Faktor-IPS

Übungsunterlagen zur Schulung

(Dokumentversion 1404)

Übungen zu Kapitel 1: Projekt einrichten, erste Klasse

Übung 1

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

Fügen Sie die Faktor-IPS Nature zu dem Projekt hinzu mit den folgenden Eigenschaften:

- Project type = Model project
- Sourceverzeichnis = modell
- Base Package Name = org.faktorips.schulung.modell
- Runtime ID Prefix = hausrat.

Übung 2

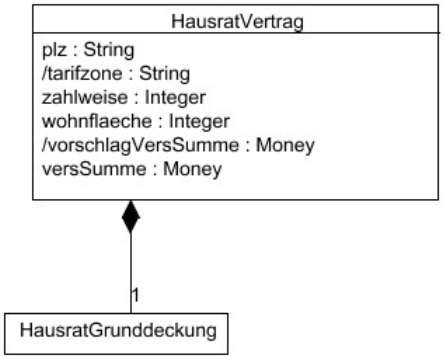
Legen Sie im Sourceverzeichns „modell“ das Faktor-IPS Package „hausrat“ an.

Legen sie die Klasse „HausratVertrag“ in dem Package „hausrat“ an.

Sehen Sie sich den generierten Sourcecode für das published Interface und die Klasse an.

Schauen Sie sich das Projekt im Modellexplorer an.

Beispielmodell



Übungen zu Kapitel 2.1: Attribute auf Vertragsseite

Übung 1

Legen Sie die Attribute der Klasse HausratVertrag an.

Name : Datentyp	***Beschreibung, Bemerkung***
zahlweise : Integer	Die Zahlweise des Vertrages. Die erlaubten Werte sind 1, 2, 4 und 12.
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 in das Feld max <null> eintragen 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!
versSumme : Money	Die Versicherungssumme. Der erlaubte Wertebereich ist min=0 EUR und max=<null>.

Übung 2

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

Übung 3

Implementieren Sie die Ermittlung der Tarifzone. Geben Sie immer Tarifzone „I“ zurück. (Später werden wir die Tarifzone anhand der Postleitzahl aus einer Tarifzonentabelle ermitteln.)

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.)

Legen sie eine Junit Testklasse HausratTest an. Schreiben sie Testmethoden für die Methoden getTarifzone() und getVorschlagVersSumme().

Übungen zu Kapitel 2-2: Beziehungen auf Vertragsseite

Übung

Legen Sie die Klasse „hausrat.HausratGrunddeckung“ an.

Legen Sie die Composite-Beziehung zwischen HausratVertrag und HausratGrunddeckung mit der Multiplizität [0..1] an.

Übungen zu Kapitel 3-1: Attribute auf Produktseite

Übung 1

Legen Sie ein neues Java-Projekt "Hausratprodukte" an und fügen Sie die IPS-Nature mit den folgenden Eigenschaften hinzu:

- Project type = Product definition project
- Sourceverzeichnis = produktdaten
- Base Package Name = org.faktorips.schulung.produktdaten
- Runtime ID Prefix = hausrat.

Stellen sie mit Hilfe des IPS-Projekteigenschaft-Dialogs eine Abhängigkeit des Projektes zum Hausratmodell Projekt her:

Kontextmenü im Package Explorer -> Eigenschaften -> IPS Build Path -> Reiter „Projekte“

Mit dem Hinzufügen-Button das Hausratmodell hinzufügen

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

Übung 2

Legen Sie die Klasse „HausratProdukt“ an und verknüpfen Sie „HausratVertrag“ mit „HausratProdukt“.

Definieren Sie das Attribute „produktname“

Legen Sie im Projekt „ Hausratprodukte“ im Sourceverzeichnis „produktdaten“ ein neues Package „produkte“ an.

Setzen Sie das aktuelle Wirksamkeitsdatum auf den nächsten Quartalsbeginn.

Legen Sie die beiden Produkte HR-Kompakt und HR-Optimal an und geben Sie im Editor den Produktnamen ein.

Übung 3

Definieren Sie an der Klasse „HausratProdukt“ ein neues Attribut „vertriebsname“.

Geben Sie für die beiden Produkte einen Vertriebsnamen an.

Übungen zu Kapitel 3-2: Änderungen im Zeitablauf

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

Definieren Sie das das Attribut „vorschlagVersSummeProQm“ am HausratratProdukt und implementieren Sie die Berechnung des Vorschlags für die Versicherungssumme in der Klasse HausratVertrag unter Verwendung dieses Attributes. (Lösung im Sourcecode-Ausschnitt 1 im Anhang)

Legen sie einen neuen JUnit Testfall HausratTestMitProdukt an. Diese zweite Testfallklasse ist

notwendig, weil im Folgenden im Testfall gegen Produktdaten getestet wird. Da es eine Abhängigkeit des Produktdatenprojektes zum Hausratprojekt gibt, sind dem Produktdatenprojekt die Klassen und Resourcen des Hausratmodellprojektes bekannt, umgekehrt jedoch nicht. Später werden wir eine Alternative kennenlernen.

Zum Zugriff auf das Repository verwenden Sie folgenden Setup-Code.

```
private IRuntimeRepository repository;

public void setUp() {
  // Repository erzeugen
  repository= ClassloaderRuntimeRepository.create(
  "org/faktorips/schulung/produktdaten/internal/faktorips-repository-toc.xml");
}
```

Lesen sie eines der Produkte aus dem Repository aus und verwenden sie dieses im Testfall der Methode getVorschlagVersSumme().

Übungen zu Kapitel 3-3: Konfigurierbare Vertragsattribute

Markieren Sie die folgenden Attribute der Klasse „HausratVertrag“ als konfigurierbar:

- zahlweise
- wohnflaeche
- versSumme

Geben Sie die Daten für die beiden Produkte gemäß der folgenden Tabelle ein:

Konfigurationsmöglichkeit	***HR-Kompakt***	***HR-Optimal***
Defaultwert Zahlweise	jährlich	jährlich
Erlaubte Zahlweisen	halbjährlich, jährlich	monatlich, vierteljährlich, halbjährlich, jährlich
Defaultwert Wohnfläche	<null>	<null>
Erlaubter Bereich Wohnfläche	0-1000 qm	0-2000 qm
Defaultwert VersSumme	<null>	<null>
Erlaubter Bereich VersSumme	10Tsd – 2Mio Euro	10Tsd – 5Mio Euro

Übungen zu Kapitel 3-4: Beziehungen auf Produktseite

Legen Sie die Produktklasse „HausratGrunddeckungstyp“ an.

Legen Sie die Beziehungen zwischen „HausratProdukt“ und „HausratGrunddeckungstyp“ an. (Jedes Hausratprodukt soll genau einen Grundeckungstypen enthalten).

Anlegen der Deckungsbausteine HRD-Optimal und HRD-Kompakt

Beziehung zwischen den Hausratprodukt- und den HausratDeckungsbausteinen herstellen

Übungen zum Kapitel 3-5: Zugriff auf Produktdaten zur Laufzeit

Übung 1

Geben sie die folgenden Informationen über das Kompakt-Produkt (in einer test() Methode) auf der Konsole aus:

- Produktname
- Vorschlag für einen Quadratmeter Wohnfläche
- Defaultwert für die Zahlweise
- Erlaubte Zahlweisen
- Erlaubter Wertebereich für die Versicherungssumme
- Erlaubter Wertebereich für die Wohnfläche

Das Ausführen des Testfalls sollte folgendes Ergebnis liefern:

```
Produktname: Hausrat Kompakt
Vorschlag Vs pro 1qm: 600.00 EUR
Default Zahlweise  : 1
Erlaubte Zahlweisen: [1, 2]
Bereich Vs: 10000.00 EUR-2000000.00 EUR
Bereich Wohnflaeche: 0-1000
```

Übungen zu Kapitel 4.1: Grundlagen der Verwendung von Tabellen

Übung 1

Legen Sie die Tabellenstruktur „Tarifzonentabelle“ im Projekt „Hausratmodell“ an. Definieren Sie einen Bereich für plzVon, plzBis und legen Sie den UniqueKey auf diesen Bereich an.

Plz-Von	*Plz-bis*	*Tarifzone*
17235	17237	II
45525	45549	III
59174	59199	IV
47051	47279	V
63065	63075	VI

Legen Sie den zugehörigen Tabelleninhalt „Tarifzonentabelle“ im Projekt „Hausratprodukte“ an.

Übung 2

Implementieren Sie die Methode HausratVertrag.getTarifzone() unter Verwendung der Tarifzonentabelle.

Testen Sie ihre Implementierung, indem Sie den JUnit Test um eine Methode testGetTarifzone() erweitern.

Übungen zu Kapitel 4.2: Zusammenhang Bausteine-Tabellen

Übung 1

Legen Sie die Tabellenstruktur „TariftabelleHausrat“ im Projekt „Hausratmodell“ an. Stellen Sie den Tabellentyp auf „Multiple Contents“.

Legen Sie im Projekt Hausratprodukte die Tariftabellen für die beiden Hausratprodukte an.

Tariftabelle-Optimal

Tarifzone	*Beitragssatz*
I	0.8
II	1.0
III	1.44
IV	1.70
V	2.00
VI	2.20

Tariftabelle-Kompakt

Tarifzone	*Beitragssatz*
I	0.6
II	0.8
III	1.21
IV	1.50
V	1.80
VI	2.00

Übung 2

Definieren Sie im Hausratmodell, dass die Klasse HausratGrundeckungstyp die TariftabelleHausrat verwendet (in der Rolle Tariftabelle).

Ordnen Sie den beiden Grundeckungstypen die jeweilige Tariftabelle zu.

Übung 3

Legen Sie an der Klasse HausratGrunddeckung ein abgeleitetes (cached) Attribut „jahresbasisbeitrag“ vom Typ Money an. Der Jahresbasisbeitrag ist der jährlich zu zahlende Beitrag ohne Zuschläge und Nachlässe, ohne Ratenzahlungszuschlag und ohne Versicherungssteuer.

Definieren Sie eine Methode `void berechneJahresbasisbeitrag()` in der Modellklasse Hausratgrundeckung (nicht direkt im Java-Code) zur Berechnung des Jahresbasisbeitrags.

Implementieren Sie die Methode. Verwenden Sie hierzu die im Baustein zugeordnete Tariftabelle.

Die Berechnungsvorschrift ist wie folgt:

- Ermittlung des Beitragsatzes pro 1000 Euro Versicherungssumme aus der Tariftabelle
- Division der Versicherungssumme durch 1000 Euro und Multiplikation mit dem Beitragssatz

Testen Sie ihre Implementierung, indem Sie den JUnit Test um eine Methode testBerechneJahresbasisbeitrag() erweitern.

Hinweis:

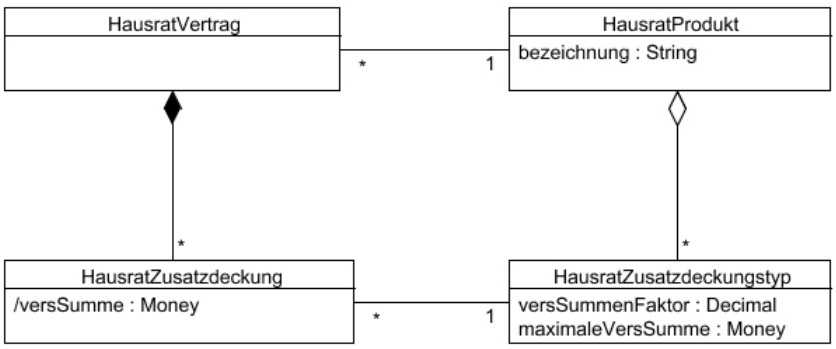
Sie müssen wie in den vorangegangenen Testfällen zuerst einen HausratVertrag erzeugen und dessen Attribute setzen. Zusätzlich müssen Sie nun eine HausratGrunddeckung erzeugen und zum Vertrag hinzufügen. Dies geht wie folgt.

```
// Grunddeckungstyp holen, der dem Produkt in der Anpassungsstufe zugeordnet ist.
IHausratGrunddeckungstyp deckungstyp = kompaktAnpStufe.getHausratGrunddeckungstyp();

// Grunddeckung erzeugen und zum Vertag hinzufuegen
IHausratGrunddeckung deckung = vertrag.newHausratGrunddeckung(deckungstyp);
```

Übungen zum Kapitel 5

Übung 1

Erweitern Sie das Hausratmodell um die Zusatzdeckung und den Zusatzdeckungstyp gemäß dem folgenden Diagramm:



Legen Sie die folgenden Zusatzdeckungen an:

	Fahrraddiebstahl 2009-01	**Überspannung 2009-01**
VersSummenFaktor	0.01	0.05
MaximaleVersSumme	5000EUR	<null>

Fügen sie die Deckungen dem HR-Optimal Produkt hinzu.

Übung 2

Implementieren Sie die Berechnung der Versicherungssumme der Zusatzdeckung.

Übung 3

Legen Sie das Attribut „jahresbasisbeitrag“ und eine Methode `berechneJahresbasisbeitrag()` zur Berechnung des Attributes an der Klasse Zusatzdeckung an.

Definieren Sie die Formelsignatur für die Berechnung des Jahresbasisbeitrags am Zusatzdeckungstyp. Definieren Sie nun den Nettobasisbeitrag für die Zusatzdeckungstypen in den Bausteinen wie folgt:

- Fahrraddiebstahl: 10% der Versicherungssumme der Deckung
- Überspannung: 10EUR + 3% der Versicherungssumme der Deckung

Übung 4

Testen Sie die Beitragsberechnung der Zusatzdeckung, indem Sie den JUnit Test um eine Methode testBerechneJahresbasisbeitrag() erweitern.

Übungen zum Kapitel 6

Übung 1

Legen Sie ein Wertebereichsprüfung für das Attribut wohnflaeche des Hausratvertrags an. Verwenden sie in dem Prüfungstext einen Parameter an den zur Laufzeit die Maximale Wohnflaeche zu übergeben ist z.B. {maxWohnflaeche}.

Schreiben sie einen Testfall der die validate()-Methode des HausratVertrages aufruft und die Prüfung fehlschlagen lässt. Überprüfen sie anhand des Message Codes ob die entsprechende Message in der MessageList der validate()-Methode enthalten ist. Prüfen sie ebenfalls den MsgReplacementParameter und dessen Wert.

Übungen zum Kapitel 7

Übung 1

Legen Sie eine abstrakte Klasse HausratDeckung an und geben Sie diese als Oberklasse von HausratGrunddeckung und HausratZusatzdeckung an. Legen Sie vom HausratVertrag eine Beziehung Deckungen zur HausratDeckung an und markieren Sie diese als derived union. Tragen Sie diese Beziehung dann bei den bestehenden Beziehungen als die Beziehung ein, deren Subset die bestehenden Beziehungen jetzt sind. Verfahren Sie analog für die abstrakte Produktklasse HausratDeckungsTyp etc.

Übung 2

Legen Sie das Merkmal /`jahresbasisbeitrag` und die Methode `berechneJahresbasisbeitrag`() an der HausratDeckung an. Implementieren Sie die Methode `berechneJahresbasisbeitrag` so, dass sie eine abstrakte Methode `berechneJahresbasisbeitragInternal` aufrufen, die Sie dann in HausratGrunddeckung und HausratZusatzdeckung mit den zuvor gemachten Implementierungen überschreiben.

Übung 3

Legen Sie das Merkmal `/jahresbasisbeitrag` und die Methode `berechneJahresbasisbeitrag`() am HausratVertrag und implementieren Sie diese Methode so, dass sie über alle HausratDeckungen iteriert und deren Basisbeitragsberechnung anstößt und aufaddiert.

Übungen zum Kapitel 8

Übung 1

- Erstellen einer Klasse TestContent im Package „hausrat“ des Hausratmodell Projektes.
- Anlegen eines Attributes vom Typ InMemoryRepository und getter-Methode.
- Bevor das Repository mit Inhalt gefüllt werden kann, ist es notwendig einige Klassen abzuleiten, damit sie für Testzwecke verwendet werden können. Dies sind
  - HausratGrunddeckungstypAnpStufe *(bzw. bei Verwendung der VAA-Namenskonvention ~Gen statt ~AnpStufe)*: Für diese Klasse muss eine Ableitung (TestHausratGrunddeckungstypAnpStufe) erzeugt werden, die um eine Methode *setTariftabellenName(String)* ergänzt wird, mit der das protected Attribut *tariftabellenName* gesetzt werden kann.
  - Tarifzonentabelle: In einer Ableitung dieser Klasse wird im Konstruktor das Attribut rows der Superklasse mit einer ArrayList initialisiert und diese anschließend mit Beispieldatensätzen also Instanzen der Klasse TarifzonentabelleRow gefüllt.
  - Tariftabelle: entsprechend wie Tarifzonentabelle

```
/**
 * Es ist eine Ableitung der Klasse HausratGrunddeckungstypAnpStufe
 * notwendig, da auf das protected Atribut <i>tariftabellenname</i>
 * zugegriffen werden muss.
 */
private static class TestHausratGrunddeckungstypAnpStufe extends
          HausratGrunddeckungstypAnpStufe {

          public TestHausratGrunddeckungstypAnpStufe(HausratGrunddeckungstyp productCmpt) {
                    super(productCmpt);
          }

          public void setTariftabellenName(String name) {
                    this.tariftabelleName = name;
          }
}

/**
 * Die Ableitung der Tarifzonentabelle ist notwendig um sie mit
 * Beispielinhalt zu füllen.
 */
private static class TestTarifzonentabelle extends Tarifzonentabelle {

          public TestTarifzonentabelle() {
                    super();
                    rows = new ArrayList<TarifzonentabelleRow>();
                    rows.add(new TarifzonentabelleRow("10000", "19999", "I"));
                    rows.add(new TarifzonentabelleRow("20000", "29999", "II"));
                    rows.add(new TarifzonentabelleRow("30000", "39999", "III"));
                    initKeyMaps();
          }
}

/**
 * Die Ableitung der Tarifzonentabelle ist notwendig um sie mit
 * Beispielinhalt zu füllen.
 */
private static class TestTariftabelle extends Tariftabelle {

          public TestTariftabelle() {
                    super();
                    rows = new ArrayList<TariftabelleRow>();
                    rows.add(new TariftabelleRow("I", Decimal.valueOf("0.5")));
                    rows.add(new TariftabelleRow("II", Decimal.valueOf("0.7")));
                    rows.add(new TariftabelleRow("III", Decimal.valueOf("0.9")));
                    initKeyMaps();
          }
}
```

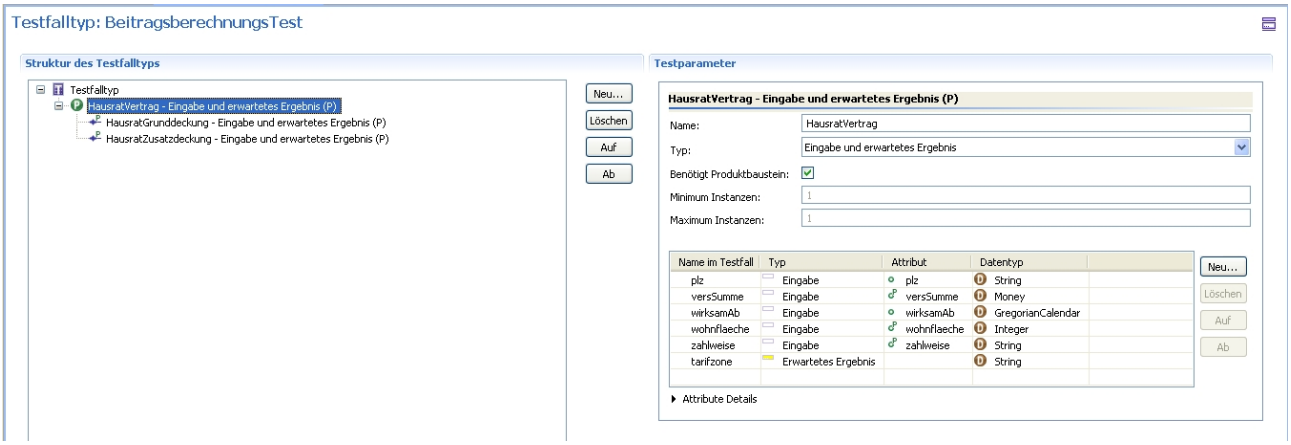
Erstellen einer initialize-Methode, in der die Test-Produktklassen und -Tabellen erzeugt und dem Repository hinzugefügt werden

```
private void initialize() {
          // Erstellen eines Test-Produktbausteins
          HausratProdukt testProdukt = new HausratProdukt(localRepository,
                              "TestProdukt 2009-01", "TestProdukt", "2009-01");
          HausratProduktAnpStufe testProduktAnpStufe = new HausratProduktAnpStufe(
                              testProdukt);
          testProduktAnpStufe.setValidFrom(DateTime.parseIso("2009-01-01"));
          testProduktAnpStufe.setProduktname("TestProdukt");
          testProduktAnpStufe.setVorschlagVersSummeProQm(Money.euro(500));

          // Hinzufügen der Anpassungstufe zum Repository. Es wird automatisch
          // auch der Produktbaustein hinzugefügt
          localRepository.putProductCmptGeneration(testProduktAnpStufe);

          // Erstellen eines Test-Grunddeckungsbausteins
          HausratGrunddeckungstyp testDeckungstyp = new HausratGrunddeckungstyp(
          localRepository, "TestDeckung 2009-01", "TestDeckung", "2009-01");
          TestHausratGrunddeckungstypAnpStufe testDeckungAnpStufe = new
                    TestHausratGrunddeckungstypAnpStufe(testDeckungstyp);
          testDeckungAnpStufe.setValidFrom(DateTime.parseIso("2009-01-01"));
          // Der Tabellenname muss der gleiche sein unter dem man die
          // Testtariftabelle im Repository abspeichert. Siehe unten
          testDeckungAnpStufe.setTariftabellenName("Tariftabelle");
          testProduktAnpStufe.setHausratGrunddeckungstyp(testDeckungstyp);

          // Hinzufügen der Anpassungstufe zum Repository. Es wird automatisch
          // auch der Baustein hinzugefügt
          localRepository.putProductCmptGeneration(testDeckungAnpStufe);

          // Hinzufügen der Testtarifzonentabelle zum Repository
          localRepository.putTable(new TestTarifzonentabelle());

          // Hinzufügen der Testtariftabelle zum Repository. Wichtig ist, da es
          // eine Tabelle mit multiplen Inhalten ist, muss der Name der Tabelle
          // angegeben werden, unter dem die Tabelle im Repository gespeichert
          // wird.
          localRepository.putTable(new TestTariftabelle(), "Tariftabelle");
}
```

- Hinzufügen einer weiteren test-Methode im HausratTest entsprechend der test-Methode, die den berechneten Beitrag der Grunddeckung berechnet, jedoch auf Basis der Daten des Repositories des TestContents.

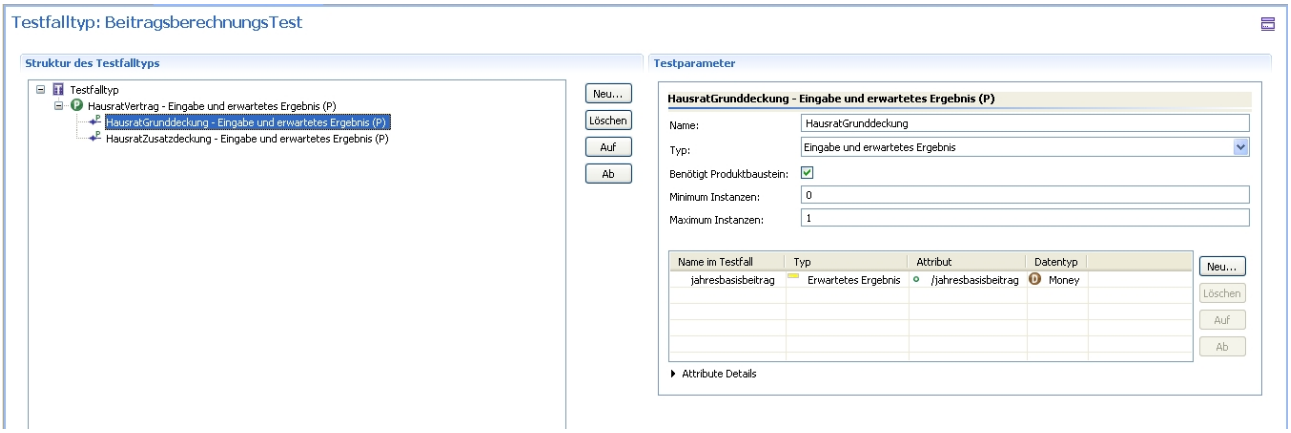
Übung 2

- Anlegen eines Testfalltyps *BeitragsberechnungsTest* im Package hausrat des Hausratmodell Projektes
- Aufbau der Struktur des Testfalltyps:
- Hinzufügen eines Hausratvertrages (benötigt produktbaustein: ja) und setzen der Attribute entsprechend dem Bildausschnitt:

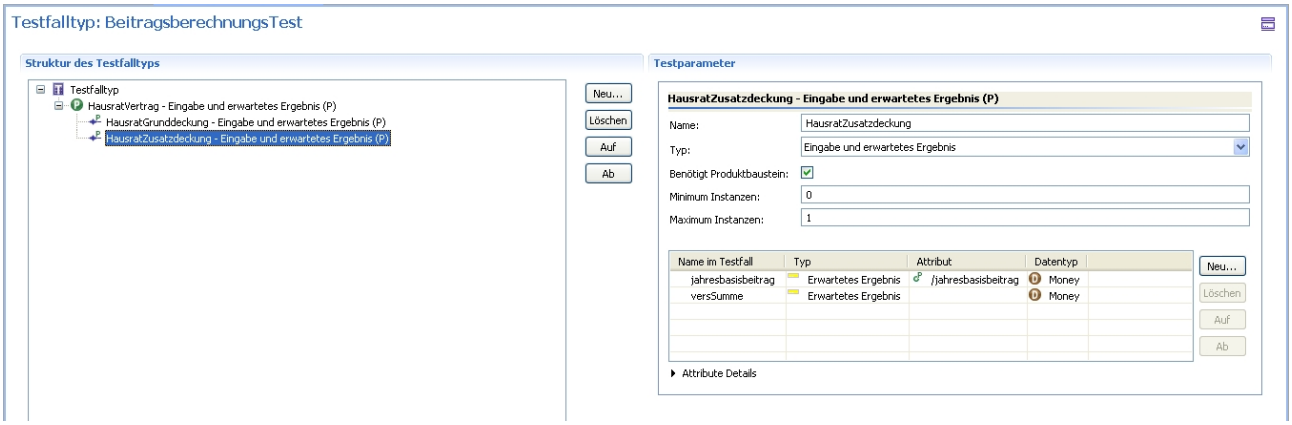


- Hinzufügen einer Hausratgrunddeckung zum HausratVertrag (benötigt produktbaustein: ja)

und setzen der Attribute entsprechend dem Bildausschnitt:



- Hinzufügen einer HausratZusatzdeckung zum HausratVertrag (benötigt produktbaustein: ja)



Übung 3

- Implementierung der generierten Testfalltyp Klasse *Beitragsberechnungstest* im Package org.faktorips.schulung.modell.internal.hausrat
- Implementierung der executeBusinessLogic-Methode

```
/**
 * Fuehrt die zu testende Geschaeftslogik aus.
 *
 * @generated NOT
 */
public void executeBusinessLogic() {
        for (IHausratDeckung deckung : inputHausratVertrag.getHausratDeckungen()) {
                deckung.berechneJahresbasisbeitrag();
        }
        inputHausratVertrag.berechneJahresbasisbeitrag();
}
```

- Implementierung der executeAsserts-Methode

```
/**
 * Fuehrt die Pruefungen (Asserts) aus, d.h. vergleicht die erwarteten Werten mit den tatsaechlichen Ergebnissen.
 *
 * @generated NOT
 */
public void executeAsserts(IpsTestResult result) {
        String tarifzoneErw = (String) getExtensionAttributeValue(erwartetHausratVertrag,
                        TESTATTR_HAUSRATZUSATZDECKUNG_VERSSUMME);
        String tarifzoneIst = inputHausratVertrag.getTarifzone();
        assertEquals(tarifzoneErw, tarifzoneIst, result, "HausratVertrag#0", "tarifzone");

        IHausratGrunddeckung grunddeckungErw = erwartetHausratVertrag.getHausratGrunddeckung();
        IHausratGrunddeckung grunddeckungIst = inputHausratVertrag.getHausratGrunddeckung();

        assertEquals(grunddeckungErw.getJahresbasisbeitrag(), grunddeckungIst.getJahresbasisbeitrag(), result,
                        "HausratVertrag#0.HausratGrunddeckung#0", "jahresbasisbeitrag");

        List<IHausratZusatzdeckung> zusatzdeckungenIst = inputHausratVertrag.getHausratZusatzdeckungen();
        List<IHausratZusatzdeckung> zusatzdeckungenErw = erwartetHausratVertrag.getHausratZusatzdeckungen();
        for (int i = 0; i < zusatzdeckungenErw.size(); i++) {
                assertEquals(zusatzdeckungenErw.get(i).getJahresbasisbeitrag(), zusatzdeckungenIst.get(i)
                                .getJahresbasisbeitrag(), result, "HausratVertrag#0.HausratZusatzdeckung#" + i,
                                "jahresbasisbeitrag");
                Money versSummeErw = (Money) getExtensionAttributeValue(zusatzdeckungenErw.get(i), "versSumme");
                Money versSummeIst = zusatzdeckungenIst.get(i).getVersSumme();
                assertEquals(versSummeErw, versSummeIst, result, "HausratVertrag#0.HausratZusatzdeckung#" + i,
                                "versSumme");
        }
}
```

Übung 4

- Erstellen eines IPS-Packages *test* unter dem IPS-Package *produkte* im Produktdaten Projekt
- Anlegen eines Testfall auf Basis des eben erzeugten Testfalltyps
  - Hierzu in der Toolbar den Button *Testfall erzeugen* klicken und im Wizard im Feld Testfalltyp den oben erzeugten angeben.
- Eigenen Testfall erstellen indem entsprechend des Testfalltype die produktabhängigen Vertragsklassen erzeugt werden und die Attribute und erwarteten Werte eingegeben werden.