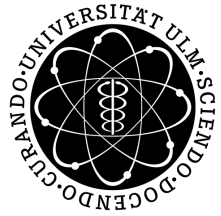

Universität Ulm

Freiwilliges EidI Projekt

Placeholder

Tut Mi 16 2202

supervised by
Raphael Störk

16. Dezember 2019

Zusammenfassung

In diesem Dokument sind alle wichtigen Prozessschritte, Entscheidungen und Ergebnisse dokumentiert, welche im Verlauf des Projektes 'Placeholder' entstanden sind. Das Projekt selber ist eine eigenständige, frewillige Zusatzleistung der Tutoriumsteilnehmer, welches als Unterstützung der Studenten beim Erlernen wichtiger Grundlagen helfen soll. Das Projekt dient insbesondere NICHT als Pflichtteil der Übung sondern ist ein individuelles Zusatzangebot des Tutors.

[Ergebnisse]

Inhaltsverzeichnis

1 Einführung **3**

2 Grundlagen **4**
2.1 Vorgaben 4
2.2 Projekt-Parameter 5
2.2.1 Technische Vorgaben 5
2.2.2 Genre 5
2.2.3 Konzept 5
2.2.4 Story 5

3 Anforderungen **6**
3.1 Funktionale Anforderungen 6
3.2 Qualitäts Anforderungen 6

4 Entwicklung **7**
4.1 Woche 1 - Grundlagen 7
4.2 Woche 2 - Erste Abfragen 9
4.3 Woche 3 - Rechnungen/Mini-Game 10
4.4 Woche 4 - Strings und Schleifen 12
4.5 Woche 5 - Arrays 13
4.6 Woche 6 - Methoden 15
4.7 Woche 7 - Objektorientierung 19

5 Implementierung **20**
5.1 Framework 20
5.2 Architektur 20
5.3 Vorstellung der Software 20

6 Einordnung **21**
6.1 Verwandte Arbeiten 21
6.2 Evaluation 21
6.3 Ausblick 21

1 Einführung

In Verbindung mit den aktuellen Tutorien in Einführung in die Informatik an der Universität Ulm möchten die Teilnehmer des Tutoriums von Raphael Störk, Mittwoch 16-18, neben den Pflichtübungen einen zusätzlichen Arbeitsaufwand leisten um eine breitere Grundlage und ein größeres Basiswissen für ihr weiteres Studium zu schaffen. Dafür wurde von dem Tutor ein gemeinsames Projekt vorgeschlagen, welches in den Tutorien als Zusatzübung erarbeitet werden soll. Ziel dieses Projektes ist nicht in erster Linie die Produktion einer vollständig entwickelten und getesteten Software sondern die Erlernung Grundlegender Programmierkenntnisse anhand praktischer Beispiele, welche im Verlauf des Projekts erarbeitet werden.

Der grundlegende Gedanke hinter diesem Projekt ist die Anwendung der im Tutorium kennen gelernten und vorgestellten Themen auf ein großes, kontinuierlich erweitertes Projekt um das Verständnis der Studenten in der Hinsicht zu fordern, dass klar wird wofür diese Themen im Bereich der Informatik und Programmierung wichtig sind und verwendet werden.

Klar muss auch sein, dass dieses Projekt eine rein freiwillige Übung für die Studenten ist. Die Teilnahme und Einbringung an dieser Übung wird nicht in die Bepunktung der Übungsvorleistung mit eingerechnet und ist auch niemals als eine Voraussetzung für das Bestehen einer Universitären Leistung gedacht. Auch ist dem Tutor keine weitere Tutoriumsgruppe bekannt, die solch ein Projekt durchzuführen gedenkt. Daher sollte eindeutig gesagt werden, dass dieses Projekt als reine Zusatzübung entwickelt wird. Entsprechend sollte die Richtigkeit und Vollständigkeit mit Bedacht genossen werden.

2 Grundlagen

2.1 Vorgaben

Folgende Vorgaben wurden zu Beginn des Semesters an das Tutorium angebracht:

Motivation Das Projekt wird nur verfolgt, wenn die Teilnehmer des Tutoriums dies auch möchten. Hierfür wurde eine entsprechende Umfrage erstellt bei der heraus kam, dass die Mehrzahl der Teilnehmer dieses Projekt angehen möchten.

Intention Das Projekt soll gemeinsam in der Zeit des Tutoriums, soweit es die pflichtigen Aufgaben zulassen, entwickelt werden. Die Intention hinter diesem Projekt ist die zusätzliche Bereitstellung von theoretischem und praktischem Wissen im Bereich der Grundlagen der Informatik und Programmierung.

Themenfreiheit Die Teilnehmer haben demokratische Stimmgewalt über die Themen und Funktionen des Projektes. Der Tutor behält sich jedoch ein allgemeines Veto-Recht vor, welches in Stichsituationen zum Einsatz kommen kann.

Erreichbarkeit Alle im Tutorium entwickelten Ressourcen werden den Teilnehmern online jederzeit zugänglich gemacht. Damit soll erreicht werden, dass die Teilnehmer jedwedem Thematischen Vorwissen die Möglichkeit haben alle erarbeiteten Projektteile zu verstehen und zu diskutieren. Zudem dienen diese Ressourcen als Hilfe für weitere Aufgaben und als zusätzliche Vorbereitung auf die Klausur. Hierbei ist zu beachten, dass keine Musterlösungen oder Plagiate den Weg in dieses Projekt finden dürfen.

2.2 Projekt-Parameter

In diesem Abschnitt wird im Verlauf der Projektentwicklung aufgeführt, welche Parameter durch die Teilnehmer erdacht und gewählt wurden. Der bisherige Stand findet sich in folgender Auflistung:

2.2.1 Technische Vorgaben

Sprache min. Java 8 / Entwickelt mit Java 13

IDE Zu Beginn keine, eventuell wird im Verlauf des Semesters auf Eclipse/IntelliJ umgestiegen

Versionierung Git, mit Hilfe von GitHub und Gitkraken, für Teilnehmer nur optional

2.2.2 Genre

Noch offen

2.2.3 Konzept

Noch offen

2.2.4 Story

Folgende Storyelemente wurden bisher erarbeitet:

Zeit Industrialisierung, 18./19. Jahrhundert

Ort London, England

Lebensraum Industriegebiet/Armenviertel

Keywords Rebellengruppe, Kommunismus, Steampunk, Kriegstreibende Regierung, Drogenkriminalität

Main Character Jugendlich/Teenager, Arbeiterfamilie, Indisch/Irische Herkunft, Teil der Rebellengruppe

3 Anforderungen

3.1 Funktionale Anforderungen

3.2 Qualitäts Anforderungen

4 Entwicklung

Dieses Kapitel enthält eine kurze Übersicht über die in den einzelnen Übungswochenen erarbeiteten Themen und den damit verbundenen Projektteilen.

4.1 Woche 1 - Grundlagen

KEYWORDS

Algorithmen, Zahlenbasen, Horner-Schema, Java Mini-Workflow

In Woche 1 (hier auch Woche 0 miteinbezogen, diese hat kein eigenes Kapitel verdient) wurde als Grundlage für die weiteren Wochen gezeigt, wie eine Kommandozeile zu verwenden ist und wie man mit dieser ein einfaches Java-Programm kompiliert (*übersetzt*) und ausführt. Wichtig dabei sind die Befehle **javac** zum compilen und **java** zum Ausführen. Ein Programm in einer Datei namens *Hello.java* würde man also mit folgenden Befehlen ausführen:

```
.../> javac Hello.java

.../> java Hello
Hello World!
```

Außerdem wurde in dieser Woche über Zahlensysteme gesprochen. Es ist bei der Programmierung oft hilfreich und manchmal notwendig das Konzept binärer und anderer Zahlen zu kennen. Eine Zahl im allgemeinen ist eine Aneinanderreihung von Ziffern, die Stelle der Ziffer in dieser Reihung bestimmt deren Gewichtigkeit.

So ist im Dezimalsystem die Zahl 123_{10} zu verstehen als $1*10^2+2*10^1+3*10^3$. Entsprechend wäre die Zahl 1010_2 im Binärsystem zu verstehen als $1*2^3+0*2^2+1*2^1+0*2^0=8+2=10_{10}$. Die Umrechnung zwischen verschiedenen Systemen kann mittels des **Horner-Schemas** erreicht werden (vgl. Vorlesung).

Vor allem aber wurde in dieser Woche bereits die Entwicklung von Algorithmen besprochen. Dies geschah über die Schritte **Problemspezifikation, Problemabstraktion, Algorithmenentwurf, Verifikation und Aufwandsanalyse**. Mit diesen Schritten wurde für das Projekt versuchsweise ein Algorithmus erarbeitet, der einer Punkte-Berechnung am Ende eines Spiels darstellen könnte. Es entstand folgender Pseudocode:

```
Algorithm "Check Won":

input s, p, c;
c = c + round_down(p / 10);
while c > 1:
  if c % 2 == 0
    c = c / 2;
    s = s + 1;
  else
    c = 0
  end if
end loop

if s > 10
  print 'spieler hat gewonnen'
else
  print 'spieler hat verloren'
end if
```

Es wurde bestimmt, dass dieser Algorithmus einen Aufwand der Form $log(n)$ aufweist.

Zuletzt wurde auch eine **Java-Klasse** als Start für unsere Anwendung aufgebaut. Diese enthielt nur eine simple main-Methode und eine Begrüßung des Nutzers:

```
"Application.java":

import java.util.Scanner;

public class Application {

  public static void main(String[] args) {

    // create a new Scanner that lets us read in input from the user
    Scanner scan = new Scanner(System.in);
    System.out.print("Welcome to <placeholder>, what is your name?");
    String name = scan.nextLine();
    System.out.print("Hello " + name + "!");
    scan.close();
  }
}
```

4.2 Woche 2 - Erste Abfragen

KEYWORDS

Pseudocode, Datentypen, Boolsche Ausdrücke, Menüführung

In der zweiten Übungswoche wurde das Konzept der Algorithmenentwicklung mit Pseudocode wiederholt. Zudem wurden Grundlegende Datentypen in Java kennen gelernt. Die primitiven Datentypen sind dabei: **byte, short, int, long, char, float, double und boolean**. Als nicht-primitiver Datentyp wurden **Strings** angesprochen. Es wurde erklärt wie diese Datentypen den zugehörigen Speicher belegen und in welcher Situation welcher Datentyp sinnvoll wäre.

Auch wurde bereits das Konzept Logischer Ausdrücke und die Verwendung von Wahrheitswerten besprochen. Dazu wurde die Funktion von Negation !, logischem und &&, logischem oder || und dem Vergleich == erklärt und angewandt.

Mit diesen nun zugänglichen Grundlagen wurde für das Projekt dann gemeinsam ein simples Anwendungsmenü aufgebaut. Es wurde ermittelt, dass ein Nutzer über die Eingabe verschiedener Zahlen entsprechende Punkte des Menüs erreichen kann. Über sinnvolle Abfragen im Programm soll die Eingabe des Nutzers mit den vorgegebenen Punkten abgeglichen werden. Dabei enstand folgendes Menü:

```
"Application.java" - Main Method

[...]
  public static void main(String[] args)
    // create a new Scanner that lets us read in input from the user
    Scanner scan = new Scanner(System.in);
    System.out.print("Welcome to <placeholder>, what is your name?");
    String name = scan.nextLine();
    System.out.print("Hello " + name + "!");

    // Let user choose an option
    System.out.print("What do you want to do?");
    System.out.print("\t(1) Play Game");
    System.out.print("\t(2) Options");
    System.out.print("\t(3) Loading");
    System.out.print("\n::MENU::> ");
    int opt = scan.nextInt();
```

```
    // The menu options
    System.out.println("You have entered option: " + opt);

    if(opt == 1) {
      System.out.print("This is game");
    } else if(opt == 2) {
      System.out.print("This is options");
    } else if(opt == 3) {
      System.out.print("This is loading");
    } else {
      System.out.print("This is not a valid input");
    }
    scan.close();
  }
[...]
```

4.3 Woche 3 - Rechnungen/Mini-Game

KEYWORDS

Schleifen, Abfragen, Rechnungen, switch-case, Ausgaben

In der dritten Woche wurde der Übergang von Algorithmen zu Programmen behandelt. Es wurden grundlegende Konzepte von Algorithmen mit Java-Syntax angesprochen und ein erstes Programm **implementiert**.

Dadurch war es möglich nicht nur das bestehende Menü dieser Anwendung in einen switch-case umzubauen, sondern auch bereits ein erstes kleines Mini-Spiel zu programmieren. Hierzu wurde im Menü-Punkt eins testweise ein Spiel mit folgenden Regeln entwickelt:

- Ein Spieler muss einfache Rechenaufgaben lösen
- Es gibt pro Runde zehn Rechnungen
- Gewonnen hat ein Spieler wenn mindestens 7 Aufgaben richtig gelöst wurden
- Außerdem hat ein Spieler nur 60 Sekunden Zeit pro Runde

Hierbei wurde nun einmal die funktionsweise von **for-Schleifen** betrachtet, die Möglichkeit Zahlen vom Benutzer einzulesen, die Möglichkeit Zahlen und Ergebnisse in **Variablen** abzuspeichern, zufällige Fragen zu generieren und mit Hilfe der **Java API** die benötigte Zeit zu messen. Es entstand folgender Code:

```
// declare variables
int correct = 0;
long start = System.currentTimeMillis();

// loop 10 times
for(int i = 0; i < 10; i++) {
  int x = (int)(Math.random() * 100);
  int y = (int)(Math.random() * 100);
  int ergebnis = x + y;

  // question to user
  System.out.println("Frage " + i + ": " + x + " + " + y);

  // input from user
  int input = scan.nextInt();

  // check if input is correct
  if(input == ergebnis) {
    System.out.println("CORRECT");
    correct++;
  } else {
    System.out.println("FALSE");
  }
}

long end = System.currentTimeMillis();
long duration = (end - start) / 1000;

// check if won
if(duration < 60 && correct >= 7) {
  System.out.println(name + ": YOU WON!");
} else {
  System.out.println(name + ": YOU LOST");
}
```

4.4 Woche 4 - Strings und Schleifen

KEYWORDS
Schleifen Part 2, String-Operationen

Die vierte Woche diente dazu weitere Schleifen-Arten kennen zu lernen. Es wurde besprochen wie unterschiedliche Schleifen aufgebaut sind und wie sie sich zueinander verhalten. Es wurden **for-, while-, und do-while-Schleifen** erlernt und angewandt.

Desweiteren wurden in dieser Woche die verschiedenen Operationen auf Strings vorgestellt, welche sehr oft verwendet werden. Dazu zählt die Ermittlung der Länge eines Strings mit **myString.length()** und die Abfrage nach Zeichen an einer bestimmten Stelle im String mit **myString.charAt(index)** (index sollte eine Zahl zwischen 0 und myString.length() - 1 sein, vgl. Tutorium). Wichtig ist hierbei auch die Gleichheitsabfrage auf Strings mit **myString.equals(otherString)**, da es sich bei Strings nicht um primitve Datentypen handelt und die Verwendung von '==' hier zu Fehler führen kann.

Die Verbindung dieser Operationen mit for-Schleifen lässt uns eine ganze Reihe an String-basierten Methoden entwickeln. Dazu gehören Zeichen- und Mustersuche, Ersetzungen, Textgenerierung und weiteres.

Diese Konzepte wurden in unserer Anwendung in einer 'Random Name Challenge' verwendet. Die Idee war bei der Charakter Erstellung am Beginn des Spiels dem Spieler die Möglichkeit zu geben einen zufälligen Name generieren zu lassen. Grundlage war hier eine eigens erdachte, Silbenbasierte Namens-Syntax, die wie folgt beschrieben werden kann:

- Eine Silbe besteht aus einem Konsonanten gefolgt von einem Vokal
- Ein Name ist eine Aneinanderreihung mehrerer Silben
- Die erste Silbe eines Namens kann auch Nur ein Vokal sein, aber nur fallls ihr eine weitere Silbe folgt
- Ein Name sollte nicht länger als 5 Silben lang sein (Plus eventuelle erste Vokal-Silbe)

Hierfür wurden nun Konsonanten und Vokale in eigene Strings gepackt um später über einen zufälligen Index jede mögliche Silbe erhalten zu können. Dies war mit den oben genannten Operationen ohne weitere Probleme mögliche. Es entstand folgender Code:

```
// prepare variables to generate a random name
String name = "";
String vokale = "aeiou";
String konsonanten = "bdfghklmnprstyz";
int length = 1 + (int)(Math.random() * 5);
char vokal;
char konsonant;
int konIndex;
int vokIndex;

// maybe add a vocal to start of name
if(Math.random() > 0.5) {
  vokIndex = (int)(Math.random() * vokale.length());
  vokal = vokale.charAt(vokIndex);
  name = name + vokal;
}

// generate a few syllables
for(int i = 0; i < length; i++){
  vokIndex = (int)(Math.random() * vokale.length());
  vokal = vokale.charAt(vokIndex);
  konIndex = (int)(Math.random() * konsonanten.length());
  konsonant = konsonanten.charAt(konIndex);
  name = name + konsonant + vokal;
}

System.out.print("Your name is now: " + name);
```

4.5 Woche 5 - Arrays

KEYWORDS

Arrays, Mehrdimensionale Arrays

In Woche fünf wurden nun komplexere Datentypen in Form von Arrays vorgestellt. Arrays dienen in Java zur Abbildung von Listen und Vektoren in statischer Form. Initialisiert werden Arrays anders als die bisherigen Datentypen über den Befehl *new*: **int[] myFirstArray = new int[10];** wäre ein Beispiel zur Erzeugeung eines Arrays mit 10 Plätzen. Diese Länge 10 des Arrays ist fest und kann später

nicht mehr verändert werden. Um auf die Plätze in einem Array zuzugreifen reicht eine einfache Angabe des Index in eckigen Klammern:

myFirstArray[2] = 42;
int solToUniv = myFirstArray[2];

So lassen sich nun logisch zusammengehörige Reihen einfach speichern und verändern. Da man diese Arrays auch als Vektoren betrachten kann liegt es Nahe auch mehrdimensionale Vektoren, also Matrizen, mit Arrays darzustellen. Dies ist durch einfache Mehrfachklammerung bei der Deklarierung möglich:

int[][] myMatrix; // 2-dim-Vektor
int[][][] highDim; // 3-dim-Vektor

Vorstellen kann man sich diese Matrizen als 'Arrays von Arrays', man sollte bedenken, dass es einfache Auflistungen von weiteren Arrays sind. Die Indizierung erfolgt von außen nach innen: Ein Aufruf an myMatrix[2][1] bedeutet, dass man im äußeren Array an die Position 2 schaut. Dort befindet sich ein inneres Array. Bei diesem schaut man dann an Stelle 2. Dort befindet sich der gesuchte Wert. Wichtig auch hier zu beachten, ist das die Stellen bei 0 starten. Siehe folgende Darstellung:

myMatrix = {
0: $\{0 : a,\ \ 1 : b,\ \ 2 : c\}$,
1: $\{0 : d,\ \ 1 : e,\ \ 2 : f\}$,
2: $\{0 : g,\ \ 1 : h,\ \ 2 : i\}$
}

Wobei die Zeichen *'a',...,'i'* für beliebige Integer stehen. Der Wert myMatrix[2][1] wäre hier also **h**, der Wert myMatrix[0][2] entsprechend **c**.

Arrays sind also vielseitig einsetzbar. Für eine erste Übung wurde in der Anwendung die Verwendung eines Arrays als Inventar für den Spieler erdacht. Idee war in dieser Woche erste Story-Elemente zu entwickeln und eine Grundlage für die Interaktion des Spielers mit verschiedenen Items zu erstellen. Dafür wurden mögliche Items in einem ersten Array angelegt und das Inventar als weiteres, leeres Array eingeführt. Es wurden nun in einer Schleife je ein zufälliges Item ermittelt und der Nutzer wurde gefragt ob dieses Item in das Inventar gelegt werden sollte. Es entstand folgender Code:

```
// items
String[] items = new String[]{"Crowbar", "Schere", "Stift", "Fackel", "Buch"}
// Inventar
String[] inventory = new String[10];

for(int i = 0; i < inventory.length; i++) {
  inventory[i] = "empty";
}
```

```
// the game loop
while(true) {
  String foundItem = items[(int)(Math.random() * items.length)];
  System.out.println(name + " found " + foundItem);
  String q = "Pick up?";
  String yesOption = "y";
  String noOption = "n";
  System.out.print(q + "\n\t(" + yesOption + ")\n\t(" + noOption + ")\n\t::>");
  String answer = scan.nextLine();

  if(answer.equals(yesOption)) {
    for(int i = 0; i < inventory.length; i++) {
      if(inventory[i].equals("empty")) {
        inventory[i] = foundItem;
        break;
      }
    }
    System.out.println(name + " picked up " + foundItem);
  } else {
    System.out.println("Nothing was picked up");
  }
}
```

Klar ist, dass in diesem Beispiel viele Fälle noch nicht abgefragt wurden: was wenn das Inventar voll ist? Wie kann das Inventar betrachtet werden? Kann man ähnliche Items stapeln? Kann man Items aus dem Inventar wieder entfernen? Diese Fragen sollten sich interessierte Teilnehmer selber überlegen, eine verbesserte Version wird vermutlich im Lauf des Entwicklungsprozesses eingebaut werden.

4.6 Woche 6 - Methoden

KEYWORDS

Methoden (Syntax, Aufruf, Parameter, Sichtbarkeit), globale Variablen

---

In dieser Woche wurden **Methoden** (oder auch Funktionen/Unterprogramme/...) eingeführt. Speziell wurde auf deren Syntax, Verwendung und die zugehörigen Themen **Sichtbarkeit, Überdeckung und Überladung** eingegangen.

Eine Methode ist ein in sich angeschlossenes Teilprogramm, welches von anderen Stellen im Programm verwendet werden kann. Sie stehen in Java innerhalb einer Klasse aber außerhalb der main-Methode. Es können Werte an die Methode übergeben werden und es kann ein Wert von der Methode zurück gegeben werden. Ein Methodenkopf ist folgendermaßen aufgebaut:

[<Zugriff>][static]<Typ><Name>([<Paramter>]*)

Beispiele:

public static int laenge(String testString)

static void ausgabe(int eins, int zwei, int drei)

Der Typ einer Methode gibt an um welchen Typ es sich bei der Rückgabe handelt. Wird void angegeben kann die Methode nichts zurück geben. Wird nicht void angegeben **muss** die Methode in jedem Fall eine Rückgabe mit diesem Wert machen. Dies erfolgt über eine oder mehrere Angaben der **return** Anweisung.

Beispiel:

```
public static int laenge(String testString) {
  if(testString == null) {
    return 0;
  } else {
    int laengeDesStrings = testString.length();
    return laengeDesStrings;
  }
}
```

Wichtig zu beachten ist hier, dass die Angabe von **return** nicht nur eine Rückgabe darstellt, sondern auch das Unterprogramm an dieser Stelle beendet. Für diese Terminierung wird dieser Befehl daher auch in void-Methoden verwendet.

Im Rahmen der Entwicklung des Projektes wurden zwei Schritte durchgeführt. Einmal wurde versucht bestehende Code-Fragmente in Methoden auszulagern. Danach wurden gegebene Methoden an entsprechender Stelle mit Aufrufen eingebaut.

Im ersten Teil wurde zum Beispiel der Menü-Punkt 'Optionen' in eine eigene Methode verfrachtet und an die Originale Stelle ein Aufruf an diese Methode gesetzt:

```
[...]
case 2:
  options();
  break;
case 3:
[...]
```

```
[...]
public static void options() {
  // implement options
}
```

Dies wurde auch für weitere Teile des Codes getestet. Im zweiten Teil waren nun folgende Methodendeklarationen gegeben:

- *pause(long millis): void*, hält das Programm for 'millis' Millisekunden an
- *clear(): void*, entpricht dem Befehl clear in der Konsole
- *title(String text): void*, gibt einen schön formatierten Titel in der Konsole aus
- *print(String text): void* gibt einen einfachen text aus
- *getAnswer(String question, String... options): String*, fragt 'question' den Nutzer und gibt die Antwort des Nutzers als String zurück. Die String... options stehen für eine beliebige Anzahl an Strings, die hier für möglich Antwort-Möglichkeiten stehen
- *show(String name, String[] list): void*, zeigt eine Liste an Strings mit Titel 'name' auf der Konsole an

Diese wurden an entprechener Stelle im Programm verwendet um das 'Look and Feel' der Anwendung zu verbessern.

Beispiel:

```
[Methode 'playGame()']

// the method where the game logic is implemented
public static void playGame() {
  characterCreation();
  clear();
  title("START GAME");
  pause(2000);
  clear();
  title("PLACEHOLDER");
  pause(2000);
  clear();
  title("AND NOW THE STORY BEGINS...");
  System.out.println();
```

```
  pause(2000);

  // Preparing Gameplay Parameters
  // items
  String[] items = new String[]{"Crowbar", "Schere",
     "Stift", "Fackel", "Buch"};
  // Inventar
  String[] inventory = new String[10];

  for(int i = 0; i < inventory.length; i++) {
    inventory[i] = "empty";
  }

  // the game loop
  while(true) {
    clear();
    title("ITEM FOUND");
    String foundItem = items[(int)(Math.random() * items.length)];
    print(mainCharacter.name + " found " + foundItem);
    pause(1000);
    String q = "Pick up?";
    String yesOption = "y";
    String noOption = "n";
    String answer = getAnswer(q, yesOption, noOption);

    if (answer.equals(yesOption)) {
        for(int i = 0; i < inventory.length; i++) {
            if(inventory[i].equals("empty")) {
                inventory[i] = foundItem;
                break;
            }
        }
        System.out.println(name + " picked up " + foundItem);
    } else {
        System.out.println("Nothing was picked up");
    }
    pause(1000);
  }
}
```

4.7 Woche 7 - Objektorientierung

KEYWORDS

Grundlagen OO, Objekte, Instanzen, Konstruktoren, Parameterübergabe

---

5 Implementierung

5.1 Framework

5.2 Architektur

5.3 Vorstellung der Software

6 Einordnung

6.1 Verwandte Arbeiten

6.2 Evaluation

6.3 Ausblick