Machen Sie sich nochmals den Algorithmus für die Berechnung der Prüfziffernberechnung klar. Als nächsten Schritt überführen Sie das Programm in die Objektorientierung wie folgt:

1) Modellieren Sie den Sachverhalt mit folgenden Objekten: Bank(-filiale), Bankkunde, Kreditkarte, Bankautomat. Diskutieren Sie die Beziehungen zu den Objekten. Gehen Sie von folgendem vereinfachten Sachverhalt aus: Eine Bank emittiert eine Kreditkarte. Ein Bankkunde beantragt eine Kreditkarte und bekommt diese nach erfolgreicher Prüfung seines Bankkontos. Auf dem Bankkonto sollte mindestens ein Betrag von x Euro liegen, um eine Genehmigung der Kreditkarte zu bekommen.
   Bevor ein Kunde am Bankautomat Geld abheben kann, wird eine Prüfziffernberechnung vorgenommen. Bei erfolgreicher Prüfung, erfolgt die Auszahlung des Geldes.
2) Implementieren Sie die Klassen. Zunächst in einem Paket und alles public. Später kapseln Sie die Attribute. Diskutieren Sie in der Gruppe, welche Getter und Setter Methoden sinnvoll sind. Welche Methoden darüber hinaus sollten die Klassen besitzen? Probieren Sie, einzelne Objekte zu erzeugen und Methoden zu testen.
3) Schreiben Sie ein Programm, welches notwendige Objekte erzeugt und verschiedene Use-Cases abbildet. Ein Sequenzdiagramm hilft, Use-Cases zu definieren und abzubilden. (https://de.wikipedia.org/wiki/Sequenzdiagramm).
4) Überlegen Sie sich den Sachverhalt zu erweitern. Hier könnte man auch eine Bankaufsicht (BAFIN, eine Behörde) hinzufügen, die Banken überprüfen/beaufsichtigen. Eine Bank muss, um geschäftsfähig zu sein, Bedingungen erfüllen. Dies gilt es von der BAFIN zu prüfen. Recherchieren Sie, was eine BAFIN beispielsweise prüft. Die BAFIN gibt es nur einmal (-> Singleton). Auch wären noch explizit Kreditkartenunternehmen modellierbar. Eine weitere Idee wäre, dass Privatpersonen/Geschäftskunden mit der Kreditkarte einkaufen. Eine Kreditkarte kann auch mal verloren gehen. Lassen Sie ein wenig der Phantasie freien Lauf, aber bleiben Sie fokussiert – also nicht zu viel auf einmal machen.
5) Bevor Sie die BAFIN und andere Use-Cases implementieren, erweitern Sie das zugehörige UML Diagramm und erstellen Sie zugehörige Sequenzdiagramme.
6) Implementieren Sie die Erweiterungen.

Bank – Use Case:

1. Kontentypen:

   Der Bankkunde hat ein Girokonto bei der Bank. Neben dem Girokonto kann er auch ein Tagesgeldkonto beantragen. Der Bankkunde kann das Tagesgeldkonto beantragen, wenn er ein Guthaben von 250 Euro auf dem Girokonto hat. Das Tagesgeldkonto wird monatlich mit 2,5 Prozent verzinst. Der Zinssatz wird durch das monatlich niedrigste Guthaben verzinst am jeweils ersten Tag des Folgemonats. Es kann laufend Geld zwischen dem Girokonto und dem
   Tagesgeldkonto übertragen werden. Entsprechend der Übertragungsrichtung wird das Geld dem jeweiligen Konto gutgeschrieben bzw. Abgezogen.

   Weiter kann sich der Bankkunde auch für ein Festgeldkonto entscheiden. Das Festgeldkonto wird jährlich mit 5 Prozent verzinst. Das Guthaben, dass auf ein Festgeldkonto eingezahlt wird, darf für 2 Jahre nicht vom Festgeldkonto abgehoben werden. (ggf. Kreditkartenkonto noch ergänzen)

Kreditkartenkonto wurde bei unserem Projekt ergänzt.

Es akkumuliert die Auszahlungen über die Kreditkarte. Am ersten Tag jedes Monats werden die Beträge vom Girokonto ausgeglichen. Auszahlungen werden nur solange zugelassen, wie sie unter 500 Euro sind und das Guthaben nicht überschritten wird, d.h. eine Gegenrechnung zw. Girokonto, Auszahlbetrag und Kreditkartenkonto ist bei jeder Kontenbewegung vonnöten.

Beteiligung:

- Bank
- Kunde
- Konten

Kontenarten:

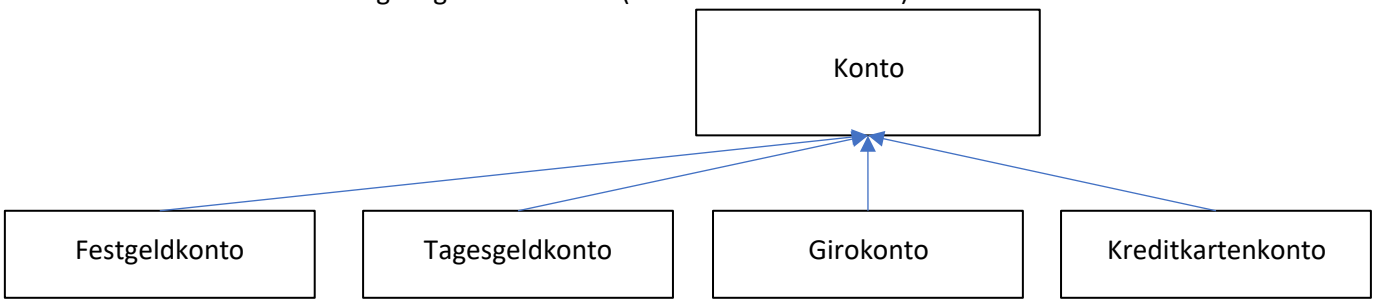
Girokonto (Keine Bedingungen, kann laufend Geld mit Tagesgeldkonto austauschen)

Tagesgeldkonto (Bedingung + 250 Euro auf Girokonto , kann laufend Geld mit Girokonto austauschen, monatliche Zinsen 2,5 %)

Festgeldkonto (Keine Bedingungen, Eingezahlte Beträge festverzinst 2 Jahre, jährliche Zinsen 5 %)

Kreditkartenkonto (Bedingung 500 Euro auf Girokonto, wird mit Girokonto abgeglichen.) // Keine Gebühren, um es nicht komplizierter zu machen.

Hier kann die Vererbung eingebaut werden (siehe Klassenübersicht):



2. Verifizierung:

Der Bankkunde hebt Geld am Bankautomaten ab oder zahlt Geld ein. Beim Abheben muss geprüft werden, ob der Bankkunde noch über ein Guthaben auf dem Girokonto verfügt. Das Geld kann mit einer Kreditkarte oder mit einer EC-Karte abgehoben werden. Beim Abheben wird über eine PIN-Abfrage geprüft, ob die Abhebung berechtigt ist. Beim Einzahlen wird dem Bankkunden das Guthaben gutgeschrieben. (Variation: auf das jeweilige Konto der genutzten Karte)

Beteiligung:

- Bank
- Kunde
- Bankautomat
- Girokonto, Kreditkartenkonto

Um ein MVP zu bauen, schlage ich vor, einstweilen die EC- oder Girokarte wegzulassen.

3. Kartensperrung bei Verlust:

   Der Bankkunde verliert die Girokarte oder die Kreditkarte. Die verlorene Karte muss deaktiviert werden. Bei der Nutzung einer Karte muss geprüft werden, ob die Karte die einzige aktive Karte (Giro bzw. Kreditkarte) des Kunden ist.

   Hier kann das Überladen von Operatoren eingebaut werden: z.B.: “==” für den Abgleich von Objekten

   Beteiligung:

   - Bank
   - Kunde
   - Kreditkarte, Girokarte

4. Überweisung:

   Kunde A überweist Kunden B XXX Euro Geld. Für die Überweisung muss das Konto von Kunde A gedeckt sein. Dem Kunde A muss die Kontonummer von Kunden B bekannt sein. Die Überweisungsfreigabe erfolgt über die PIN-Abfrage. Das überwiesene Geld wird Kunde A vom Guthaben abgezogen und Kunden B gutgeschrieben.

   Beteiligung:

   - Bank
   - Mehrere Kunden