Faktor-IPS Übungsunterlagen

(Dokumentversion 1296)

Inhaltsverzeichnis

Übungen zu Kapitel: Hello World

Legen Sie ein neues Java-Projekt mit dem Namen „Grundmodell“ an.

Compliance Level des Java Compilers auf 1.4 setzen
(In den Preferences unter: Java→Compiler: Compiler Compliance Level).

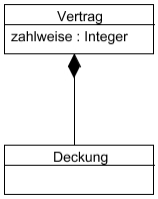
Fügen Sie die FaktorIPS-Nature zu dem Projekt hinzu mit

- Sourceverzeichnis = modell
- Base Package Name = de.qv.produkt.modell
- Runtime ID Prefix = base.

Legen Sie ein Package „base“ an.

Legen sie die Klasse „base.Vertrag“ an.

Übungen zu Kapitel: Arbeiten mit Modell und Sourcecode



Definieren Sie ein Attribut Zahlweise (Datentyp Integer) am Vertrag.

Probieren Sie die Wirkungsweise der Annotations @generated, @generated NOT aus.

Definieren Sie den Wertebereich für das Attribut Zahlweise: 1, 2, 4, 12

Legen Sie die Klasse „base.Deckung“ an.

Definieren Sie die Composite-Beziehung zwischen Vertrag und Deckung.

Übungen zu Kapitel: Modellierung des Hausratvertrags

Übung 1

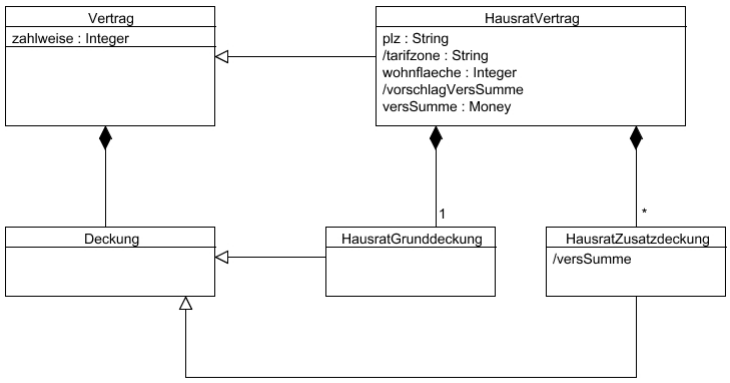
Legen Sie ein neues Java-Projekt "Hausratmodell" an und fügen die IPS-Nature hinzu.

Fügen sie dem Buildpath des Java-Projektes die Referenz auf das Projekt Grundmodell hinzu.

Fügen sie in der Datei ".ipsproject" dem IpsObjectPath die Referenz auf das Projekt Grundmodell hinzu.

```
<IpsObjectPath ...>
  <Entry .../>
  <Entry type="project" referencedIpsProject="Grundmodell"/>
</IpsObjectPath ...>
```

Erfassen sie die Klassen und Attribute inklusive deren Wertebereiche gemäss dem Klassendiagramm und der folgenden tabellarischen Beschreibung.



!!!Bitte noch NICHT die Beziehungen erfassen !!!

Name : Datentyp	*Beschreibung, Bemerkung*
plz : String	Postleitzahl des versicherten Hausrats
/tarifzone : String	Die Tarifzone ergibt sich aus der Postleitzahl und ist maßgeblich für den zu zahlenden Beitrag. => Achten sie also bei der Eingabe darauf den AttributeType auf derived (computation on each method call) zu setzen!
wohnflaeche : Integer	Die Wohnfläche des versicherten Hausrats in Quadratmetern. Der erlaubte Wertebereich ist min=0 und max=unbeschränkt. Hierzu das Feld max leer lassen.
/vorschlagVersSumme : Money	Vorschlag für die Versicherungssumme. Wird auf Basis der Wohnfläche bestimmt. => Achten sie bei der Eingabe darauf den AttributeType auf derived (computation on each method call) zu setzen!

Name : Datentyp	*Beschreibung, Bemerkung*
versSumme : Money	Die Versicherungssumme. Der erlaubte Wertebereich ist min=0 EUR und max=unbeschränkt.

Übung 2

Implementieren sie die Ermittlung des Vorschlagswertes für die Versicherungssumme. Der Vorschlag ergibt sich durch `wohnfläche * 650 Euro`. (Später werden wir die 650 Euro produktseitig konfigurierbar machen.)

Übung 3

Definieren Sie die Beziehung zwischen Vertrag-Deckung als Container-Beziehung.

Legen Sie die fehlenden Beziehungen im Hausratmodell an.

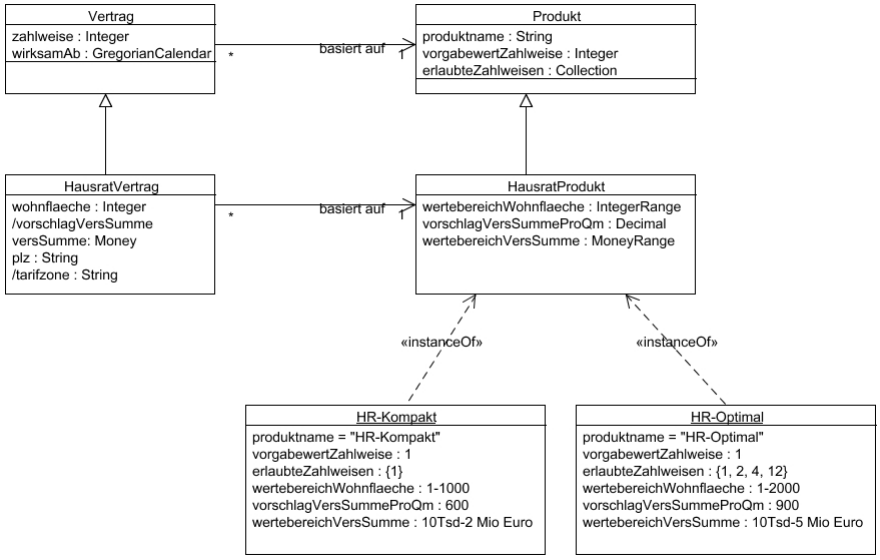
Zusatzübung

Implementieren sie einen JUnit Testfall für die Ermittlung des Vorschlagswertes für die Versicherungssumme.

Modell zum Kapitel Aufnahme von Produktaspekten ins Modell

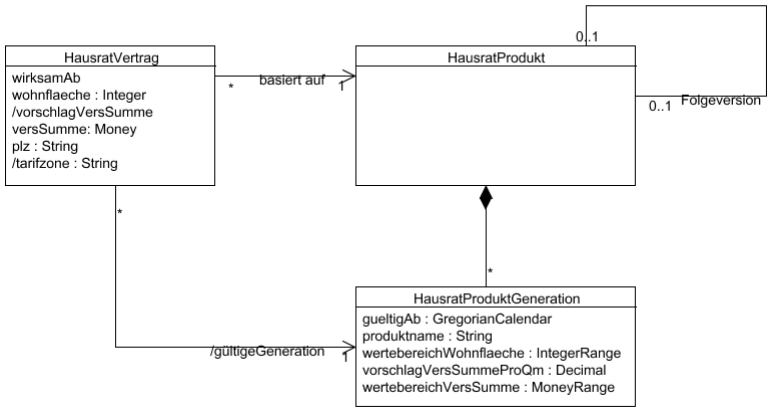
Hausratvertrag und Hausratprodukt

Ohne Berücksichtigung von zeitlichen Änderungen



Nach Berücksichtigung von zeitlichen Änderungen

(Der Begriff Generation muss durch Anpassungsstufe ersetzt werden. Die Ableitungen von den Basisklassen sind weggelassen)



Übungen zum Kapitel: Aufnahme von Produktaskpekten ins Modell

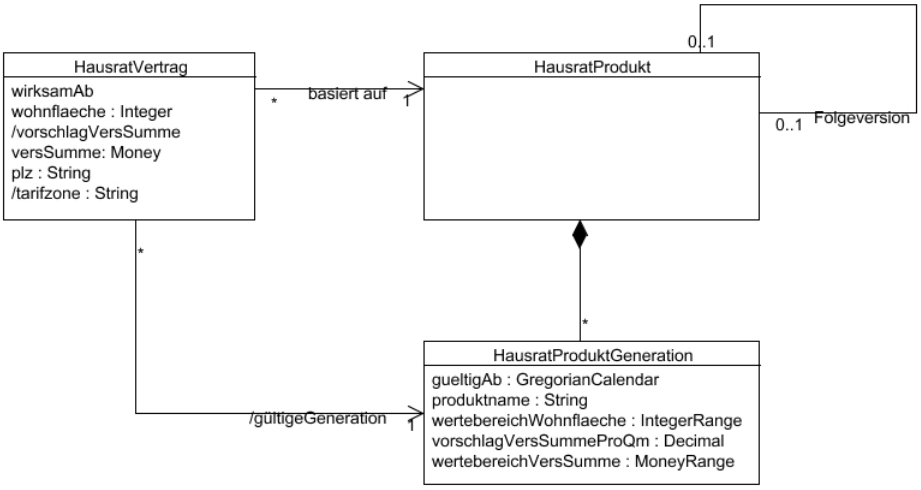
Übung 1

Stellen Sie in der ".ipsproject" Datei die Namensgebung für die Änderungen im Zeitablauf von VAA auf PM.

```
<GeneratedSourcecode changesInTimeNamingConvention="PM" docLanguage="de_DE"/>
```

Fügen Sie das Attribut "wirksamAb" vom Typ GregorianCalendar zum Vertrag hinzu und implementieren sie die getEffectiveFromAsCalendar() Methode am Vertrag.

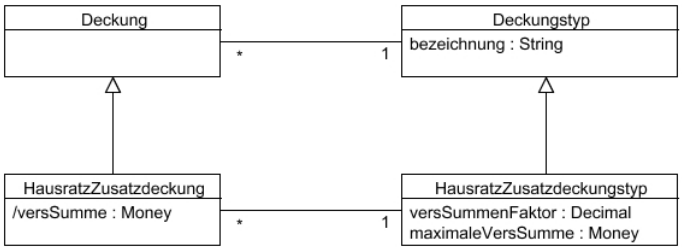
Definieren Sie das konstante, produkt-relevante Attribut VorschlagVersSummeProQm und implementieren Sie die Berechnung des Vorschlags für die Versicherungssumme in der Klasse HausratVertrag. (Lösung im Sourcecode-Ausschnitt 1 im Anhang)



Übung 2

Definieren sie das konstante, produkt-relevante Attribut "bezeichnung" an der Deckung.

Die Versicherungssumme der Hausratzusatzdeckung ergibt sich aus der Versicherungssumme des Vertrags durch Multiplikation mit einem produktseitig festgelegten Faktor (versSummenFaktor) und ist durch die maximaleVersSumme gedeckelt. Definieren Sie die beiden zusätzlichen Attribute "versSummenFaktor" und "maximaleVersSumme" und implementieren sie die Berechnung der Versicherungssumme der HausratZusatzdeckung. (Lösung im Sourcecode-Ausschnitt 2 im Anhang)



Übung 3

Definieren Sie die produktseitigen Beziehungen zwischen Produkt und Deckungstyp sowie zwischen HausratProdukt und den hausratspezifischen Deckungstypen.

Zusatzübung

Passen die den JUnit Testfall für die Ermittlung des Vorschlagswertes für die Versicherungssumme an. Im Sourcecode-Ausschnitt 3 befindet sich der notwendige setUp() Code, um entsprechende Produktdaten zu erzeugen.

Übungen zum Kapitel: Definition der Produkte

Legen Sie ein neues Java-Projekt "Hauratprodukte" an und fügen sie die FaktorIPS Nature hinzu.

Erweitern Sie den Java Buildpath des Projektes um Referenzen auf die beiden anderen Projekte.

Erweitern Sie den IpsObjectPath um eine Referenz auf das Hausratmodell-Projekt. (In der ".ipsproject" Datei.

```
<IpsObjectPath ...>
  <Entry .../>
  <Entry type="project" referencedIpsProject="Hausratmodell"/>
</IpsObjectPath ...>
```

Setzen Sie das Wirksamkeitsdatum für Änderungen auf den nächsten Quartalsbeginn.

Erzeugen Sie ein Package "hausrat.produkte" und ein Package "hausrat.deckungstypen"

Definieren Sie das Produkt HR-Optimal

Konfigurationsmöglichkeit	***HR-Optimal***
Produktname	Hausrat Optimal
Vorschlag Versicherungsumme pro qm Wohnfläche	900EUR
Vorgabewert Zahlweise	1 (jährlich)
Erlaubte Zahlweisen	1, 2, 4, 12
Vorgabewert Wohnflaeche	<null>
Erlaubte Wohnflaeche	0-2000
Vorgabewert Versicherungssumme	<null>
Versicherungssumme	10000EUR – 5000000EUR

Legen sie einen Baustein HRD-Grunddeckung-Optimal basierend auf der Vorlage "HausratGrunddeckungstyp" im Ordner "Deckungstypen" an.

Legen Sie zwei Zusatzdeckungen „HRD-Fahrraddiebstahl“ und „HRD-Überspannung“ basierend auf der Vorlage „HausratZusatzdeckungstyp“ im Ordner „Deckungstypen“ ab. Bei der Fahrraddiebstahlsdeckung soll die Versicherungssumme auf 1% der im Vertrag vereinbarten Versicherung und maximal 3000 Euro begrenzt sein. Bei der Überspannungsdeckung sollen es 5% ohne Deckelung sein. (5% werden als 0.05 eingegeben).

Ordnen sie die Deckungstypen dem Produkt HR-Optimal zu.

Definieren Sie das Produkt HR-Kompakt mit den folgenden Eigenschaften

Konfigurationsmöglichkeit	***HR-Kompakt***
Produktname	Hausrat Kompakt
Vorschlag Versicherungsumme pro qm Wohnfläche	600EUR
Vorgabewert Zahlweise	jährlich
Erlaubte Zahlweisen	halbjährlich, jährlich
Vorgabewert Wohnflaeche	<null>
Erlaubte Wohnflaeche	0-1000 qm
Vorgabewert Versicherungssumme	<null>
Versicherungssumme	10000EUR – 2000000EUR

Legen sie eine Grunddeckung für das Produkt an "HRD-Grunddeckung-Kompakt". Ordnen sie die Grunddeckung und die Zusatzdeckungen dem Produkt HR-Kompakt zu. Die Zusatzdeckungstypen sollen in dem Kompaktprodukt optional hinzugewählt werden können.

Anhang: Sourcecode Ausschnitte

In den Sourcecodebeispiele ist die VAA-Namensgebung für Produktänderungen im Zeitablauf verwendet.

Ausschnitt 1: Ermittlung Vorschlag Versicherungssumme

Klasse: Hausratvertrag

```
public Money getVorschlagVersSumme() {
      IHausratProduktGen gen = getHausratProduktGen();
      if (gen==null) {
            return Money.NULL;
      }
      return gen.getVorschlagVersSummeProQm().multiply(wohnflaeche);
}
```

Ausschnitt 2: Ermittlung Versicherungssumme Zusatzdeckung

Klasse: Hausratzusatzdeckung

```
public Money getVersSumme() {
    IHausratZusatzdeckungstypGen gen = getHausratZusatzdeckungstypGen();
    if (gen==null) {
        return Money.NULL;
    }
    Decimal faktor = gen.getVersSummenFaktor();
    Money vsVertrag = getHausratVertrag().getVersSumme();
    Money vs = vsVertrag.multiply(faktor, BigDecimal.ROUND_HALF_UP);
    if (vs.isNull()) {
        return vs;
    }
    Money maxVs = gen.getMaximaleVersSumme();
    if (maxVs.greaterThan(vs)) {
        return maxVs;
    }
    return vs;
}
```

Ausschnitt 3: Setup Code fuer den JUnit Testfall fuer die Ermittlung der Versicherungssumme

```
private InMemoryRuntimeRepository repository;
private HausratProdukt produkt;
private HausratProduktGen gen;

protected void setUp() {
   repository = new InMemoryRuntimeRepository();
   produkt = new HausratProdukt(repository, "HausratProdukt-2007",
            "HausratProdukt", "2007");
   gen = new HausratProduktGen(produkt);
   gen.setValidFrom(new DateTime(2007, 1, 1));
   repository.putProductCmptGeneration(gen);
}
```

Ausschnitt 4: Ermittlung Tarifzone

Klasse: Hausratvertrag

```
public String getTarifzone() {
    if (plz==null) {
        return null;
    }
    IRuntimeRepository repository = getHausratProdukt().getRepository();
    Tarifzonentabelle tabelle = Tarifzonentabelle.getInstance(repository);
    TarifzonentabelleRow row = tabelle.findRow(plz);
    if (row==null) {
        return "I";
    }
    return row.getTarifzone();
}
```