verhaltens ist, falls nicht die korrekte Menge von Schnee geschaufelt wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>10</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<shovelAction character="testBunny"/>
<saveOutput filename="test6Out1.xml" />
<addItemToCharacter item="Shovel" character="testBunny"/>
<shovelAction character="testBunny"/>
<saveOutput filename="test6Out2.xml" />
</test>
```

```
</icefield>
```

- **Erwartete Ausgang**

Die erste Ausgangsdatei soll so aussehen:

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>9</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

Die zweite Ausgangsdatei soll so aussehen:

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>7</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
<items>
<item  type=”Shovel”/>
            </items>
      </character>
</characters>
</config>
</icefield>
```

8.2.7 Player builds rocket

- **Beschreibung**

Der Spieler basteln die Melder-rakete zusammen.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Der Spiel soll gewonnen sein. Eine mögliche Fehler ist, dass das Spiel nicht gewonnen wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
<items>
                  <item  type=”RocketPart”/>
<item  type=”RocketPart”/>
<item  type=”RocketPart”/>
            </items>
      </character>
</characters>
</config>
<test>
<buildRocketAction character=”testBunny”/>
<saveOutput filename=”test7Out.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state=”won”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
<items>
                  <item  type=”RocketPart”/>
<item  type=”RocketPart”/>
<item  type=”RocketPart”/>
            </items>
      </character>
</characters>
```

```
</config>
</icefield>
```

8.2.8 Researcher examines Ice-float

- **Beschreibung**

Der Polarforscher sieht an wie viele Personen die Eisscholle fördern kann.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Mit der gesuchten Anzahl soll der Polarforscher informiert werden. Vielleicht kommt ein solchen Fehler vor wenn nicht die korrekte Anzahl angeschaut wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Researcher</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<checkAction character=”testBunny” float=”0”/>
</test>
</icefield>
```

- **Erwartete Ausgang**

Der Spieler soll informiert werden über den Kapazität der Eisscholle. In dem aktuellen Programm soll diese Information auf der standardisierte Ausgang (Console) beobachtbar sein.

8.2.9 Eskimo builds Igloo

- **Beschreibung**

Ein Eskimo baut ein Iglu auf.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Diese Arbeit soll erfolgreich sein. Ein solche Fehler kann vorkommen, wenn der Igloo nicht gebaut würde.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
```

```
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<buildAction character="testBunny"/>
<saveOutput filename="test9Out.xml" />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state="ongoing">
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>true</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

8.2.10 Snowstorm appears

- **Beschreibung**

Ein Schneesturm taucht auf.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Die Funktionen des Schneesturmes soll getesten werden. Diese Funktionen beinhalten: Schnee legen, Iglu zerstören, Temperatur niedrigen. Eine gute Möglichkeit Fehler zu machen ist, wenn ein Spieler in Iglu ist, in diesem Fall soll die Temperatur nicht erniedrigt werden.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
```

```
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>true</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>1</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<snowstorm>
      <ifloat id=”0”/>
      <ifloat id=”1”/>
</snowstorm>
<saveOutput filename=”test10Out.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>1</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>1</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
```

```
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>1</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>1</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

8.2.11 Polar Bear Moves

- **Beschreibung**

Die Bewegung des Eisbären soll durch diesen Test simuliert werden.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Die Bewegung soll erfolgreich sein. Die Funktionalität, wenn der Bär einen Spieler isst, soll auch getestet werden. Wenn der Spieler in Iglu ist, soll er nicht gefressen werden.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>true</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>5</bodytemp>
      </character>
```

```
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
<polarbear position=”2” />
</config>
<test>
<polarBearMove direction=”left”/>
<saveOutput filename=”test11Out1.xml” />
<polarBearMove direction=”left”/>
<saveOutput filename=”test11Out2.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

Die erste Ausgangsdatei soll so aussehen:

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>true</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
```

```
      </character>
</characters>
<polarbear position=”1” />
</config>
</icefield>
```

Die erste Ausgangsdatei soll so aussehen:

```
<icefield state=”lost”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>true</igloo>
            <tent>false</tent>
      </icefloat>
      <icefloat>
            <type>stable</type>
            <capacity>2</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny1</name>
            <position>0</position>
            <bodytemp>5</bodytemp>
      </character>
<character>
            <type>Eskimo</type>
            <name>testBunny2</name>
            <position>1</position>
            <bodytemp>5</bodytemp>
      </character>
</characters>
<polarbear position=”0” />
</config>
</icefield>
```

8.2.12 Breakable Shovel breaks

- **Beschreibung**

Ein zerbrechliche Schaufel wird zu viel benutzt, und dadurch zerstört.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Der Schaufel soll zerstört werden. Eine mögliche Fehler ist, dass der Schaufel nicht zerstört wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>10</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
<test>
<addItemToCharacter item=”BreakableShovel” character=”testBunny”/>
<shovelAction character=”testBunny”/>
<shovelAction character=”testBunny”/>
<shovelAction character=”testBunny”/>
<saveOutput filename=”test12Out.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity></capacity>
            <item></item>
            <snowcount>4</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
</config>
</icefield>
```

8.2.13 Player builds tent

- **Beschreibung**

Ein Spieler, der schon ein Zelt aufgenommen hat, baut ein Zelt auf.

- **Kontrollierte Funktion, erwartbarer Platz des Fehlers**

Ein Zelt soll auf die Eisscholle gelegt werden, auf der der Spieler sich befindet. Mögliche Fehler kann zum Beispiel sein, falls das Zelt nicht aufgebaut wird.

- **Eingang**

```
<icefield>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>false</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
<items>
                  <item  type=”Tent”/>
            </items>
      </character>
</characters>
</config>
<test>
<buildTentAction character=”testBunny”/>
<saveOutput filename=”test13Out.xml” />
</test>
</icefield>
```

- **Erwartete Ausgang**

```
<icefield state=”ongoing”>
<config>
<icefloats>
      <icefloat>
            <type>stable</type>
            <capacity>1</capacity>
            <item></item>
            <snowcount>0</snowcount>
            <igloo>false</igloo>
            <tent>true</tent>
      </icefloat>
</icefloats>
<characters>
      <character>
            <type>Eskimo</type>
            <name>testBunny</name>
            <position>0</position>
            <bodytemp>2</bodytemp>
      </character>
</characters>
```

```
</config>
</icefield>
```

8.3 A tesztelést támogató programok tervei

Für das Testen wird eine einfache python Script benutzt. Erstens, muss man einen Test im Prototyp durchführen, und die Ausgangsdatei (XML) speichern. Dann kann man den folgenden Befehl ins Kommandozeile eintippen:

```
python test.py <reference> <actual>
```

Die Argumente sind die Folgende:

`<reference>`: Die Pfad zur erwarteten XML Datei.
`<actual>`: Die Pfad zur Ausgangsdatei der Ablauf des Programms.

Die Ausgang der Script kann SUCCES oder FAILED sein. Falls es FAILED ist, wird die Unterschied zwischen der zwei Dateien ausgeschrieben. Die reihenfolge der Kindelemente wird nicht überprüft.

Zum Beispiel:

```
C:\dev\xmldiff>python test.py ref.xml actual.xml
FAILED
[UpdateAttrib(node='/icefield[1]', name='state', value='lost')]
```

Also hier wurde statt <icefield state=”ongoing”> <icefield state=”lost”> geschrieben.

Um dieses Script laufen lassen zu können, muss man Python 3 installieren, und durch pip die folgende Bibliotheken herunterladen: `xmldiff, colorama.`