Object Calisthenics

Refactoring in practice

Starting point

```
public class ProcessorFactory {
   private ProcessorFactory() {
   }

   public static ProduktProcessor getProcessor(final String produktNr) {
      ProduktProcessor processor = getKVProcessor(produktNr);
      if (processor != null) {
         return processor;
      }
      processor = getFRVProcessor(produktNr);
      if (processor != null) {
         return processor;
      }
      throw new NotImplementedException("Für Produkt mit Nummer: " + produktNr + " existiert noch kein Processor");

   }

   private static ProduktProcessor getKVProcessor(final String produktNr) {
      FachlicherSchluesselVerkaufsproduktKV fachlSchluessel = FachlicherSchluesselVerkaufsproduktKV.fromValue(produktNr);
      if (fachlSchluessel == null) {
         return null;
      }
      switch (fachlSchluessel) {
         case KUR_PFLEGE:
            return new KurPflegeProcessor();
         case SPITALTAGGELD:
            return new SpitaltaggeldProcessor();
         case SPITAL_MYFLEX:
            return new SpitalMyFlexProcessor();
         case AMBULANT_MYFLEX:
            return new AmbulantMyFlexProcessor();
         case AMBULANT_SANAGATE:
            return new AmbulantSanagateProcessor();
```

- A huge switch -> a lot of hidden elses
- Static method getProcessor()
- 200 lines of code
- Used abbrevations

Step 1: Replace switch with map

```
public class ProcessorFactory {
   private static final Map<FachlicherSchluesselVerkaufsproduktKV, ProduktProcessor> produktKVProcessorMap = new HashMap<>();
   private static final Map<FachlicherSchluesselVerkaufsproduktFRV.AVB2016, ProduktProcessor> produktFRVProcessorMap = new HashMap<>();

   public ProcessorFactory() {
      setupKVMap();
      setupFRVMap();
   }

   public ProduktProcessor getProcessor(final String produktNr) {
      ProduktProcessor processor = getKVProcessor(produktNr);
      if (processor != null) {
         return processor;
      }
      processor = getFRVProcessor(produktNr);
      if (processor != null) {
         return processor;
      }
      throw new NotImplementedException("Für Produkt mit Nummer: " + produktNr + " existiert noch kein Processor");

   }

   private ProduktProcessor getKVProcessor(final String produktNr) {
      FachlicherSchluesselVerkaufsproduktKV fachlSchluessel = FachlicherSchluesselVerkaufsproduktKV.fromValue(produktNr);
      if (fachlSchluessel == null) {
         return null;
      }
      return produktKVProcessorMap.get(fachlSchluessel);
   }

   private ProduktProcessor getFRVProcessor(final String produktNr) {
      FachlicherSchluesselVerkaufsproduktFRV.AVB2016 fachlSchluessel = FachlicherSchluesselVerkaufsproduktFRV.AVB2016.fromValue(produktN
      if (fachlSchluessel == null) {
         return null;
      }
      return produktFRVProcessorMap.get(fachlSchluessel);
   }

   private void setupKVMap() {
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.KUR_PFLEGE, new KurPflegeProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITALTAGGELD, new SpitaltaggeldProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITAL_MYFLEX, new SpitalMyFlexProcessor());
```

- Two maps as instance variables
- Replaced static method by regular method
- Still 200 lines of code
- Used abbrevations

Step 2: Remove superfluous code

```java
public class ProcessorFactory {
   private static final Map<FachlicherSchluesselVerkaufsproduktKV, ProduktProcessor> produktKVProcessorMap = new HashMap<>();
   private static final Map<FachlicherSchluesselVerkaufsproduktFRV.AVB2016, ProduktProcessor> produktFRVProcessorMap = new HashMap<>();

   public ProcessorFactory() {
      setupKVMap();
      setupFRVMap();
   }

   public ProduktProcessor getProcessor(final String produktNr) {
      ProduktProcessor processor = getKVProcessor(produktNr);
      if (processor != null) {
         return processor;
      }
      processor = getFRVProcessor(produktNr);
      if (processor != null) {
         return processor;
      }
      throw new NotImplementedException("Für Produkt mit Nummer: " + produktNr + " existiert noch kein Processor");

   }

   private ProduktProcessor getKVProcessor(final String produktNr) {
      FachlicherSchluesselVerkaufsproduktKV fachlSchluessel = FachlicherSchluesselVerkaufsproduktKV.fromValue(produktNr);

      return produktKVProcessorMap.get(fachlSchluessel);
   }

   private ProduktProcessor getFRVProcessor(final String produktNr) {
      FachlicherSchluesselVerkaufsproduktFRV.AVB2016 fachlSchluessel = FachlicherSchluesselVerkaufsproduktFRV.AVB2016.fromValue(produktNr)

      return produktFRVProcessorMap.get(fachlSchluessel);
   }

   private void setupKVMap() {
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.KUR_PFLEGE, new KurPflegeProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITALTAGGELD, new SpitaltaggeldProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITAL_MYFLEX, new SpitalMyFlexProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.AMBULANT_MYFLEX, new AmbulantMyFlexProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.AMBULANT_SANAGATE, new AmbulantSanagateProcessor());
      produktKVProcessorMap.put(FachlicherSchluesselVerkaufsproduktKV.ALTERNATIV_MYFLEX, new AlternativMyFlexProcessor());
```

- querying the map with a null value returns null -> ok

Step 3a: Extract collections in own class

```
public class KVProcessorFactory {
    private static final Map<FachlicherSchluesselVerkaufsproduktKV, ProduktProcessor> processorMap = new HashMap<>();

    public KVProcessorFactory() {
        setup();
    }

    public ProduktProcessor lookup(FachlicherSchluesselVerkaufsproduktKV fachlSchluessel) {
        return processorMap.get(fachlSchluessel);
    }

    private void setup() {
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.KUR_PFLEGE, new KurPflegeProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITALTAGGELD, new SpitaltaggeldProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITAL_MYFLEX, new SpitalMyFlexProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.AMBULANT_MYFLEX, new AmbulantMyFlexProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.AMBULANT_SANAGATE, new AmbulantSanagateProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.ALTERNATIV_MYFLEX, new AlternativMyFlexProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.SPITAL_SANAGATE, new SpitalSanagateProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.ALTERNATIV_SANAGATE, new AlternativSanagateProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.ALTERNATIV01, new Alternativ01Processor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.CSS_STANDARD, new StandardversicherungProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.NOTFALL, new NotfallversicherungProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.ZAHNPFLEGE, new ZahnpflegeProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.ZAHNPFLEGE_SANAGATE, new ZahnpflegeSanagateProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktKV.DENTA, new DentaProcessor());
```

Step 3b: same for FRV map

```
public class FRVProcessorFactory {
    private static final Map<FachlicherSchluesselVerkaufsproduktFRV.AVB2016, ProduktProcessor> processorMap = new HashMap<>();

    public FRVProcessorFactory() {
        setupFRVMap();
    }

    public ProduktProcessor lookup(FachlicherSchluesselVerkaufsproduktFRV.AVB2016 fachlSchluessel) {
        return processorMap.get(fachlSchluessel);
    }

    private void setupFRVMap() {
        processorMap.put(FachlicherSchluesselVerkaufsproduktFRV.AVB2016.HEILUNGSKOSTEN, new FrvProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktFRV.AVB2016.PERSONENASSISTANCE, new FrvProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktFRV.AVB2016.AUSLANDRECHTSSCHUTZ, new FrvProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktFRV.AVB2016.REISEGEPAECK, new FrvProcessor());
        processorMap.put(FachlicherSchluesselVerkaufsproduktFRV.AVB2016.ANNULLIERUNGSKOSTEN, new FrvProcessor());
    }
}
```

Step 4: Replace abbrevations

```
public class ProcessorFactory {
   KVProcessorFactory kvProcessorFactory = new KVProcessorFactory();
   FRVProcessorFactory frvProcessorFactory = new FRVProcessorFactory();

   public ProduktProcessor getProcessor(final String produktNummer) {
      ProduktProcessor processor = getKVProcessor(produktNummer);
      if (processor != null) {
         return processor;
      }
      processor = getFRVProcessor(produktNummer);
      if (processor != null) {
         return processor;
      }
      throw new NotImplementedException("Für Produkt mit Nummer: " + produktNummer + " existiert noch kein Processor");
   }

   private ProduktProcessor getKVProcessor(final String produktNummer) {
      FachlicherSchluesselVerkaufsproduktKV fachlicherSchluessel = FachlicherSchluesselVerkaufsproduktKV.fromValue(produktNummer);

      return kvProcessorFactory.lookup(fachlicherSchluessel);
   }

   private ProduktProcessor getFRVProcessor(final String produktNummer) {
      FachlicherSchluesselVerkaufsproduktFRV.AVB2016 fachlicherSchluessel =
            FachlicherSchluesselVerkaufsproduktFRV.AVB2016.fromValue(produktNummer);

      return frvProcessorFactory.lookup(fachlicherSchluessel);
   }
}
```

- Class is now less then 40 lines
- Abbrevations eliminated as usefull

Step 5: Use Set instead of Map

```
import static ch.css.produkt.key.utils.impl.enumeration.FachlicherSchluesselVerkaufsproduktFRV.AVB2016.*;

public class FRVProcessorFactory {
    private static final Set<FachlicherSchluesselVerkaufsproduktFRV.AVB2016> supportedFRVs = new HashSet<>();

    public FRVProcessorFactory() {
        setup();
    }

    public ProduktProcessor lookup(FachlicherSchluesselVerkaufsproduktFRV.AVB2016 fachlicherSchluessel) {
        if (supportedFRVs.contains(fachlicherSchluessel)) {
            return new FrvProcessor();
        }
        return null;
    }

    private void setup() {
        supportedFRVs.addAll(Arrays.asList(
                HEILUNGSKOSTEN,
                PERSONENASSISTANCE,
                AUSLANDRECHTSSCHUTZ,
                REISEGEPAECK,
                ANNULLIERUNGSKOSTEN
        ));
    }
}
```

- Map not needed only one return value
- Use static import to make it more readable

Open issues

- A lot of instances are created for the map
  - Possible solution with Classes: instead of `Map<FachlicherSchluesselVerkaufsproduktKV, ProduktProcessor>` use `Map<FachlicherSchluesselVerkaufsproduktKV, Class<? extends ProduktProcessor>>` and return `newInstance()` of Class
- Wrap produktNummer?
  - Own combined enum?
- Replacement of static member getProcessor correct?