11. Spezifikation der graphischen Oberfläche

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

11. Spezifikation der graphischen Oberfläche

11.1 Das graphische Interface

Das Spiel begrüßt die Spieler mit dem folgenden Bildschirm. In diesem Sample haben wir schon 4 Spieler aufgenommen.

Die benutzten Ikons und Bilder für das Spiel sind die folgenden.

Schaufel 

Zerbrechliche Schaufel 

Taucheranzug 

Essen 

Iglu 

Seil 

Leichtpistole 

Zelt 

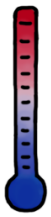
Loch 

Polarbär 

Forscher (Ikon) 

Eskimo (Ikon)

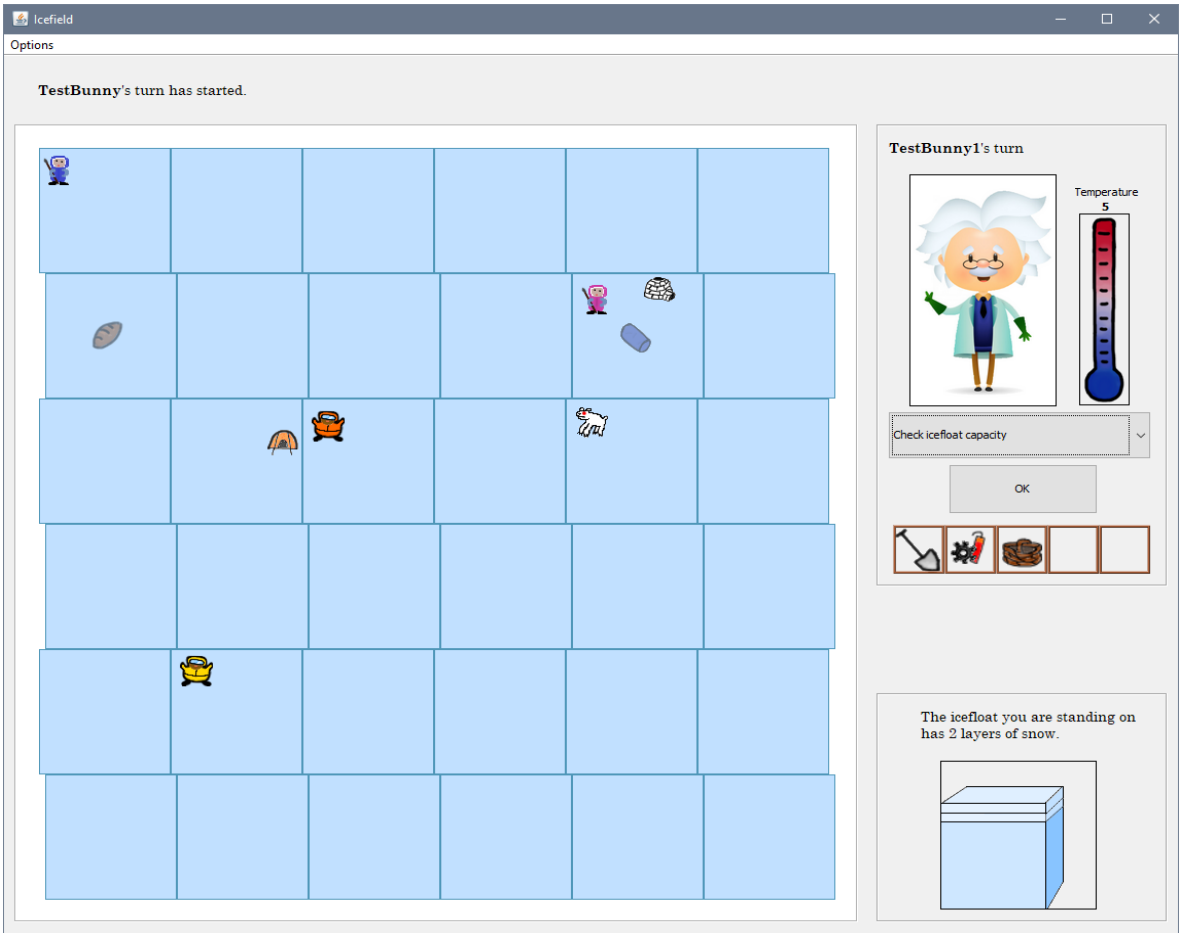
Thermometer



Forscher (rechts)

Eskimo (rechts)

Der Hauptschirm wird etwas so aussehen.

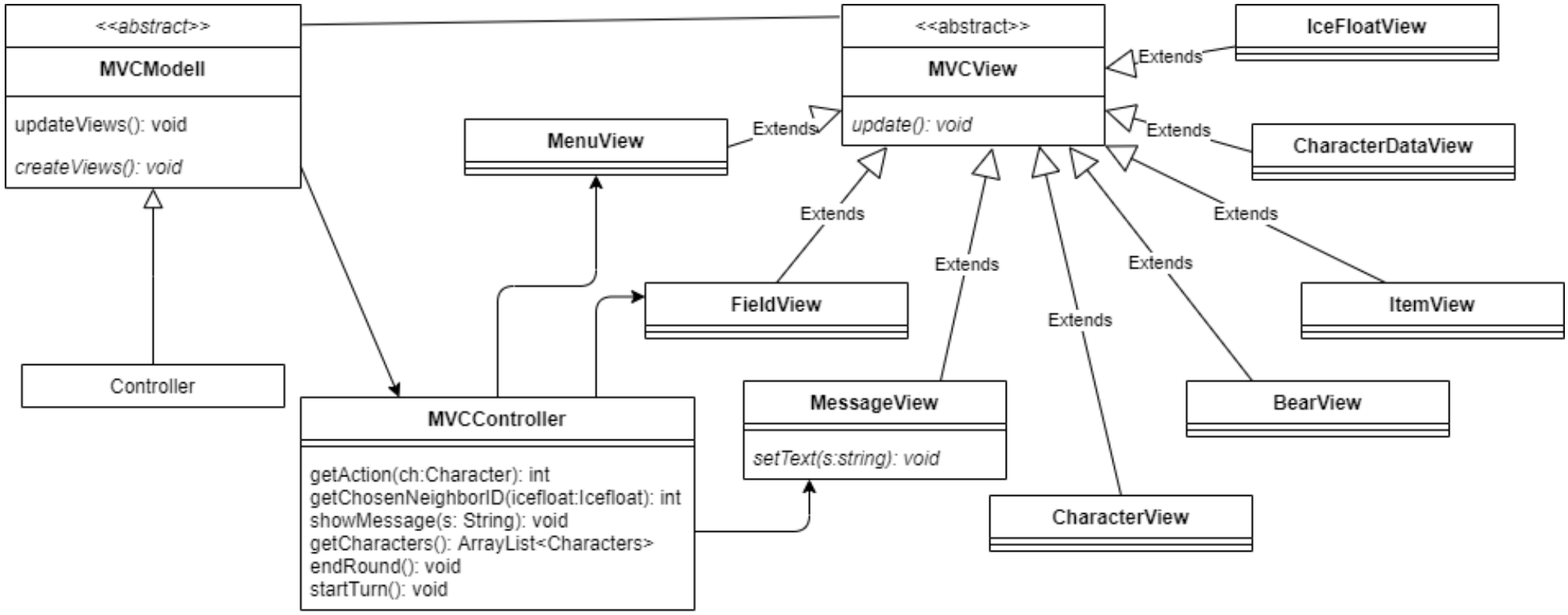


11.2 Architektur des graphischen Systems

11.2.1 Prinzip der Funktionalität der Oberfläche

Die Architektur unserer Oberfläche implementiert die MVC Architektur, und sie ist auf "push" basierend. Das bedeutet, dass unsere bisherige Klassen werden Teile des Modells. Unsere "Controller" Klasse steuert und sammelt die verschiedene Bauelemente des Modells. Also die einzige Beziehung zu der GUI-Architektur ist durch die "Controller" Klasse durch eine Vererbung von einem abstrakten Klasse. Die Darstellung ist einerseits durch "labels" (MenuView, FieldView, CharacterDataView, MessageView) organisiert, andererseits durch die Klassen(IceFloatView, CharacterView, BearView, ItemView).

11.2.2 Die Klassen-Architektur der Oberfläche



11.3 Aufzählung der graphischen Objekte

11.3.1 MVCModell

- **Verantwortung**

Modell der Aufgabe. Abstrakte Klasse.

- **Attribute**

-List<MVCView> views: Eine Kollektion aller instanzierte MVCViews.
-MVCController mvcController: Referenz auf dem Controller der MVC Architektur.

- **Methoden**
  - **+ updateViews()**: Zeichnet alles erneut aus (wenn etwas sich ändert).
  - **+ *createViews()***: Für jede Klasseninstanz, erstellt eine ansprechende View-Klasseninstanz.

11.3.2 Controller

- **Verantwortung**

Sorgt für die logische Hintergrund der Darstellung.

- **Basisklasse**
  - MVCModell
- **Methoden**
  - **+ createViews()**: Für jede Klasseninstanz, erstellt eine ansprechende View-Klasseninstanz.

11.3.3 MVCController

- **Verantwortung**

Implementiert die GUI Schnittstelle, und somit bietet eine Möglichkeit für den Programm, um die nötige informationen vom Benutzer zu bekommen.

- **Attribute**
  - **menuView: MenuView**: Referenz zur MenuView
  - **fieldView: FieldView**: Referenz zur MenuView
  - **messageView: MessageView**: Referenz zur MessageView

- **Interfaces**

GUI

- **Methoden**
  - \+ **showMessage(s: String)**: Schreibt die Nachricht auf das MessageView.
  - \+ **getAction(character: Character): int**: Fragt den Spieler, welche Aktion er ausführen lassen will. Gibt zurück die Nummer der Aktion.
  - \+ **getChosenNeighborID(icefloat: IceFloat): int**: Gibt zurück, auf welche Eisscholle der Spieler treten möchte.
  - \+ **getCharacters(): ArrayList<Character>**: Durch einem Pop-Up-Fenster erzeugt 3-6 Characters und gibt sie zurück.

+ **endRound()**: Behandelt das Ende der Runde (schreibt die dazugehörigen Information aus).
+ **startTurn()**: Behandelt das Starten der Runde eines Spielers (schreibt die dazugehörigen Information aus).

11.3.4 MVCView

- **Verantwortung**

Ein solches Interface, der die Auszeichnungen der einzelne Schnittstellen der Benutzeroberfläche behandeln kann.

- **Attributen**
  - **-MVCModell** modell: Referenz auf MVCModell
- **Methoden**
  - **+ update()**: Die Klasse, die sie ändert, zeichnet die ansprechende Instanz (wieder) aus.

11.3.5 MessageView

- **Verantwortung**

Sorgt für die Ausschreibungen der einzelne Nachrichten des Systems. z.B Wenn ein Researcher die Kapazität einer Eisscholle abfragt, odes wenn ein Schneesturm kommt, die Zustände des Spiels, usw.

- **Basisklasse**
  MVCView
- **Attribute**
  - **JLabel label**: in diesem Label wird die verschiedene Nachrichten ausgeschrieben
- **Methoden**
  - **setText(s: string):void:** Setter von Text der Label

11.3.6 CharacterDataView

- **Verantwortung**

Ein Label, der beschreibt die momentane Daten eines Spielers (der dran ist).

- **Basisklasse**
  MVCView
- **Attribute**
- **Methoden**

11.3.7 ItemView

- **Verantwortung**

Zeichnet das Bild der Gegenstand aus. Verschiedene Gegenstandstypen habe verschiedene Bilder.

- **Basisklasse**
  MVCView

- **Attribute**
  -path: String : Pfad zur Bilddatei.

- **Methoden**

11.3.8 BearView

- **Verantwortung**

Sorgt für die Auszeichnung des Eisbärs.

- **Basisklasse**
  MVCView

- **Attribute**
  -path: string: Die Zugriffsverzeichniss des Bildes wodurch das Bär auszeichnet wird.

- **Methoden**

11.3.9 CharacterView

- **Verantwortung**

Zeichnet die entsprechende Karakters anhand der gegebene Bilder aus.

- **Basisklasse**
  MVCView

- **Attribute**
  -path: string: Die Zugriffsverzeichniss des Bildes wodurch das Bär auszeichnet wird.

- **Methoden**

11.3.10 IceFloatView

- **Verantwortung**

Zeichnet die Gegenstände die an der Eisscholle sind: Zelt / Iglu / Item / Schneemenge.

- **Basisklasse**
  MVCView

- **Attribute**

  -iglooPath: string: Die Zugriffsweg des Bildes wodurch das Iglu auszeichnet wird.

  -tentPath: string: Die Zugriffsverzeichniss des Bildes wodurch das Zelt auszeichnet Wird.

- **Methoden**

11.3.11 MenuView

- **Verantwortung**

  Das Basisfenster des Spiels. Enthält auch ein Menu oben.

- **Basisklasse**
  MVCView

- **Attribute**

- **Methoden**

11.3.12 FieldView

- **Verantwortung**
  Gibt ein Rahmen des EisFeldes.
- **Basisklasse**
  MVCView

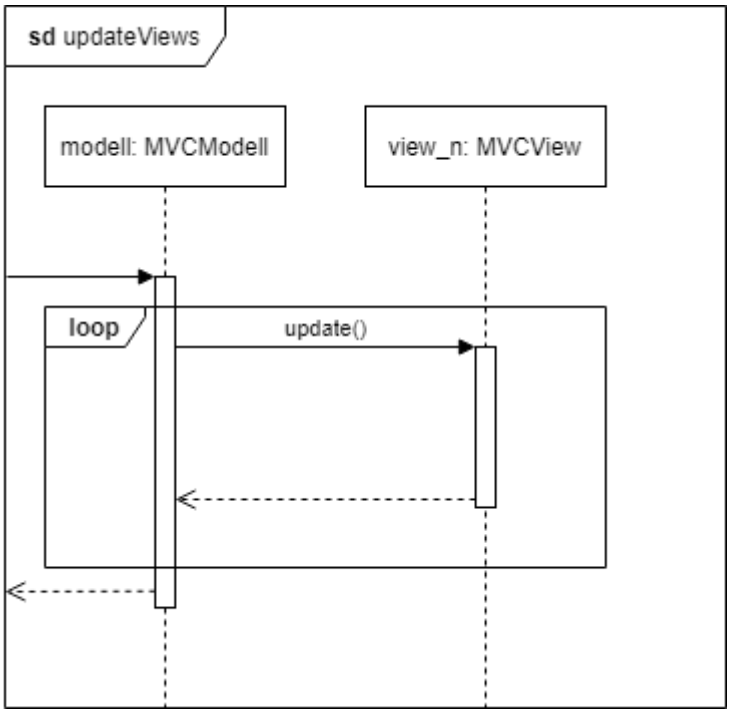
- **Attribute**

- **Methoden**

11.4 Beziehung mit dem Anwender-System

11.4.1 MVCModell

11.4.1.1 updateViews

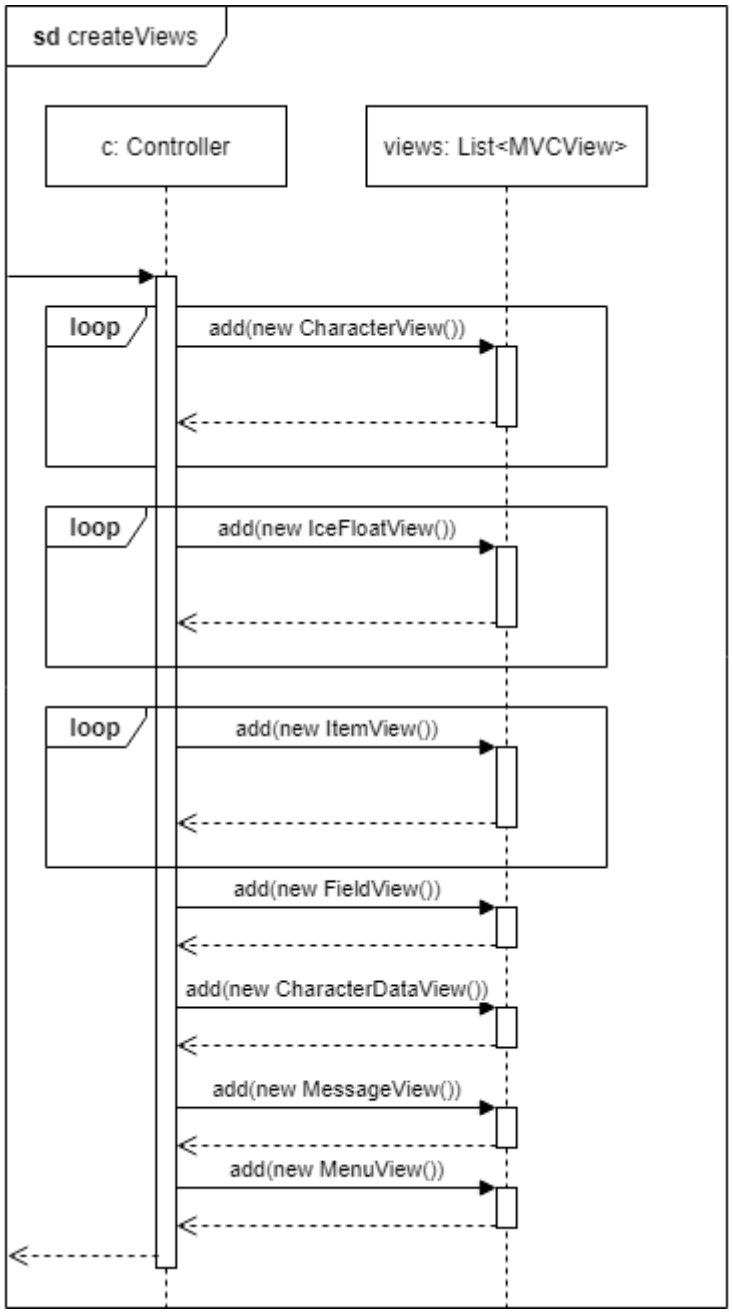


11.4.1.2 createViews

Die Methode createViews ist abstrakt, sie beinhaltet kein Code.

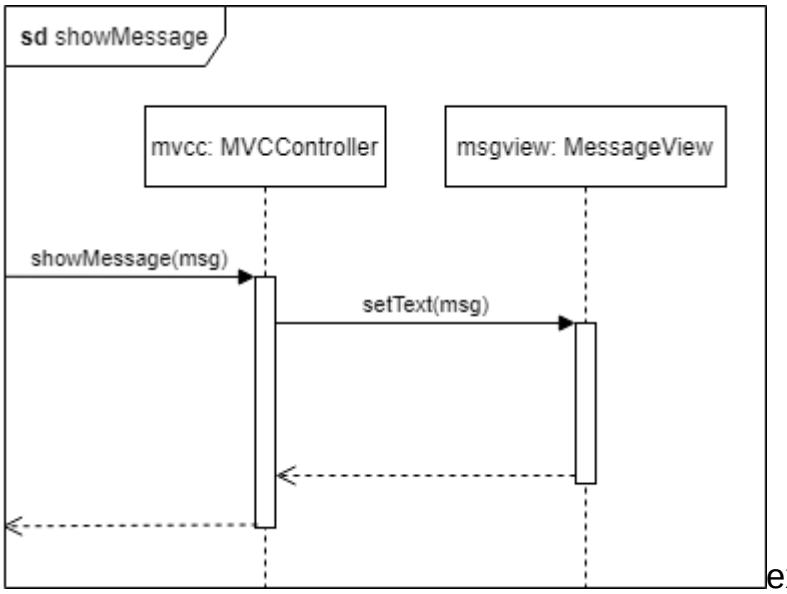
11.4.2 Controller

11.4.2.1 createViews



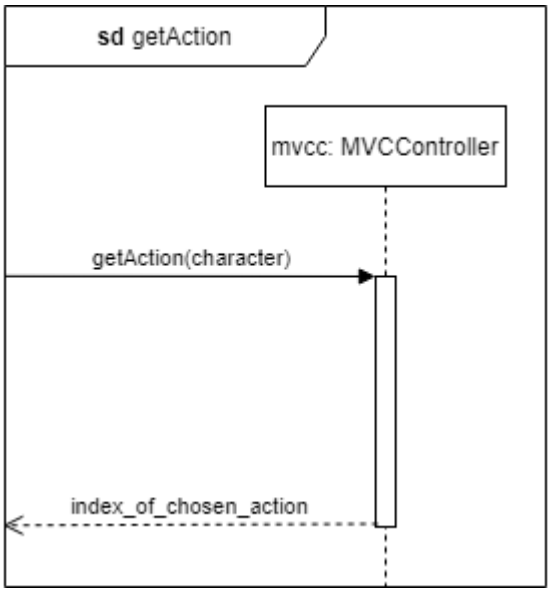
11.4.3 MVCController

11.4.3.1 showMessage

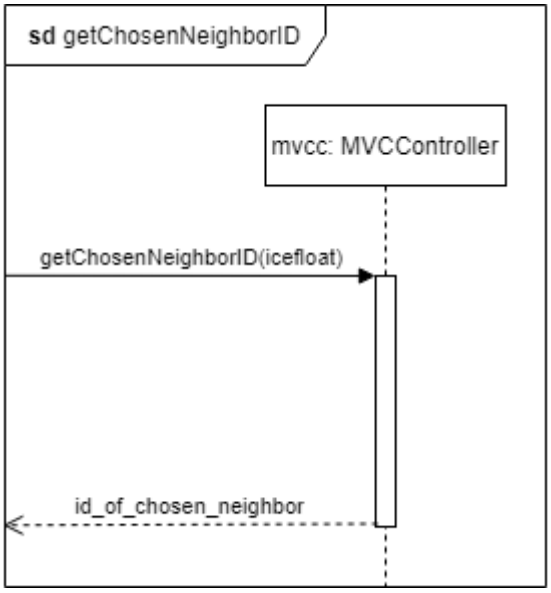


ext

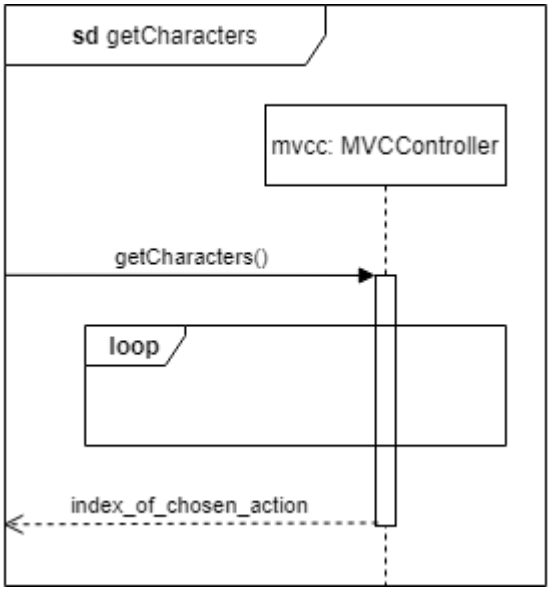
11.4.3.2 getAction



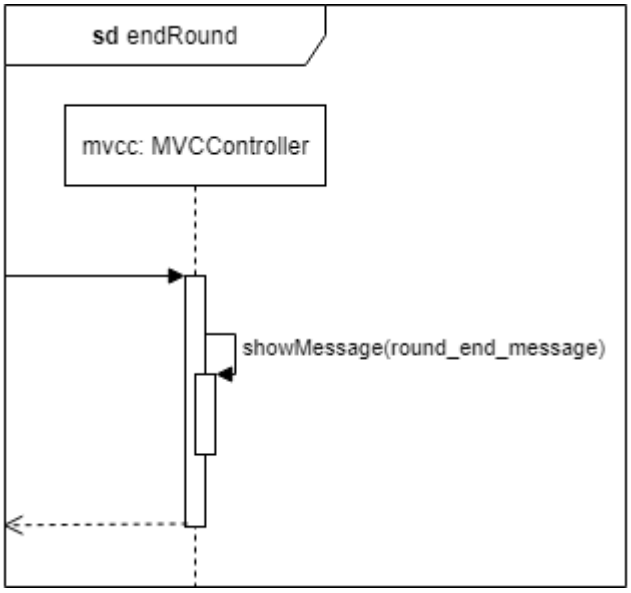
11.4.3.3 getChosenNeighborID



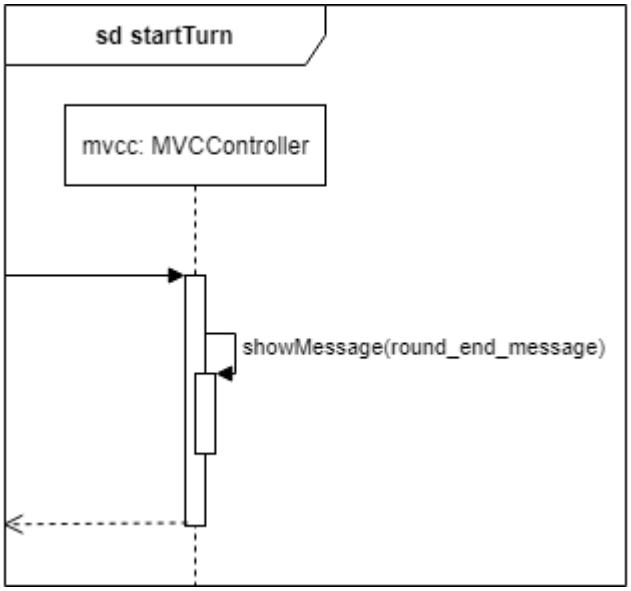
11.4.3.4 getCharacters



11.4.3.5 endRound

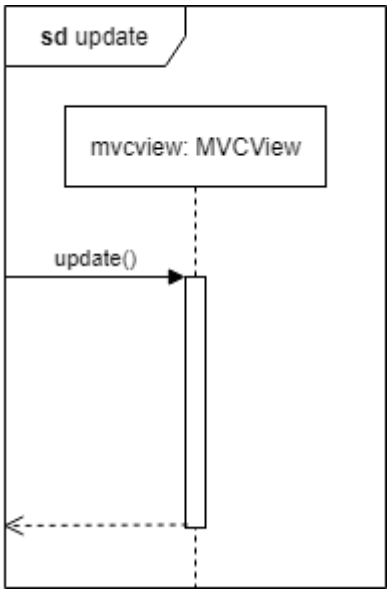


11.4.3.6 startTurn



11.4.4 MVCView

11.4.4.1 update



11.4.5 MessageView

11.4.5.1 setText