Gliederung

1. Aufgabenstellung
2. Projektplanung hinsichtlich der verwendeten Methoden und Werkzeuge
3. Anforderungsanalyse anhand der Aufgabenstellung
4. Programmentwurf mithilfe von Mockups und eines Use-Case Diagramm
5. Umsetzung in Java
6. Kritische Reflexion

Aufgabenstellung

* Eingabemaske, die Eingabe von Kreditdaten ermöglicht
* Zinsen berechnen
* Zurückzuzahlende Betrag einer Liste hinzufügen
* JSON-Datei mit den Kreditdaten der Kredite enthält erstellen

Projektplanung

* **Vorgehen mit Wasserfallmodell**
  + Einfaches Problem, daher keine stetigen Updates
  + Support Schritt weggelassen, da Programm nicht released werden soll
* **IntellIJ**
  + Einfache Entwicklung, dank sehr guten Code Completion
  + VCS sehr einfach, durch Git und GitHub Integration, zudem Maven vollintegriert
  + Zudem leichter dank Vorwissen innerhalb Gruppe
* **Git/GitHub**
  + Git in der Gruppe schon bekannt
  + Wahl auf GitHub, da Integration in IDE, zudem GitHub Desktop direkt verknüpft
  + Aufgrund kleinen Umfang, egal welche Plattform gewählt wird, da weitere Funktionen wie CI/CD Pipelines erst bei größeren Projekten interessant werden
* **MVC**
  + Aufgabenstellung lässt dies zu
  + Model - laden von Krediten
  + Controller - Ausführen aller Berechnungen
  + View - UI

Anforderungsanalyse

* Model - Vorgabe der Eigenschaften von Krediten
  + Kreditbetrag
  + Zinssatz
  + Laufzeit
  + Rückzahlbetrag
  + Zinsbetrag
* View - Eingabemaske, Liste der Kredite, Trigger der Berechnung
* Controller - Verwaltung der Kredite, Schreiben/Lesen von JSONs, Berechnung der Kredite
* 2 von 3 Schichten, da keine richtige Datenbank hinterlegt wird.
* Selbstständige Erweiterung der Aufgabe um folgende Punkte
  + **Kreditarten:** Annuitäts-, Fälligkeits- und Abzahlungskredit
  + Zahlart: monatlich, jährlich
  + Auswahl von Krediten in Liste, erneute Modifikation und Berechnung

Entwurf

* Use-Case
* UML-Diagramme
* Erklärung Berechnung
* Mockups

Umsetzung

* Grundsätzlich Maven Projekt, für Extensions
* **Model - Credit.java**
  + Orientiert an Java Spring, um Programm an DB anzuschließen und gleichzeitig eine API zu bieten können (nicht umgesetzt)
  + daher fast nur Getter/Setter
  + setParameters, um unkompliziert Objekt zu laden.
  + ansonsten Implementierung aller Attribute
* **Controller**
  + würde im Fall der Erweiterung zu einem REST Controller umgebaut werden können
  + Hier Punkt, um API Requests zu verarbeiten
  + Daher wird hier Verarbeitung der Daten gemanaged
  + createObject, saveObject, loadObjectById, loadAllObjects
  + Da keine API, sondern JSON Workaround mit convertObjectToJSON Methode
  + Benutzung von 2 Maven Repos: json-simple, jackson