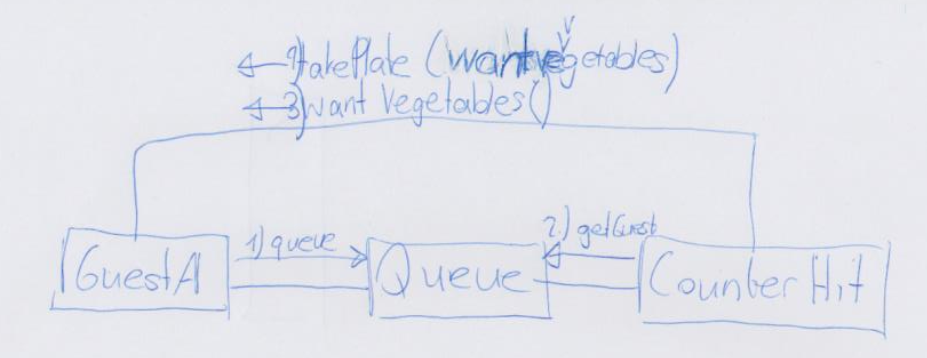
Übung 6

# Aufgabe 1: Tageshit ohne Gemüse

## Erweitertes Kommunikationsdiagram

Ablauf:

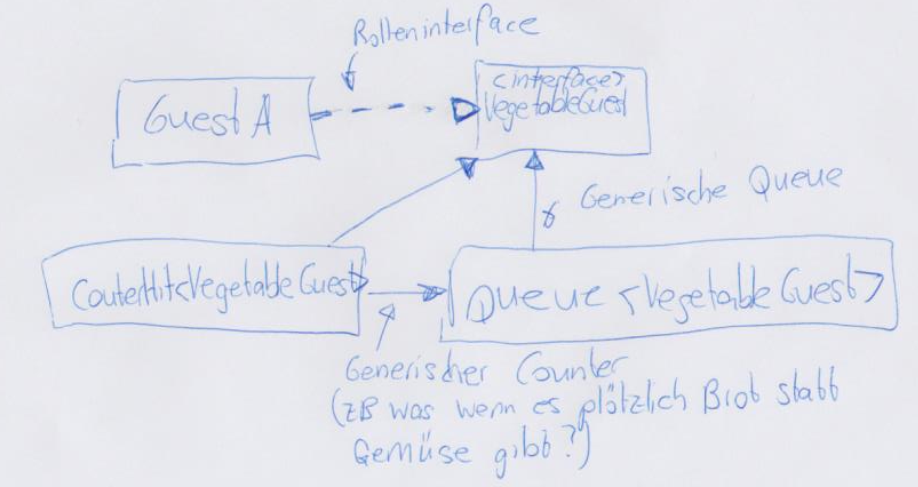
1. Gast A stellt sich an der Schlange an
2. Gast A ist an der Reihe und wird durch die Theke bedient
3. Die Theke fragt Gast A, ob er sein Menü mit oder ohne Gemüse möchte
4. Die Theke gibt dem Gast A sein Tablett zurück und vergisst das Gemüse in 5% aller Fälle, sofern er das Gemüse gemocht hätte



## Rolleninterface für den Gast mit Gemüse

|  |
| --- |
| public interface VegetableGuest {  boolean wantVegetables();    boolean takePlate(boolean wantvegetables);  }  public GuestA implements VegetableGuest {  // Implement the interface methods  } |

## Interface Segregation anhand der Restaurant Simulation

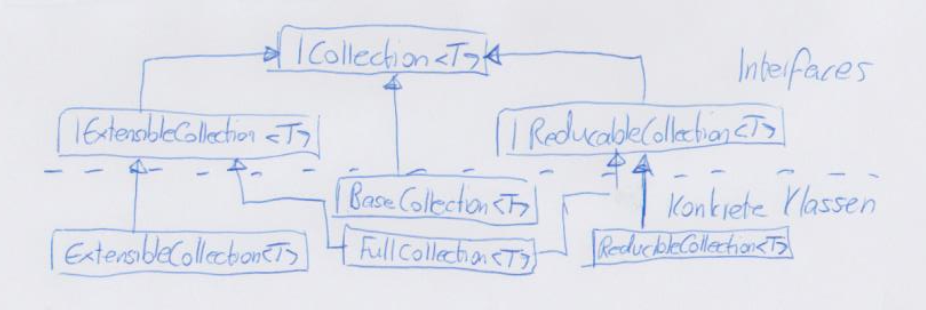


## Implementierung der erweiterten Simulation

Aus Zeitgründen musste ich diese Teilaufgabe weglassen.

# Aufgabe 2: Java Collection Framework

## Entwurf eines einfachen Collection Frameworks



## Gründe gegen die Verwendung dieses Ansatzes

1. Java und die Standard Library möchten immer abwärtskompatibel bleiben. Würde man das Prinzip der Interface Segregation anwenden könnte es sein, dass durch den Wegfall von Methoden welche eine UnsupportedMethodException werfen würden zu API Breaks kommt
2. Das Interface Segregation Prinzip würde das bereits das für viele zu komplexe Collection Framework noch weiter aufblasen (Stichwort «Perfektes» Design versus Komplexität
3. Nach Konsultation der Lösungen: Interfaces werden nicht nach oben weitergegeben

# Aufgabe 3: Exceptions und das Liskov Prinzip

Generell lässt sich sagen: Wirft eine Methode in einer implementierenden Klasse eine andere Exception als in der Signatur der Methode, so muss diese Exception auf die Exception in der Signatur ableitbar sein. Beispiel für eine Methode die die Exception Base wirft:

Foo extends Bar extends Base = Foo, Bar, und Base dürfen geworfen werden, da Signatur/ableitbar

Foo extends Base extends Bar = Foo und Base dürfen geworfen werden, da Signatur/ableitbar

|  |
| --- |
| // Funktioniert, da beide Exception der ursprünglichen Signatur entsprechen  class NiceGame implements FootballGame {  public void atBall(Player p) throws OffsideException, YellowException {  }  }  // Funktioniert, da die Exception RedException von einer Exception in der Interface Signatur abgeleitet ist  class HardGame implements FootballGame {  public void atBall(Player p) throws RedException {  }  }  // Funktioniert nicht, da die Exception StormException nicht von einer Exception in der Interface Signatur abgeleitet ist  class StormyGame implements FootballGame {  public void atBall(Player p) throws OffsideException, YellowException, StormException {  }  }  // Funktioniert nicht, da die Exception FootballException nicht von einer Exception in der Interface Signatur abgeleitet ist  class NormalGame implements FootballGame {  public void atBall(Player p) throws FootballException {  }  } |