# Teilnehmer/innen des Teams:

|  |  |
| --- | --- |
| Klasse: | Team:  Beispiel Anforderungsliste |

# 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 der Simulation:**   * 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. |
| **SOLL**  **Kriterien:**  **(Konkrete Features, die optional sind)** | **Folgende Features sollen bei Bedarf implementiert werden: (Future Release)**   * **Die Liftkabinen zeigen Anzahl Personen im Lift an.** * **Die Liftkabinen zeigen die Richtung ihrer Bewegung an.** |
| **KANN**  **Kriterien:**  (Konkrete Features, die optional sind) | **Folgende Features können zusätzlich implementiert werden: (Varianten, Kreativität)**   * Lifte bewegen sich optimiert (Intelligente Liftsteuerung) * Anzeige der Anzahl Personen in Flur * Animierte Personen * Anzahl Personen im Lift beschränkt |
| **Nicht-funktionale**  **Kriterien:**  **(Qualität, Geschwindigkeit, Benutzbarkeit)** | **Folgende nicht-funktionale Kriterien sollen umgesetzt werden:**   * Reaktionzeit Personenplatzierug: < 0.2s * Bewegung Liftsimulation veränderbar |

# Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

## Funktionsmodell

Identifizierung der Objekte und Konzepte (Inter-Aktionen):  
**Objekte**:  
Knopf oben unten, Stockwerke, Parkhaus, Liftschächte, Lifte, Türen, Personen, Anzeige

**Konzepte**:  
Transport, Richtungslogik, Liftanforderung, Fahroptimierung, Bedienung Türe  
  
Im Folgenden sind die erwarteten Eingaben und Ausgaben beschrieben / dargestellt:



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:



Legende:

* Der User bedient die Simulation, indem er das Scenario steuert und neue Personen auf die Stockwerke platziert (spawn).
* Buttons werden animiert und visualisiert
* Personen werden animiert und visualisiert
* Liftkabinen werden animiert und visualisiert
* Liftkontroller übernimmt Steuerung der Kabinen
* **Akteur User:**Der Benutzer (externer User an Greenfoot-Konsole) öffnet 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.

## Ablauf

Aus Benutzersicht ist folgender Ablauf des Programms zu erwarten:

(Storyboard)

|  |  |
| --- | --- |
|  | **Startsituation:** Szenario muss gestartet werden (RUN >) 3 Kabinen warten auf Kundschaft 6 Stockwerke sind leer |
| Ein Bild, das Text enthält.  Automatisch generierte Beschreibung | **Useraktivität**: Mit Mouse-Klick werden Personen gespawnt  **Personenaktivitäten (simuliert):** Personen drücken Anforderungsbuttons Personen besteigen Kabinen und wählen zufällig ein Zielstockwerk Personen werden in Kabine gezählt und transportiert |
| Ein Bild, das Text enthält.  Automatisch generierte Beschreibung | **Liftaktivitäten (simuliert):** Liftkabinen haben offene und geschlossene Türen Liftkabinen zeigen Anzahl Personen und Bewegungsrichtung an  **Personenaktivität (simuliert):** Personen verlassen Kabine auf dem Zielstockwerk |
| Ein Bild, das Text enthält.  Automatisch generierte Beschreibung | **Steuerung der Liftaktivitäten (simuliert)**: Kabinen werden gemäss Zielwahl und Anforderungen optimiert gesteuert |

***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\_Systemtest Bsp Liftsimulation.docx***

# Testprotokoll (LB2 Meilenstein C2: individuelle Aufgabe 4)

Ausgefülltes Testprotokoll siehe Dokument   
***M226B\_Systemtest Bsp Liftsimulation.docx***

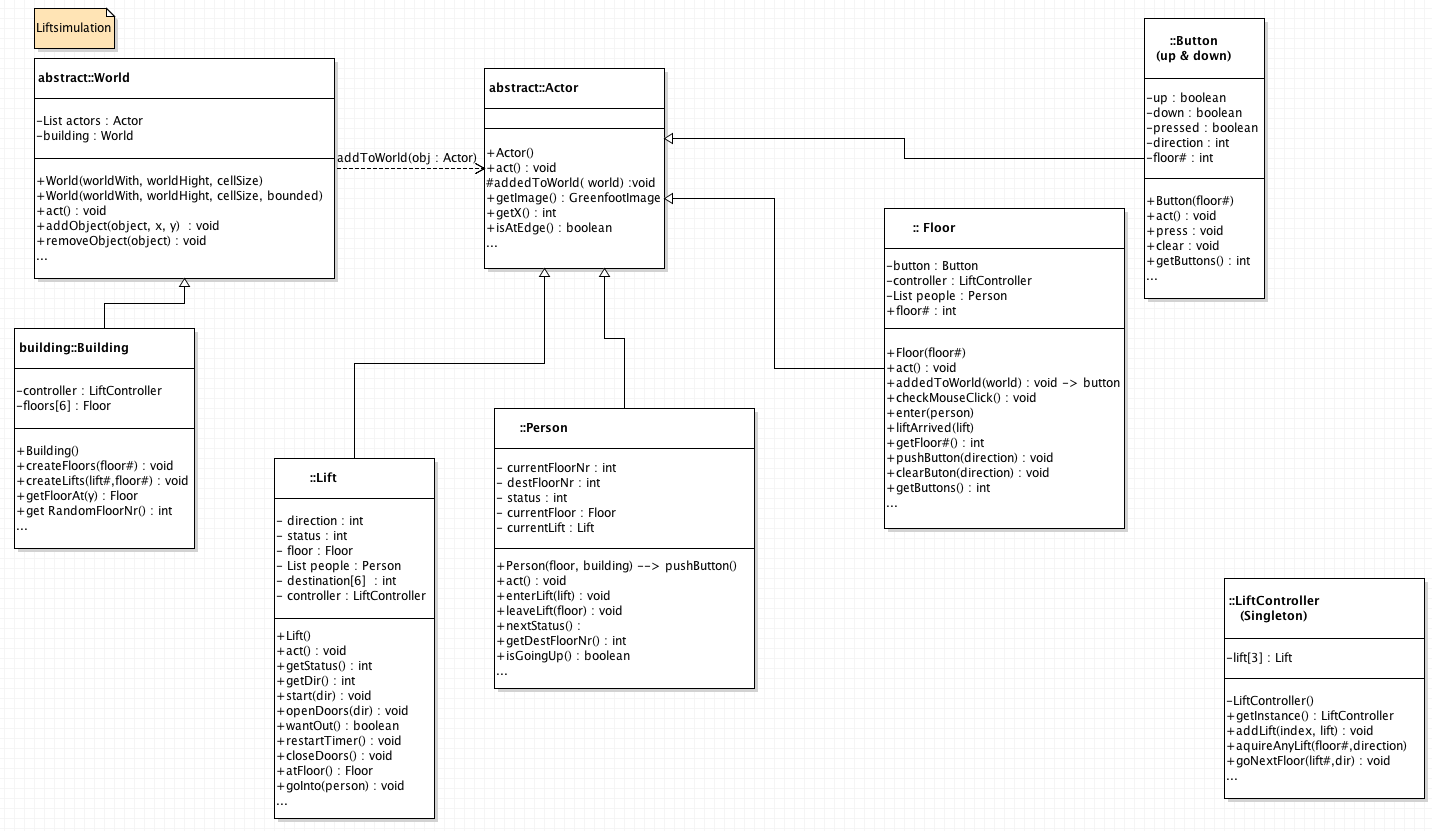
# Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

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

***kelLifts\_V0\_x.zip***

## Statisches Design: Klassendiagramm

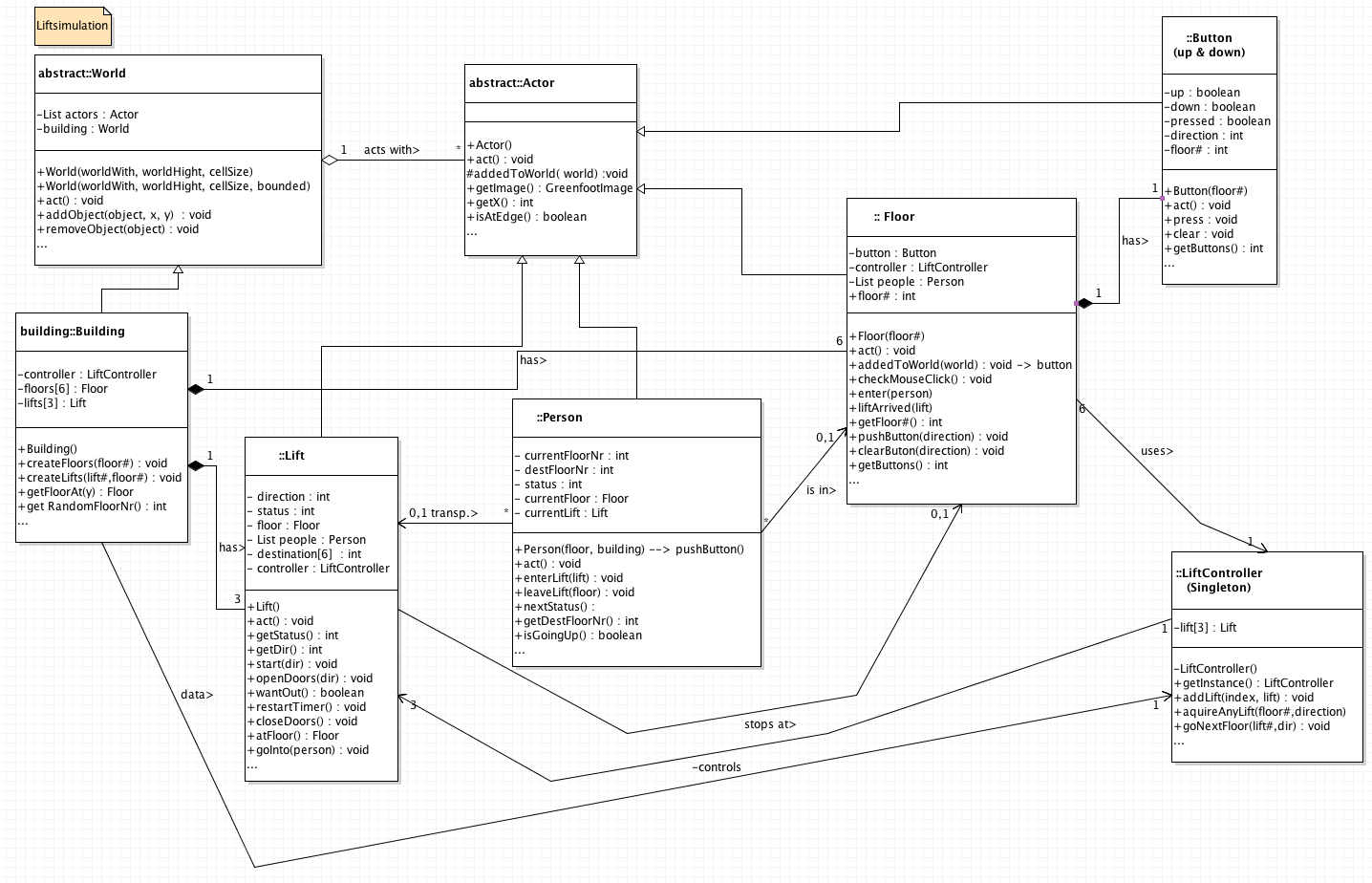
Folgend die statische Struktur des Szenarios: **Erster, grober Entwurf**

****

Legende:

* Building (Welt): Gebäude mit 6 Stockwerken und drei Liftschächte
* Liftklasse für die drei bewegten Lifte
* Personen, die auf Stockwerk gespant werden, in die bzw. aus der Liftkabine laufen und transportiert werden.
* Stockwerk mit den Anforderungsbuttons
* Die Logik ist im Kontroller platziert

Folgend die statische Struktur des Szenarios: **Finale Version**



Legende:

* Building (Welt): Gebäude mit 6 Stockwerken und drei Liftschächte
* Liftklasse für die drei bewegten Lifte
* Personen, die auf Stockwerk gespant werden, in die bzw. aus der Liftkabine laufen und transportiert werden.
* Stockwerk mit den Anforderungsbuttons
* Die Logik ist im Kontroller platziert

## Umfang / Abgrenzung / Änderungen gegenüber Design

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

* Anzeige der Anzahl wartenden Personen auf Stockwerk wurde hinzugefügt
* Es wurden keine weiteren Klassen hinzugefügt
* Bei diverse Methoden wurden Signaturen angepasst (Refactoring)
* Diverse neue Methoden wurden hinzugefügt.

**Siehe Versionshistory:**

\* Person will be set on floor (via constructir), moves to lift or leaves floor.

\*

\* @author KEL \* @version <li> V0.1 (Beta)

\* @version <li> V0.2 17.12.2017 Status LEAVE\_FLOOR added

\* @version <li> V0.3 18.12.2017 Error status-call solved

\* @version <li> V0.4 18.12.2017 old Version

\* @version <li> V0.5 21.08.2017 Refactoring

\* @version <li> V0.6 06.12.2018 Some enhancements / comments added

\* @version <li> V0.7 08.12.2020 Fix error: Numbers on floor on exit! >> currentFloorNr deleted!

\* Lift simulation: Floor

\*

\* @author KEL

\* @version <li> V0.4 18.12.2017 old Version

\* @version <li> V0.5 21.08.2017 Refactoring

\* @version <li> V0.6 30.08.2017 Liftcontrolling

\* @version <li> V0.7 12.12.2020 fixed: Persons on edge floor are left out

\* Lift is moving without buttons hit

\* @version <li> V0.8 12.12.2020 fixed: While going up an empty - not stopping for opposite direction

## Funktionalität der Implementation.

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

### Methode goingUp()

|  |  |
| --- | --- |
| **Klasse Lift** | **goingUp()** |
| Funktion | \* Act on status:   We are currently going up - perform the next step.  \* Checks if cabine is at a new floor level:  \* Stopps and checks if somebody awaits lift --> open doors |
| Input | None |
| Output | None  (Cabine is moved one pixel upwards) |

### Methode enter().

|  |  |
| --- | --- |
| **Klasse Floor** | **int enter(Person p)** |
| Funktion | \* Person p enters this floor's from lift. |
| Input | P: Person who enters the floor leaving lift |
| Output | int Value: new floor-position to draw person on floor |

### Methode change().

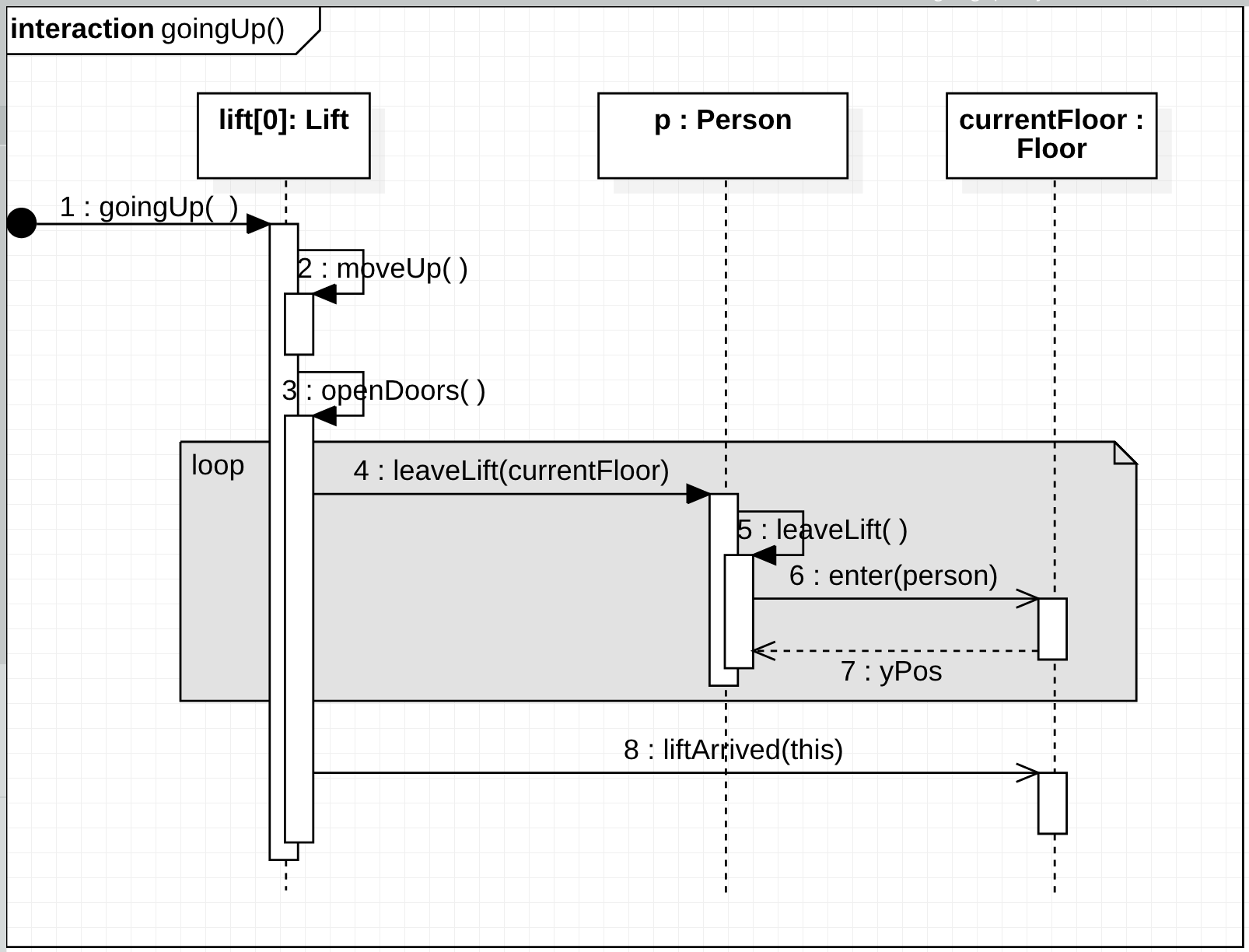
|  |  |
| --- | --- |
| **Klasse Lift** | **change( ...)** |
| Funktion | Sets one of the two buttons on or off |
| Input | **direction** is either UP or DOWN  **onoff** is true for on, false for off |
| Output | acknoledgement **0/1** |

…

## Ein Bild, das Text enthält. Automatisch generierte BeschreibungDynamische Struktur: Sequenzdiagramm

Ein zentraler Ablauf eines UseCases ist im Folgenden dargestellt

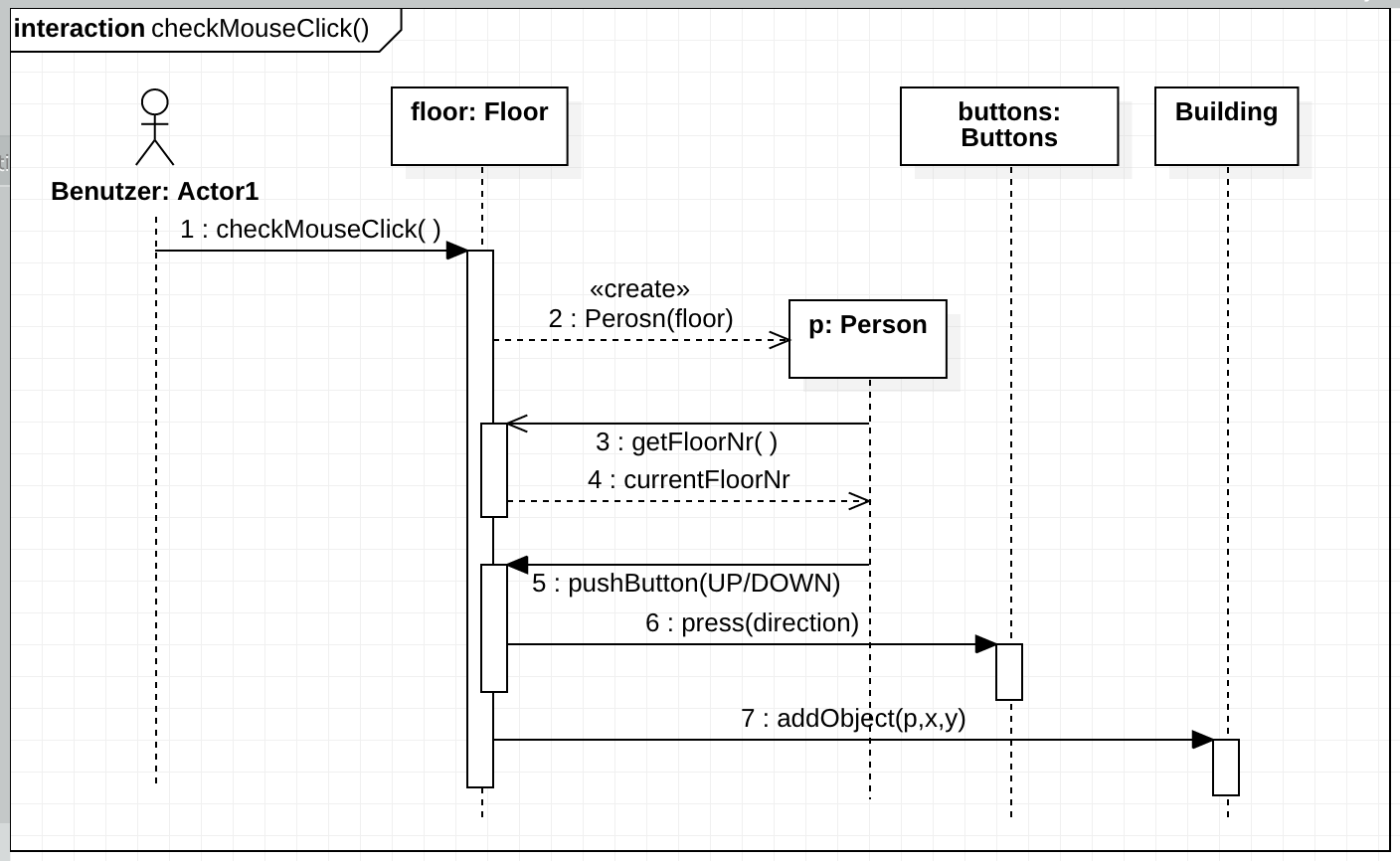
### UseCase „Move Up/Down“: Trace „Going Up“



**Trace:**

1. Methode wird aufgerufen, um Lift nach oben zu bewegen
2. Eigentliche Bewegung um ein Pixel
3. Falls auf neuem Stockwerk angekommen: Türen werden geöffnet
4. Für alle Personen p die raus wollen:
5. p verlässt Kabine
6. und betritt aktuelles Stockwerk
7. welches auf der Position Y ist (um Person am richtigen Ort zu zeichnen)
8. Liftkabine nun angekommen (und geleert) und neue Personen können rein
9. ...

### 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

|  |  |
| --- | --- |
|  | Szenario **kelLifts** mit mind. Greenfoot V3.7 laden  [RUN]-Button klicken für Start [RESET]-Button klicken für Stopp  Mit Slider kann Geschwindigkeit eingestellt werden |
| Ein Bild, das Text enthält.  Automatisch generierte Beschreibung | **Useraktivität**: Mit Mouse-Klick auf bestimmtes Stockwerk werden Personen gespawnt.  **Simulation:** Beobachten der Liftsteuerung |