

Softwareentwicklung 2

Wintersemester 2023/24

**Projekt BudgetBloom**

Ein Bild, das Blume, Zeichnung, Kinderkunst, Rose enthält.

Automatisch generierte Beschreibung

**Ein Projekt von:**

Joel Starkov - js486

Ryan Röhrich – rr062

Jana Weißflog – jw184

<https://gitlab.mi.hdm-stuttgart.de/js486/se2budgetboom>

**1. Kurzbeschreibung**

Bei unserem Projekt handelt es sich um einen Budget-Planer.

Die Benutzer haben die Möglichkeit, ein Konto anzulegen und sich einzuloggen. Bei der ersten Anmeldung gibt man seinen momentanen Kontostand an. Für jeden Benutzer wird eine individuelle Tabelle angelegt. In dieser können neue Einträge, also Einnahmen oder Ausgaben, hinzugefügt, aber auch bearbeitet oder gelöscht werden. In der Tabelle werden neben dem aktuellen Kontostand auch das Datum, der Grund, der Betrag, die Wichtigkeit, die Regelmäßigkeit und die Zahlungsart der Einnahme oder Ausgabe angezeigt.

**2. Startklasse**

Die Main-Methode/Driver befindet sich in der Klasse mainpackage.

**3. Besonderheiten**

Im Moment gibt es kaum erwähnenswerte Besonderheiten, die in unserem Projekt vorhanden sind. Trotzdem würden wir kurz einige Punkte erläutern bzw. klarstellen.

1. Bei der Registrierung muss keine persönliche E-Mail angegeben werden. Es reicht, wenn es ein String mit der Länge zwischen 6 und 69 ist. Es wird außerdem nicht nach ‚@‘ oder ähnlichem geprüft.
2. Wenn Sie ein bereits vorhandenes Konto mit einigen Datensätzen haben wollen, können Sie unser Testkonto benutzen:
   1. Benutzername: tester
   2. Passwort: 123456789!

Sie können andernfalls auch ein eigenes Konto erstellen.

1. Den Aufbau unserer Datenbanken können Sie in dem File „DatenbankenListe“ finden. Diese sind mit einer unserer Hochschuldatenbanken verbunden und werden von dort aus benutzt.
2. **Diagramme**

Die folgenden Diagramme wurden mit draw.io erstellt und sollten daher bestenfalls damit geöffnet werden.

* 1. UML- Diagramm:

Das UML- Diagramm finden Sie unter resources/Diagrams/Klassendiagramm.drawio

Anmerkung: Zur Übersichtlichkeit wurden die @FXML Attribute nicht im Klassendiagramm gelistet.

* 1. Use- Case- Diagramm:

Das Use- Case- Diagramm finden Sie unter resources/Diagrams/UseCaseDiagramm.drawio

* 1. Sequence- Diagramm:

Das Sequence- Diagramm finden Sie unter resources/Diagrams/SequenceDiagramm.drawio

Der durch das Sequence- Diagramm visualisierte Vorgang startet in der Klasse ControllerLogin und endet in der Klasse ControllerOverview. Der genaue Zeitpunkt des Starts und des Endes wurde kommentiert.

**5. Stellungnahme**

Zur Erleichterung der Lesbarkeit wurden Namen von Klassen, Packages, Interfaces etc. grün markiert.

1. **Interfaces/Architektur (das Projekt wurde auf acht verschiedene Packages verteilt):**
   1. Package Controller: Hier befinden sich die Controller, die mit den FXML-Dateien zusammenhängen
   2. Package Exceptions: Dieses Package enthält alle eigenen Exceptions und gliedert sich in weitere zwei Ordner auf:
      1. Package NewEntryExceptions: Hier sind alle Exceptions, die für die Klasse LogicNewEntry und ControllerNewEntry bestimmt sind.
      2. Package RegisterExceptions: Hier sind alle Exceptions, die für die Klasse LogicRegister bestimmt sind.
   3. Package Interfaces: In diesem Package befinden sich unsere zwei Interfaces
      1. ControllerNewEntry und ControllerEditEntry haben das EntryInterface implementiert
      2. Die Klassen: Cash, Creditcard, Girocard, OtherPayment und Paypal haben Payment implementiert
   4. Package Logic: Hier findet sich die Logik unseres Programms, insbesondere die Logik der Controller. Dabei gibt es noch zu erwähnen, dass die Klassen LogicNewEntry und LogicEditEntry von LogicSuperClass erben.
   5. Package mainpackage: Hier findet man unseren Driver
   6. Package PaymentMethod: In diesem Package finden sich unsere verschiedenen Zahlungsarten, die von dem zugehörigen Interface erben. Dabei können diese durch die Klasse PaymentFactory einfach aufgerufen werden.
   7. Package Singleton: Hier finden sich zwei Singletonklassen
   8. Package Threads: Darin befindet sich unser Thread
2. **Clean Code:**
   1. Kapselung von Attributen siehe Singleton-Klassen oder LogicTableEntry
   2. Durch LogicFacade wurden soweit fast alle Methoden nicht mehr zu public, bis auf Ausnahmen wegen dem Testordner
   3. Lose Kopplung bei beispielsweise der Verwendung der Factory -> Klasse LogicSuperClass
3. **Tests**
   1. Mehrere Tests, inklusive Negativtests im test- Ordner unter test->java->mainpackage.
4. **GUI**
   1. Unsere GUI ist vielfältig und besteht aus mehreren FXML- Dateien, die man im Ordner resources->FXML finden kann
   2. Alle FXML- Dateien wurden mit SceneBuilder erstellt, außer firstLogin.fxml
5. **Logging/ Exceptions**
   1. In unserem Projekt gibt es einige eigene Exceptions, die unter dem Package Exceptions zu finden sind.
   2. Unsere Log-Level bestehen aus mehreren Logstufen:
      1. Error: Es handelt sich um einen schwerwiegenden Fehler, es wurde ein schwerwiegender Error aufgefangen (z.B eine SQL-Exception). Hier werden die Exceptions zusätzlich ausgegeben, um einen besseren Überblick über den Fehler zu bekommen.
      2. Warn: Es ist zwar kein schwerwiegender Fehler aufgetreten, das Programm kann aber aus anderen Gründen nicht weiterlaufen (z.B der eingegebene Benutzername ist falsch oder zu kurz). Hier wird bewusst darauf verzichtet, die Exception mit auszugeben, da es zu unübersichtlich wäre und das Logging eindeutig genug ist, was für ein Fehler aufgetreten ist und wo er zu finden ist.
      3. Debug: Wichtige Informationen für den Entwickler (wichtige Variablen, wie weit ist das Programm im Moment etc.)
      4. Info: Informationen, die evtl den Kunden auch interessieren könnten (z.B wurde eine Zeile gelöscht...)
   3. Es wird bei uns in mehrere Files geloggt, es gibt jeweils ein File für alle logs, im Terminal wird ebenfalls alles geloggt.
6. **Threads**
   1. Unser Thread befindet sich im Package Threads. Er gibt im Hintergrund jede Minute Informationen über den höchsten und aktuellen Kontostand.
7. **Streams**
   1. Den parallelen Stream findet man ebenfalls in der Klasse BalanceThreads, welcher den aktuellen und höchsten Kontostand aus der Datenbankliste sortiert.