W

# wawi03

Vorführaufgabe 3

| Name | Last update |
| --- | --- |
| .idea (/pg2/wawi03/tree/master/.idea) | about 12 hours ago |
| images (/pg2/wawi03/tree/master/images) | about 12 hours ago |
| lib (/pg2/wawi03/tree/master/lib) | about 12 hours ago |
| src/gui (/pg2/wawi03/tree/master/src/gui) | about 12 hours ago |
| .gitignore (/pg2/wawi03/blob/master/.gitignore) | about 12 hours ago |
| Kundenauswahl.jpg (/pg2/wawi03/blob/master/Kundenauswahl.jpg) | about 12 hours ago |
| Produktauswahl.jpg (/pg2/wawi03/blob/master/Produktauswahl.jpg) | about 12 hours ago |
| README.md (/pg2/wawi03/blob/master/README.md) | about 12 hours ago |
| wawi03.iml (/pg2/wawi03/blob/master/wawi03.iml) | about 12 hours ago |

**README.md** (/pg2/wawi03/blob/master/README.md)

# Entwicklung eines Kassensystems

In kleineren Shops und Restaurants haben Sie sicherlich schon einmal beobachtet, dass das Personal die Bestellungen und Abrechnungen mit Smartphones oder Tablets statt mit Registrierkassen oder PCs erledigen. Im Rahmen dieser und noch zwei darauf aufbauender Vorführaufgaben werden Sie ein stark vereinfachtest Kassensystem (auch "Point of Sale"-System, POS-System, genannt) in Java entwickeln.

## Vorführaufgabe 3

### Behandelte Themen

Exceptions, Swing

### Vorbereitungen

Verwenden Sie Ihr bereits bestehendes Projekt *wawi02* weiter und erweitern Sie es um die nachstehenden Aufgabenstellungen.

## Aufgabe 1: Exceptions

Bisher wurden auftretende Fehler (z. B. nicht ausreichender Lagerbestand beim Auslagern) nicht behandelt. Dies soll nun durch Einsatz von zwei Exception-Klassen verbessert werden. Legen Sie vorbereitend hierfür das Package `exception` an

Die Klasse `BookingException` stellt eine Möglichkeit zur Behandlung von Buchungsfehlern bereit.

- Eine `BookingException` ist eine *checked exception*.
- Ein Objekt der Klasse `BookingException` kann nicht ohne eine Fehlermeldung angelegt werden.

Die Klasse `OutOfStockException` soll einen nicht ausreichenden Lagerbestand beim Auslagern anzeigen.

- Eine `OutOfStockException` ist eine *checked exception*.
- Eine `OutOfStockException` hat ein Produkt.
- Ein Objekt der Klasse `OutOfStockException` kann nicht ohne ein Produkt und nicht ohne eine Fehlermeldung angelegt werden.

Die beiden oben beschriebenen Klassen sollen folgendermaßen verwendet werden:

- Beim Auslagern von Artikeln sollte eine `OutOfStockException` geworfen werden, sofern der Lagerbestand nicht ausreicht.
- Beim Hinzufügen einer Rechnungsposition zu einer Rechnung sollte eine `OutOfStockException` geworfen werden, sofern der Lagerbestand nicht ausreicht. Bitte beachten Sie, dass dies nur für Artikel gilt!
- Beim Hinzufügen einer Rechnungsposition zu einer Rechnung bzw. beim Buchen einer Rechnung sollte eine BookingException geworfen werden, sofern die Rechnung nicht den Status `IN_ERSTELLUNG` hat.
- Die geworfenen Exceptions sollten im Bereich der GUI (s. Aufgabe 2) gefangen werden, im Bereich der Geschäftsobjekte (z. B. beim Buchen einer Rechnung) werden Exceptions einfach weiter geworfen. Ergänzen Sie entsprechende `throws` -Klauseln in den Methodenköpfen.

## Aufgabe 2: Entwicklung der Grafischen Benutzeroberfläche

#### Vorbereitungen

- Der `POS` -Dialog (siehe nachfolgende Abbildung) ermöglicht die Auswahl eines Produkts (Artikel oder Dienstleistung) über spezielle Buttons. Jeder dieser Buttons ist eine Instanz der vorgegebenen Klasse `ProduktButton` . Laden Sie diese Klasse aus dem Moodle-Kursraum und kopieren Sie diese in ein Package namens „gui“. Auch die noch zu definierende Klasse namens „POS“ soll in diesem Package definiert werden.
- Bilddateien für die Produktbuttons werden im Folder namens `images` gespeichert.

Hauptziel dieser Aufgabe ist die Definition einer Klasse namens *POS* (englisch *Point of Sale*). `POS` spezialisiert `JFrame` und bietet eine Benutzeroberfläche für die Auswahl der abzurechnenden Produkte und die schlussendliche Abrechnung (*Checkout*).

`POS` befindet sich stets in einem von zwei Modus:

- der **Produktauswahl** (siehe Abbildung oben) und
- der **Kundenauswahl** (siehe Abbildung unten)

Der gesamte Ablauf wird im Folgenden beschrieben:

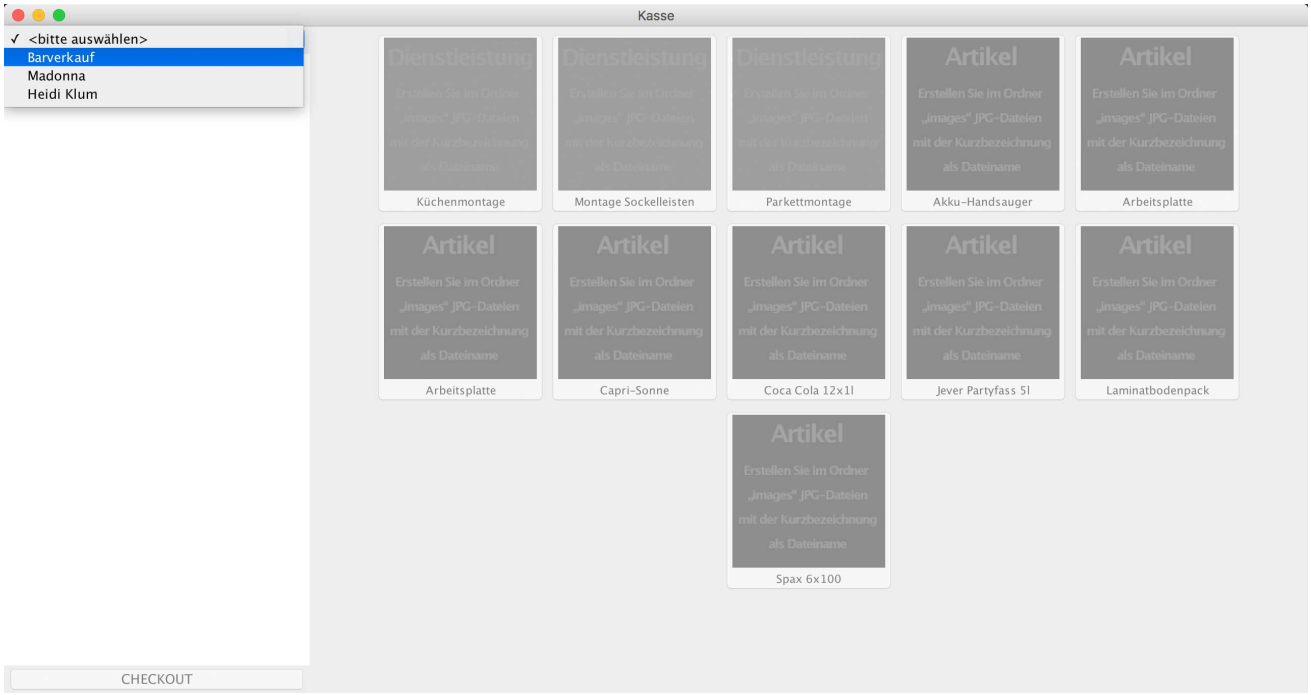
- Nach dem Erzeugen des Dialogs erscheint dieser im Modus **Kundenauswahl** (siehe Abbildung 1)
  - Während der Kundenauswahl ist nur die Auswahl des Barverkaufs (Eintrag „Barverkauf“) oder eines Kunden (Eintrag ist der jeweilige Kundenname) über die Combobox links oben möglich, d. h. die Produktbuttons sowie der Checkout-Button sind gesperrt.
  - Der erste Eintrag der Combobox lautet `<bitte auswählen>` und ist im Kundenauswahlmodus zunächst selektiert.

- Mit Auswahl eines Eintrags ungleich `<bitte auswählen>` wird ein neues Rechnungsobjekt angelegt, welchem der ausgewählte Kunde übergeben wird.
  - Die GUI ist jetzt im Modus **Produktauswahl** (siehe Abbildung 2)
  - Im Falle eines Barverkaufs wir das Rechnungsobjekt ohne Parameter erzeugt.
  - Die noch leere Rechnung wird im Textfeld dargestellt und sämtliche Buttons, für deren Produkte ein Lagerbestand vorhanden ist, werden entsperrt (gilt nur für Artikel!).
  - Die Kundenauswahl-Combobox wird gesperrt.

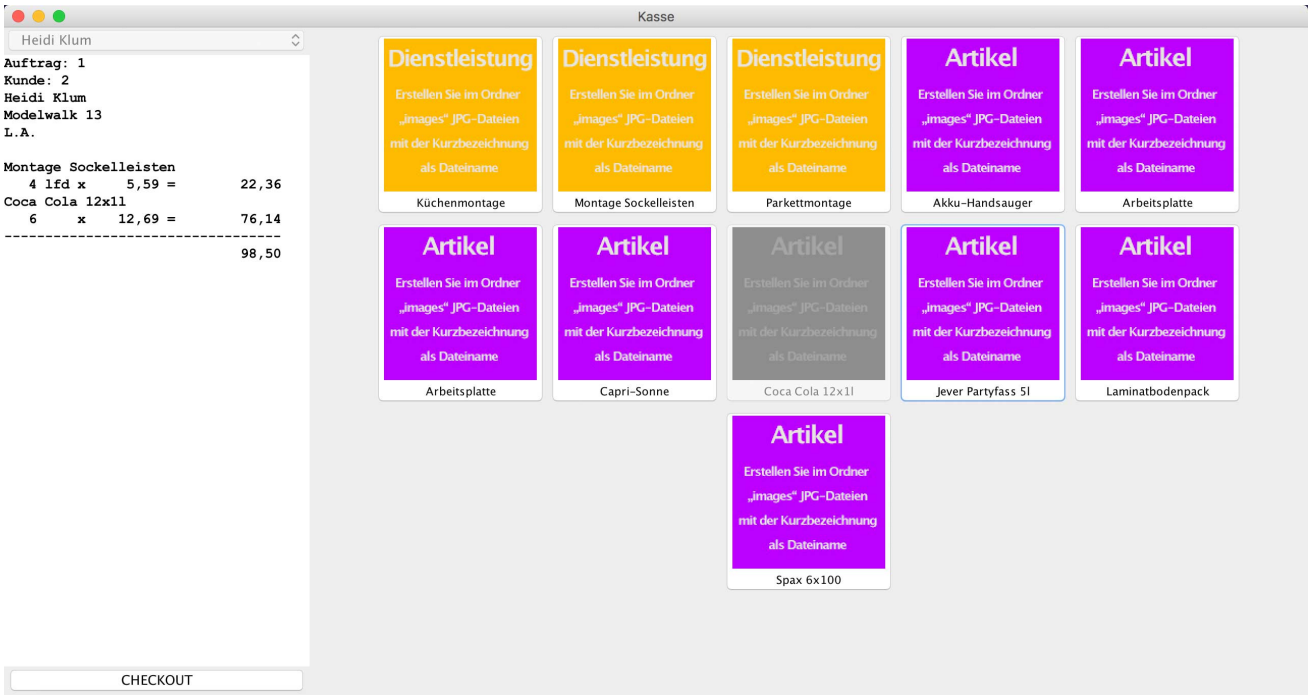
- Wird ein Produktbutton gedrückt, so wird das gewählte Produkt als neue Rechnungsposition hinzugefügt bzw. in eine bestehende entsprechend erweitert.
  - Nutzen Sie hierzu die entsprechende Methode der Klasse Rechnung.
  - Anschließend wird die Rechnungsanzeige im Textfeld aktualisiert.
  - Für den Fall, dass der Lagerbestand eines Artikels durch die Aktion auf 0 reduziert wurde, wird der Produktbutton gesperrt.

- Wird der Checkout-Button gedrückt, so wird die aktuelle Rechnung gebucht.
  - Anschließend werden alle Produktbuttons und der Checkout-Button gesperrt.
  - Die Kundenauswahl-Combobox zeigt auf den ersten Eintrag ( `<bitte auswählen>` ) und wird entsperrt.
  - Die Anwendung befindet sich damit wieder im Modus **Kundenauswahl**.

- Über den Schließen-Button des Frames/Windows kann die Anwendung jederzeit beendet werden.




(/pg2/wawi03/raw/master/Kundenauswahl.jpg) **Abbildung 1: Kassendialog im Kundenauswahlmodus**




(/pg2/wawi03/raw/master/Produktauswahl.jpg) **Abbildung 2: Kassendialog im Produktauswahlmodus**

Definieren Sie nun die Klasse `POS` u. a. mit den im nachfolgenden Codeausschnitt aufgeführten Methoden:

```
private static List<Produkt> initProducts() {
    List<Produkt> produkte = new LinkedList<>();
    Artikel p1 = new Artikel(12345, "Arbeitsplatte", 89.90);
    p1.einlagern(1);
    Dienstleistung p2 = new Dienstleistung(100,
         "Küchenmontage", 75., "h");
    Artikel p3 = new Artikel(98989876,
         "Akku-Handsauger", 129.90);

    p3.einlagern(3);
    Artikel p4 = new Artikel(5261, "Spax 6x100", 3.99);
```

```
    Artikel p4 = new Artikel(5261, "Spax 6x100", 5.99);
    p4.einlagern(4);
    Artikel p5 = new Artikel(4593, "Coca Cola 12x1l", 12.69);
    p5.einlagern(5);
    Artikel p6 = new Artikel(4594, "Capri-Sonne", 8.99);
    p6.einlagern(6);
    Artikel p7 = new Artikel(4595, "Jever Partyfass 5l", 8.99);
    p7.einlagern(7);
    Artikel p8 = new Artikel(12346, "Arbeitsplatte", 99.90);
    p8.einlagern(1);
    Artikel p9 = new Artikel(526, "Laminatbodenpack", 13.99);
    p9.einlagern(5);
    Dienstleistung p10 = new Dienstleistung(123,
         "Parkettmontage", 75.00, "qm");
    Dienstleistung p11 = new Dienstleistung(128,
         "Montage Sockelleisten", 5.59, "lfdm");
    produkte.add(p1); produkte.add(p2);
    produkte.add(p3); produkte.add(p4);
    produkte.add(p5); produkte.add(p6);
    produkte.add(p7); produkte.add(p8);
    produkte.add(p9); produkte.add(p10);
    produkte.add(p11);
      return produkte;
}

public static void main(String args[]) {
    List<Produkt> produkte = initProducts();
    List<Kunde> kunden = new ArrayList<>();
    kunden.add(new Kunde(-1, "<bitte auswählen>", "", ""));
    kunden.add(new Kunde(0, "Barverkauf", "", ""));
    kunden.add(new Kunde(1, "Madonna", "Sunset Boulevard 1",
        "Hollywood"));
    kunden.add(new Kunde(2, "Heidi Klum", "Modelwalk 13",
        "L.A."));
    new POS(produkte, kunden);
}
```

Die `main()`-Methode erzeugt u.a. eine neue `POS`-Instanz. Nach Ausführung des Konstruktors soll der Dialog wie in Abbildung 2 gezeigt erscheinen. Bitte beachten Sie die Reihenfolge der dargestellten Produktbuttons.

Ergänzen Sie die Klasse „POS“ nun um sämtliche fehlenden Eigenschaften, um die beschriebene Funktionalität zu erhalten. In einem ersten Schritt sollten Sie den Konstruktor definieren, der die benötigten Layout-Manager und die Oberflächen-Bedienelemente (UI-Controls) erzeugt und diese passend verknüpft. In einem zweiten Schritt ergänzen Sie dann bitte die benötigte Ereignisbehandlung.

Für den in der Klasse `POSTest` enthaltenen Test der Klasse `POS` werden folgende Getter benötigt, die Sie entsprechend ergänzen müssen:

- `public List<Kunde> getKunden()` : liefert die im Konstruktor übergebene Kundenliste.
- `public List<Produkt> getProdukte()` : liefert die im Konstruktor übergebene Produktliste.

- `public JTextArea getTextFeld()` : liefert das Textfeld, in dem die aktuelle Rechnung dargestellt wird.
- `public Rechnung getRechnung()` : liefert die aktuelle Rechnung.
- `public JButton getCheckoutButton()` : liefert den Checkout-Button.
- `public JComboBox<Kunde> getKundenComboBox()` : liefert die Kunden-Combobox.
- `public ProduktButton[] getProduktButtons()` : liefert die Produktbuttons.

Bitte beachten Sie auch folgende Hinweise zur Bearbeitung:

- Layout der Textfeldes für die Rechungsdarstellung: stellen Sie die Größe des Textfeldes durch folgende Anweisung ein
  - `<ihreTextfeldReferenz>.setPreferredSize(new Dimension(300, 600));`

- Einstellen des Schriftsatzes des Textfeldes: stellen Sie den Schriftsatz folgendermaßen ein:
  - `<ihreTextfeldReferenz>.setFont(new Font("Courier New", Font.BOLD, 14));`

- Belegung der Kundenauswahl-Combobox: Wandeln Sie die im Konstruktor erhaltene Kundenliste in eine geeignete Datenstruktur um und übergeben Sie diese bei der Erzeugung der Combobox. Ändern Sie die `toString()` -Methode der Klassen `Kunde` so ab, dass nur das Attribut `name` in der Combobox angezeigt wird.

## Tests

Dieses Projekt enthält mehrere Testklassen, die (analog der Tests im Praktomat) Sie selbst lokal ausführen können. Mit einem rechten Mausklick auf das Verzeichnis `test` können Sie alle Test entsprechend über das Kontextmenü starten. Die Testergebnisse (grün oder rot) sehen Sie in einem entsprechenden Fenster der Entwicklungsumgebung.

## Abgabe

Die Abgabe und automatische Testierung erfolgt über den *Praktomat* (http://praktomat3.oth-regensburg.de), indem Sie dort all Ihre entwickelten Klassen aus den Packages `geschaeftsobjekt` , `exception` und `gui` einzeln oder (viel besser) als ZIP-Datei hochladen.