# Teilnehmer/innen des Teams:

|  |  |
| --- | --- |
| Klasse:  BI | Team:  Beispiel Dokumentation M226B LB2 |

# Anforderungsdefinition (Meilenstein A)

|  |  |
| --- | --- |
| „Liftsimulation“ | |
| **Auftrag:**  (Allgemeine Beschreibung) | Kundenauftrag: In unserem Parkhaus mit drei Liften (nebeneinander) soll eine Steuerung die Personen noch oben und unten transportieren.  Die Personen können auf jedem Stockwerk über die Anforderungstasten „Nach oben“ und/oder „Nach unten“ eine Kabine anfordern.  Die Kabinen werden durch die Anforderung auf dem Stockwerk angehalten und lassen die Personen durch die geöffnete Tür ein. Die Steuerung optimiert automatisch und wählt dazu eine der drei Kabinen aus.  ../../2%20-%20Unterlagen/Inputs/LiftSimulation.pngDie eingetretene(n) Person(en) wählen Zielstockwerke an und werden dorthin transportiert.  **Simulation:**   * Neben den drei Liften sollen auch die Personen simuliert werden * Per Mausklick sollen Personen auf dem entsprechenden Stockwerken erscheinen und beliebige Destinationen anwählen.   **Skizze 🡪**   * 6 Stockwerke * 3 Lifte |
| **MUSS**  **Kriterien:**  (Konkrete Features, die umzusetzen sind) | **Ablauf des Scripts:**   * Personen erscheinen auf Stockwerk (durch Mausklick) * Personen gehen in ankommenden Lift und wählen Zielstockwerk (Zufall) * Lift transportiert Personen zum Ziel * Personen steigen beim Zielstockwerk aus und verlassen den Flur. |

|  |  |
| --- | --- |
| **KANN**  **Kriterien:**  (Konkrete Features, die optional sind) | **Zusatz:**   * Lifte bewegen sich optimiert (Intelligente Liftsteuerung) * Anzeige der Anzahl Personen in Flur und Lifte * Animierte Personen * Anzahl Personen im Lift beschränkt |

## Planung LB2

|  |  |  |  |
| --- | --- | --- | --- |
| *MS* | *Tätigkeit / Abgabe* | *Soll-Datum* | *Ist-Datum* |
| A | ProjektstartTeam BildungWahl / Ausarbeitung der Anforderungsdefinition Abnahme Anforderungsdefinition durch Lehrperson |  |  |
| B | Teamaufgabe 1:Abgabe: Lösungsdesign  (Analyse, Design: Funktionsmodell, UseCase, GUI, Storyboard) |  |  |
|  |  |  |  |
| B2 | Teamaufgabe 2:Abgabe: Testvorschrift und Testfälle |  |  |
|  |  |  |  |
| C | Einzelaufgabe 3:Abgabe Szenario (.zip) mit Inline-Dokumentation, Systemdokumentation (UML Klassen-, Sequenzdiagramm)Fachgespräch Projektabnahme |  |  |
| C2 | Einzelaufgabe 4:Abgabe: Ausgefüllter Systemtest |  |  |

# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

## Funktionsmodell

Im Folgenden sind die erwarteten Funktionen schematisch dargestellt und beschrieben:



Legende:

* **Eingabe**: Benutzer erstellt Personen per Mausklick auf Stockwerk
* **Ausgabe**: Aktuelle Personenzahlen werden je Stockwerk und Lift angegeben

**Simulation / Funktionalität**:

* Personen fordern auf einem von sechs Stockwerken eine Liftkabine an, entweder nach oben oder nach unten
* Die Lifttüren können sich öffnen und schliessen
* Der ankommende Lift wird betreten und zufällig ein Zielstockwerk gewählt
* Liftkabine transportiert Person(en) zu ihrem Zielstockwerk
* Personen verlassen den Lift und den Flur auf ihrem Zielstockwerk
* Die Liftbewegung wird optimiert nach oben oder nach unten gesteuert

## Anwendungsfälle (UseCases)

Folgende Anwendungsfälle sind hier detailliert dokumentiert:



**Akteur User:**Der Benutzer (externer User an Greenfoot-Konsole) öffnte und startet ein Szenario. Ein gestartetes Szenario kann ge-reset-tet werden. Durch Klicken auf einen Flur wird eine Person mit zufälligem Zielstockwerk platziert, was eine realistische Simulation ermöglicht.

**Akteur GF\_Engine:**Die Engine animiert Personen, Lifte und Anforderungsknöpfe im Floor. (Die Liftkonsole wird nicht animiert.)  
Eine eingebaute Logik (LiftController) steuert die Lifte gemäss Anforderungen optimiert.

**Use-Case Enter / Leave Lift:**

Eine Person kann in den Lift einsteigen oder aussteigen, wenn der Lift auf dem richtigen Stockwerk öffnet.

**Use-Case …:**

…

## Ablauf (Storyboard)

Aus Benutzersicht ist folgender Ablauf des Programms zu erwarten:

|  |  |
| --- | --- |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-0 | Ausgangslage: Lifte warten auf Anforderungen |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-0 | Durch den Benutzer wird per Mausklick eine Person im entsprechenden Flur erzeugt und mit einem zufälligen Zielstockwerk „versehen“. |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-0 | Beim Ankommen des Lifts besteigt die erste Person den Lift und bestimmt damit (evtl.) die Fahrtrichtung. |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-06 | Mehrere Personen, die in die gleiche Richtung fahren wollen, können den Lift betreten. |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-0 | Der Lift transportiert die eingestiegene(n) Person(en) in die gewünschte Richtung und hält bei den angeforderten Stockwerken. |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-0 | Ein (hinauf- bzw.) hinunterfahrender Lift nimmt die Personen mit, die nach (oben) unten gehen wollen. |
| ../../../../../../Dropbox/Screenshots/Screenshot%202017-12-06 | Aussteigende Personen verlassen den Flur (und das Spiel). |

***Hinweis****: Hier sind üblicherweise Skizzen oder Mockups (www.balsamiq.com) eines GUIs platziert*

# Testvorschrift (LB2 Meilenstein B2: Teamaufgabe 2)

Testbeschrieb und vorbereitetes Testprotokoll siehe Dokument   
***M226B\_LB2\_Testvorschrift\_MS-B2.docx***

# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

Ausgefülltes Testprotokoll siehe Dokument   
***M226B\_LB2\_Testvorschrift\_MS-C2\_Name.docx***

# Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

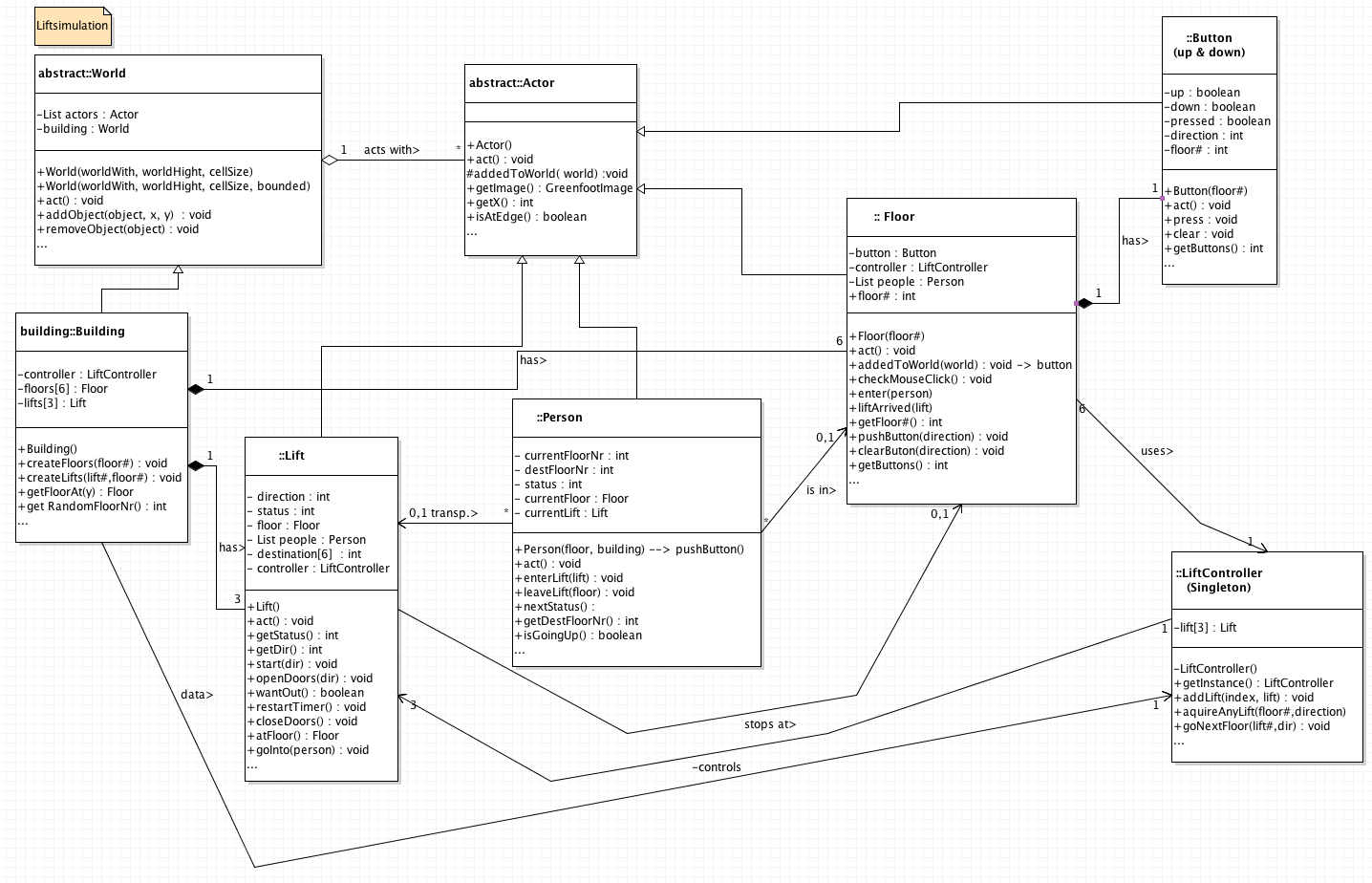
Das erstellte Java-Projekt (Greenfoot-Szenario) ist hier detailliert abgelegt:

*M226B\_ KEL\_Liftsimulation\_V0.91.zip*

*https://github.com/mkellenbergerTBZ/kelLifts*

## Statisches Design: Klassendiagramm

Folgend die statische Struktur des Szenarios „Liftsteuerung“

(UML Klassendiagramm mit Assoziationen und Kardinalitäten)  
Komposition die created werden: Building - Floor, Building – Lift  
Liftcontroller simuliert die Steuerungsinstanz

## Umfang / Abgrenzung / Änderungen gegenüber Design

Aufgrund unten beschriebener Umstände sind Anpassungen des ursprünglichen Lösungsdesigns gemacht worden:

Zusätzlich: Die Anzeige der Fahrtrichtung im Lift (Für Debugging)

## Funktionalität der Implementation.

Zusätzlich zu der Inline-Dokumentation sind hier folgende Funktionen detailliert beschrieben:

|  |  |
| --- | --- |
| Klasse.Methode | **Lift.Change( ...)** |
| Input | **direction** is either UP or DOWN  **onoff** is true for on, false for off |
| Verarbeitung | Sets one of the two buttons on or off |
| Output | acknowledgement **0/1** |

Siehe auch Inline-Kommentar und JavaDoc! (/\*\* @param @return \*\*/)

Codeausschnitt aus der Klasse Button mit JavaDoc:

/\*\*

\* Sets one of the two buttons on or off

\* **@param** direction is either UP or DOWN

\* **@param** onoff is true for on, false for off

\* **@return** acknowledgement 0/1

\*/

private int **change**(int direction, boolean onOff) {

if(direction == UP) {

up = onOff;

return 1;

}

else if(direction == DOWN) {

down = onOff;

return 0;

}

updateImage();

}

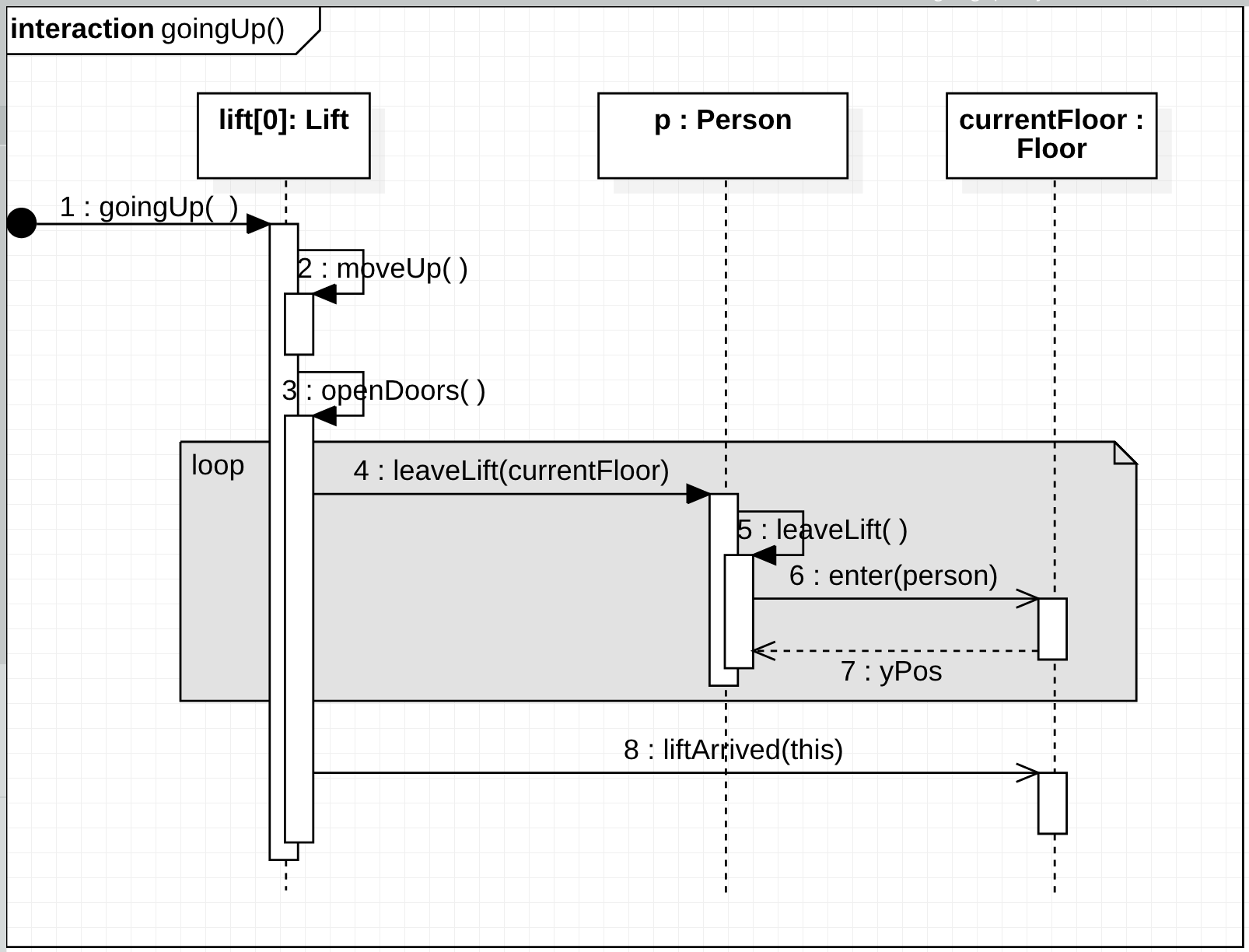
Weitere wichtige Funktionen:

...

## Dynamische Struktur: Sequenzdiagramm

Ein zentraler Ablauf eines UseCases ist im Folgenden dargestellt:

### UseCase „Move Up/Down“: Trace „Going Up“



**Trace: Stop at floor**

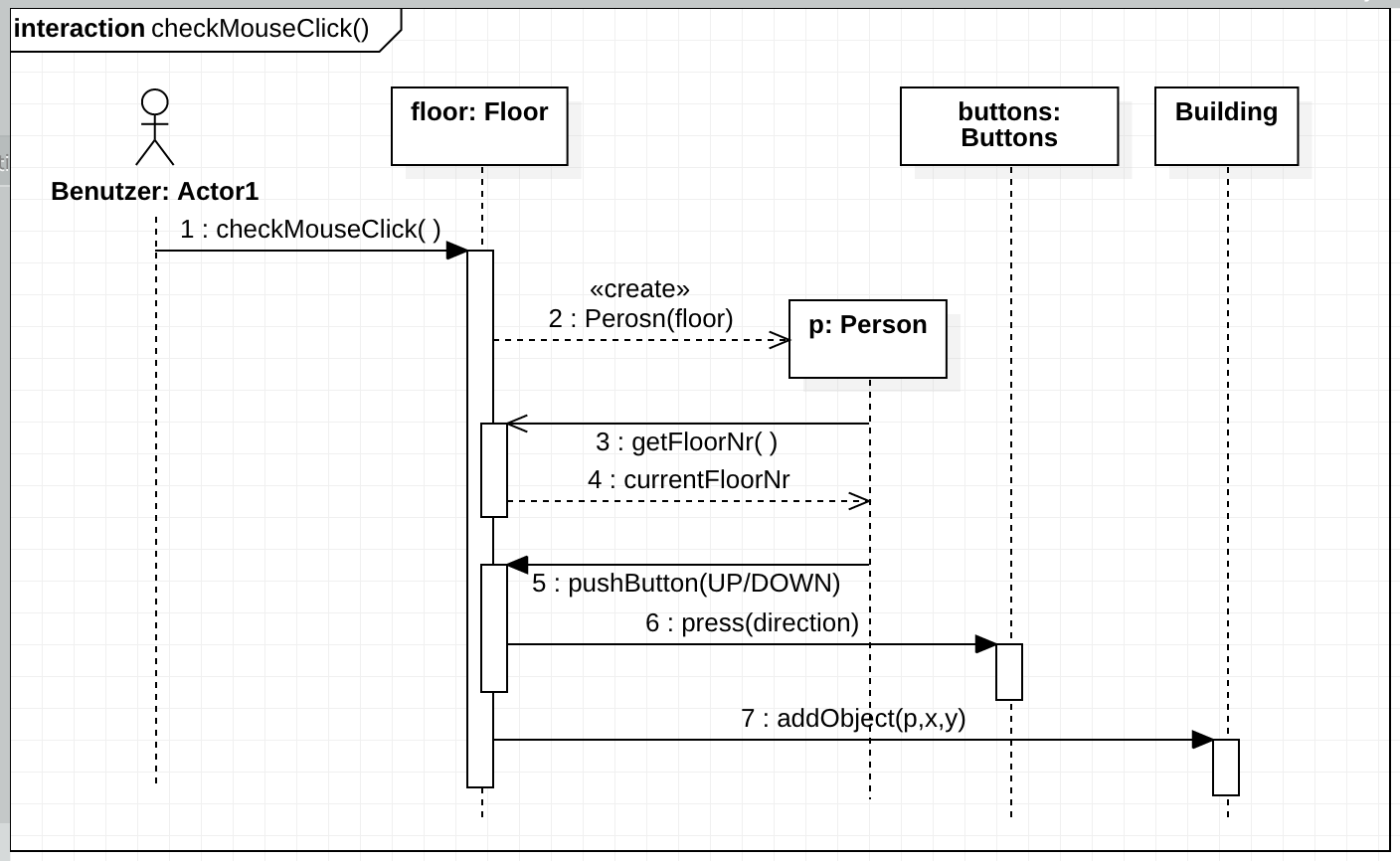
Ein hinauffahrender Lift stoppt auf einem angeforderten oder angewählten Stockwerk und öffnet die Türen.

Zuerst verlassen die an ihrem Ziel angekommenen Personen den Lift, indem sie den Flur betreten.

Dabei muss die Person in der Animation auf eine neue y-Position gesetzt werden.

Danach geht die Ausführung in der Methode liftArrived() weiter: *Personen, die in den Lift wollen treten ein...*

### UseCase „Create Person on Floor“:



**Trace Normal:**

Über die Methode CheckMouseClick() auf ein bestimmtes Stockwerk (Floor) wird eine Person instanziiert.

Die Person erhält dabei zufällig ein Zielstockwerk.

Der Konstruktor der neuen Person aktiviert einen Trigger im Button des aktuellen Stockwerkes, indem er die angeforderte Richtung des Stockwerk-Buttons drückt.

Der Stockwerkbutton wird animiert und aktiviert dann den Trigger (pressed = treu), der in der Act-Methode auslöst.

# Bedienungsanleitung (Meilenstein C: individuelle Aufgabe 3)

1. Installieren Sie Java falls nicht vorhanden (Google: «Java download»)
2. Starten Sie Greenfoot (Download @ greenfoot.org)
3. Laden Sie das entzippte Szenario Liftsimulation aus Kap. 6 (🡪GitHub clone)
4. Starten Sie die Simulation mit RUN.
5. Platzieren Sie neue Personen in den Stockwerken, indem sie ein Stockwerk anklicken.
6. Beobachten Sie die Liftsimulation ...