Lab Report 03

Assignment 03 - Sequence Diagrams and State Machine Diagrams

Authors: Dennis Loska, Tony Dorfmeister, Ai Dong 27.11.2017

Part 1: Sequence Diagrams

Assignment

As a finger exercise for Sequence Diagrams, pick one of the following example projects from the first semester and draw a sequence diagram for the main use case:

- The Clock Display / Use case: timeTick() is called
- Auction / Use case: makeABid() is called
- Tech Support / Use case: user command is entered (start() method in SupportSystem)
- The Zuul Project / Use case: user enters command (method: play() in Game.java)

Von den obern gennanten Use Cases haben wir uns für **timeTick()** entschieden:

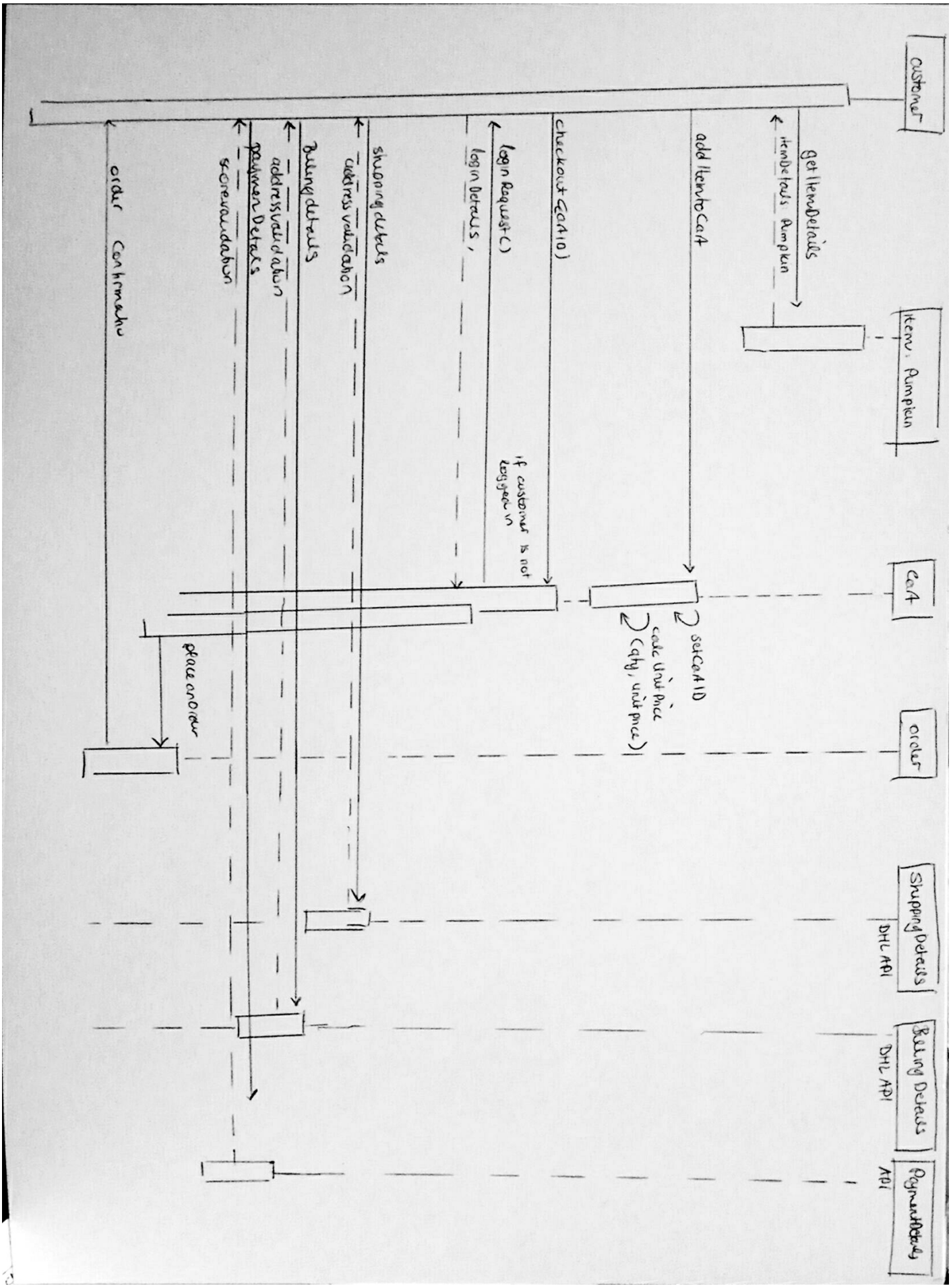
timeick() A look upon the repository reveals that clock is the main iniator as it also initiate an instance of Timethread which is derived from Thread. timeTick() itself is nested within different methods and various methods from timeTick are instance methods of several different classes maing it challenging to define the process. Additionally, the most challenging struggle involved the if-statements as there are several ways on how to implement it. Open questions are:

1. How do we deal with if/else structurally?
2. How to handle the nesting of several instance methods within methods?

Now take your scenarios from the second exercise and have a good look at them. There are a number of processes that you should have detailed in your scenarios - if not, now you learn how to be thorough :) You need to draw sequence diagrams for the following use cases:

- Ordering a Pumpkin/Treat
- one other use case of your choosing

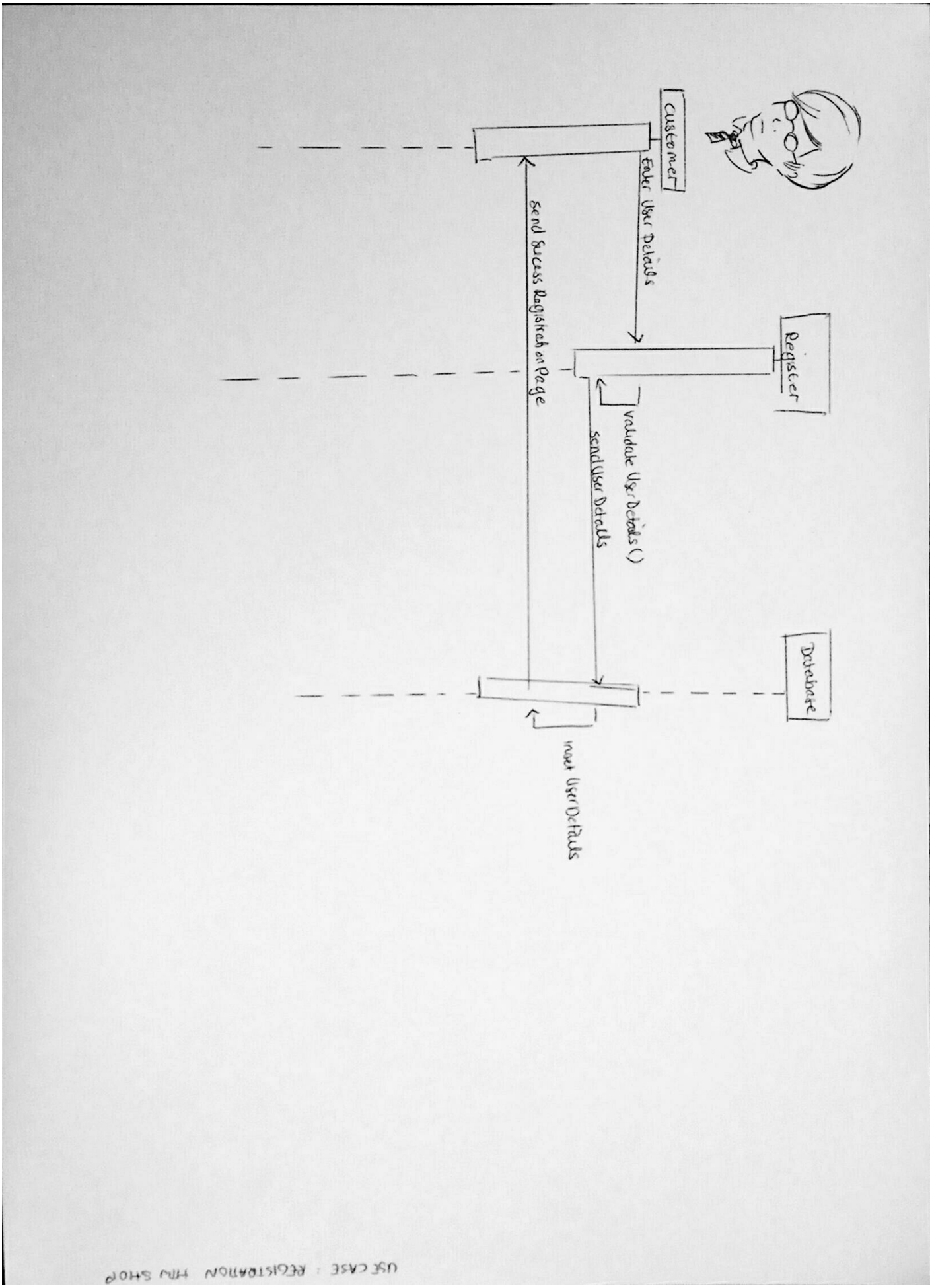
Ordering a Pumpkin/Treat



orderPumpkin() Based on the last submission, the ordering process revolves around the steps login/registration, billing and shipping input as well as order confirmation and the constant validation from several systems including API calls. Challenges: Depending on eCommerce systems ths process has several different implementations fulfilling the class diagrams.

Registration at the HTW Shop

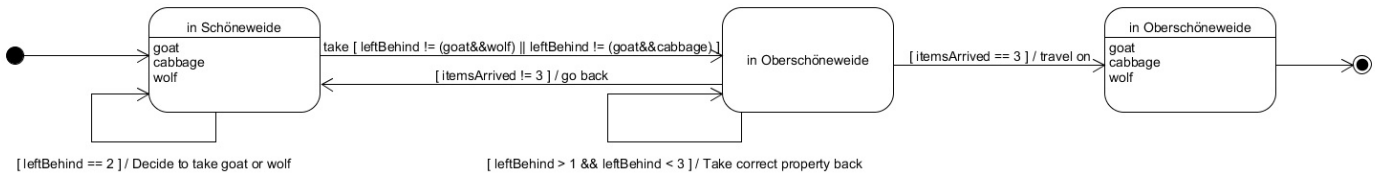
Neben dem Use Case "Ordering a Pumpkin/Treat" haben wir uns außerdem für ein Use Case entschieden, welches die Registrierung eines Users im HTW-Shop beschreibt.



registerShop() A part of the ordering process involves the registration process which is als crucial in eCommerce. Here, the validation is crucial for the database handling/ design of the implementation of the persistance framework.

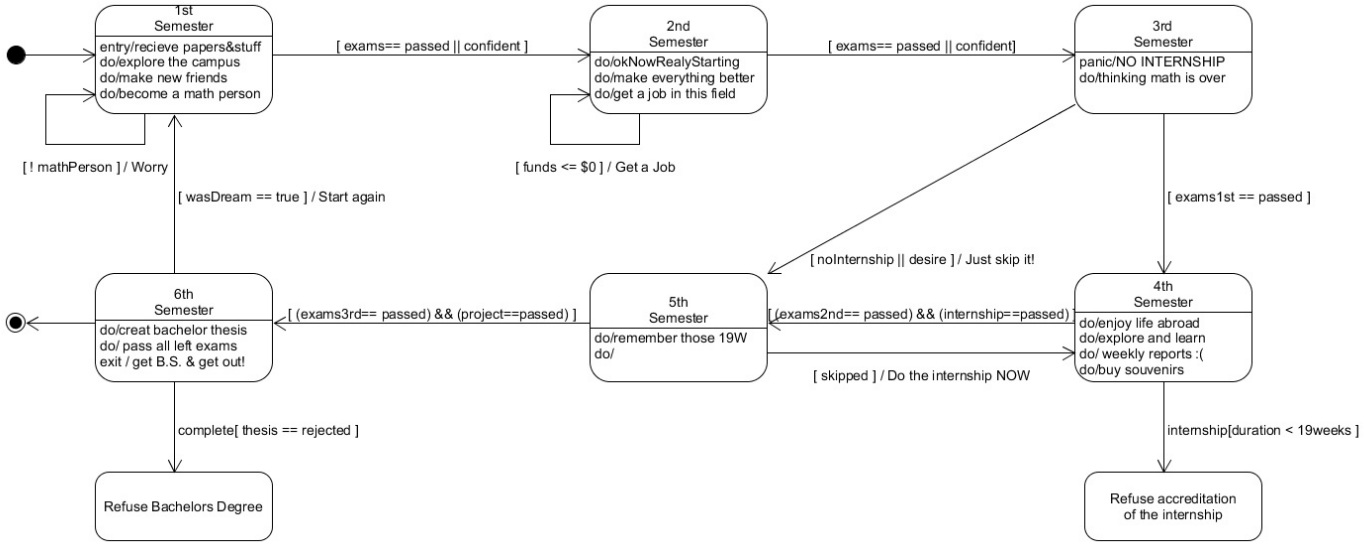
Part 2: State Machine Diagrams

Model the modified goat/cabbage/wolf problem: The farmer is in Schöneweide and wants to get his goat, his cabbage and his wolf over to Oberschöneweide. Only one thing can fit in his boat at a time beside himself. He cannot leave the cabbage and the goat or the goat and the wolf alone on the same side of the river, for obvious reasons. Is it possible for him to get all three possessions across the Spree? Draw a State Machine Diagram modelling both solutions to this problem.



Für das vorliegende Problem wurden 2 States moduliert. Jedes State repräsentiert also eine Uferseite des Flusses, worüber der Farmer mit seinem Gut vollständig passieren möchte. Das State Machine Diagram zeigt hierbei die Logik auf, in welcher Reihenfolge dies zu tun ist. In jedem Fall muss zuerst die Ziege mitgenommen werden. Als zweites kann man sich zwischen dem Wolf und dem Gemüse entscheiden und muss das entsprechende "Gegenstück" wieder auf die andere Seite des Ufers mitnehmen, sodass nie Ziege und Gemüse oder Ziege und Wolf miteinander auf einem Ufer ohne Aufsicht sind. Sobald alle 3 Sachen in Oberschöneweide sind, kann der Farmer weiterziehen.

Model the states an IMI student passes through from the first until the sixth semester. (Special prize for the most humorous model that is not offensive.)



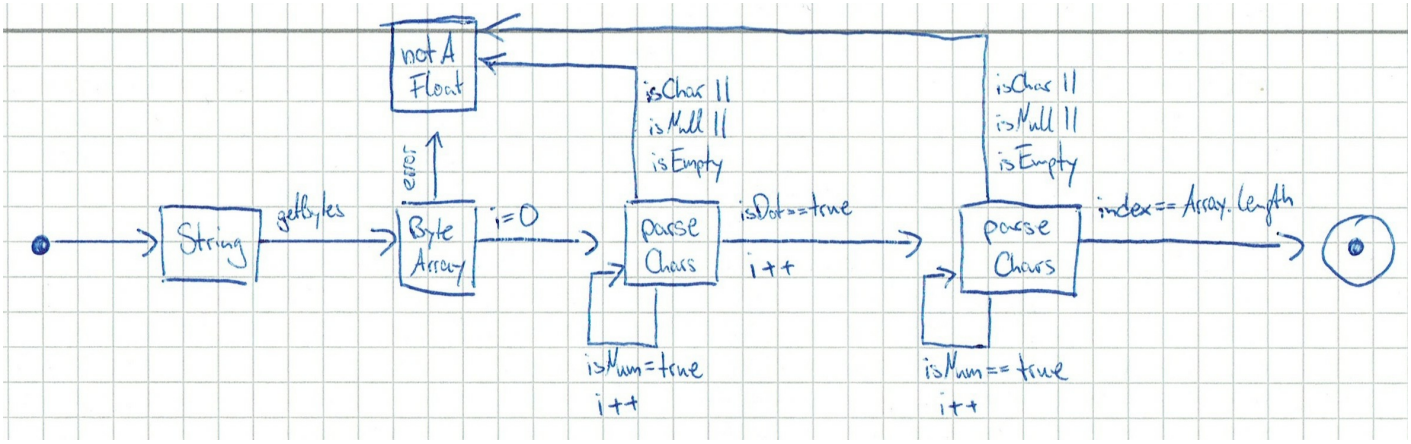
Zur Beschreibung des Studiengangs wurde für jedes Semester ein Zustand/State erstellt, in welchem für das jeweilige Semester spezifische Methoden/Aufgaben anstehen, um in das nächste Semester zu ziehen. Im Falle des 3. Semesters kann man z.B. in das 5. direkt gehen, wenn man noch kein Praktikum für das 4. Semester gefunden hat. Ansonsten sind noch das Bestehen der Klausuren mit Einbeziehung der 3-Semester-Regel eine Vorraussetzung in einigen Fällen, um vom einen Semester ins nächste zu kommen.

Model the states of a parser that determines if a given string is a proper floating-point number.

Der Parser besitzt die folgenden Zustände:

- String
- ByteArray
- parseChars
- notAFloat

Zuerst wird ein String in ein ByteArray konvertiert. Dieses ByteArray wird durchlaufen. Währenddessen wird überprüft ob der derzeitige Character auf seinen Typ überprüft. Ist der Character entweder ein Buchstabe, Sonderzeichen, NULL oder leer handelt es sich um keinen Floating-Point-Value. Handelt es um eine Ziffer oder einen Punkt ruft sich der Zustand selbst auf und der Index wird erhöht bis eine andere Bedingung erfüllt ist. Ist der Index gleich der Länge des Arrays haben wir das gesamte Array durchlaufen und es handelt sich um einen Floating-Point-Value.



Model the states a Pumpkin order in the HTW system can be in.

Das HTW System besitz die folgenden Zustände:

- Item in Cart
- pending
- cancelled
- Shipping

Zunächst muss sich der zu erwerbende Gegenstand in der Cart befinden. Von hieraus kann der User den Gegenstand bezahlen womit die Bestellung den Zustand *pending* erhält. Ist die Bezahlung nicht erfolgreich wandert der Gegenstand zurück in die Cart. Ist der Gegenstand zur Zeit nicht auf Lager bleibt die Bestellung solange im Zustand *pending* bis der Gegenstand wieder vorrätig ist. Möchte der User die Bestellung stornieren erhält die Bestellung den Zustand *cancelled* und das System ist beendet. Sind beide Bedingungen erfolgreich erhält die Bestellung den Zustand *shipping*. Konnte der Gegenstand nicht zugestellt werden erhält die Bestellung erneut den Zustand *pending*. Auch in diesem Zustand hat der User noch die Möglichkeit die Bestellung zu stornieren. War die Lieferung jedoch erfolgreich, hat das System seinen finalen Zustand erreicht.