



Faktor-IPS

Übungsunterlagen zur Schulung

(Dokumentversion 1470)

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

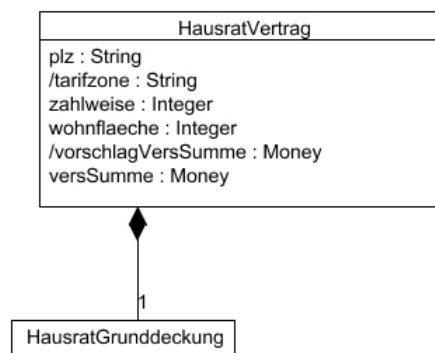
Legen Sie im Sourceverzeichnis „modell“ das Faktor-IPS Package „hausrat“ an.

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.

<i>Name : Datentyp</i>	<i>Beschreibung, Bemerkung</i>
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 $\text{wohnfläche} * 650 \text{ Euro}$. (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().

Übung zu Kapitel 2-2: Beziehungen auf Vertragsseite

Legen Sie die Klasse „hausrat.HausratGrunddeckung“ an.

Legen Sie die Composite-Beziehung zwischen HausratVertrag und HausratGrunddeckung mit der Multiplizität [0..1] an. Legen Sie auch die Rückbeziehung zwischen HausratGrunddeckung und HausratVertrag an, die Multiplizität ist ebenfalls [0..1].

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“ im Projekt „Hausratmodell“ 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.

Übung 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 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 Ressourcen 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;

@Before
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()`.

Übung 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:

<i>Konfigurationsmöglichkeit</i>	<i>HR-Kompakt</i>	<i>HR-Optimal</i>
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

Übung 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 Grunddeckungstypen enthalten).

Anlegen der Deckungsbausteine HRD-Optimal und HRD-Kompakt

Beziehung zwischen den Hausratprodukt- und den HausratDeckungsbausteinen herstellen

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

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 lqm: 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 die Spalten „plzVon“, „plzBis“ und „Tarifzone“ vom Typ String und einen Bereich für plzVon, plzBis. Legen Sie den UniqueKey auf diesen Bereich an.

<i>Plz-Von</i>	<i>Plz-bis</i>	<i>Tarifzone</i>
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. Die Methode soll für Postleitzahlen, die nicht in einen Bereich der Tarifzonentabelle fallen, nach wie vor „I“ zurückgeben.

Testen Sie ihre Implementierung, indem Sie den JUnit Test HausratTestMitProdukt 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“. Die Tarifzone ist vom Typ String, der Beitragssatz vom Typ Decimal.

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

Tariftabelle-Optimal

<i>Tarifzone</i>	<i>Beitragssatz pro 1000 Euro</i>
I	0.8
II	1.0
III	1.44
IV	1.70
V	2.00
VI	2.20

Tariftabelle-Kompakt

<i>Tarifzone</i>	<i>Beitragssatz pro 1000 Euro</i>
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 `HausratGrunddeckung` (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 `HausratTestMitProdukt` 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 Vertrag hinzufuegen
IHausratGrunddeckung deckung = vertrag.newHausratGrunddeckung(deckungstyp);
```

Übungen zu Kapitel 4.3: Aufzählungen

Übung 1

Legen Sie den Aufzählungstypen „Zahlweise“ im Projekt „Hausratmodell“ an. Legen Sie das Attribut „anzahlZahlungenProJahr“ vom Typ `int` an und tragen Sie folgende Werte ein:

ID	(Fachliche) Bezeichnung	Anzahl Zahlungen pro Jahr
Q	Quartalsweise	4
H	Halbjährlich	2
J	Jährlich	1

Ändern Sie den Datentyp des Attributs „zahlweise“ am `HausratVertrag` auf den soeben angelegten Aufzählungstyp. Verwenden Sie in einem JUnit Test die generierten Literale zum Setzen der Zahlweise:

```
hausratVertrag.setZahlweise(Zahlweise.HALBJAEBRLICH);
```

Übung 2

Exportieren Sie den Aufzählungsinhalt in eine Exceltabelle und ergänzen Sie dort die Zeile

M	Monatlich	12
---	-----------	----

Übung 3

Entfernen Sie den Inhalt aus dem Aufzählungstypen „Zahlweise“ und setzen Sie den Haken für „Werte in separatem Aufzählungsinhalt“. Erstellen Sie im Projekt „Hausratprodukte“ einen neuen Aufzählungsinhalt für „Zahlweise“ und importieren dort ihre Werte aus der Exceltabelle.

Übung 4

Legen Sie an der `HausratGrunddeckung` das bei Aufruf des Getters berechnete Attribut „Beitrag“ an. Implementieren Sie den Getter so, dass der Jahresbasisbeitrag geteilt durch die Anzahl der Zahlungen pro Jahr aus der gewählten Zahlweise zurückgegeben wird.

Anschließend schreiben Sie einen Test für `getBeitrag`. Den Zugriff auf die Aufzählungswerte erhalten Sie wie folgt über das `RuntimeRepository`:

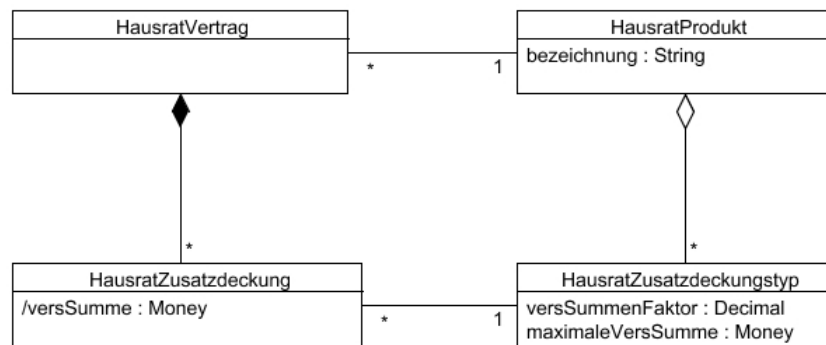
```
Zahlweise halbjaehrlich = repository.getEnumValue(Zahlweise.class, "H");
```

Dabei ist die Klasse des Aufzählungstypen und die ID des gesuchten Wertes anzugeben.

Übungen zum Kapitel 5

Übung 1

Erweitern Sie das Hausratmodell um die HausratZusatzdeckung und den HausratZusatzdeckungstyp gemäß dem folgenden Diagramm:



Achten Sie darauf beim Erzeugen der Beziehung zwischen **HausratVertrag** und **HausratZusatzdeckung** auch die Rückbeziehung anzulegen.

Legen Sie die folgenden Zusatzdeckungen an:

	Fahrraddiebstahl 2009-01	Überspannung 2009-01
VersSummenFaktor	0.01	0.05
MaximaleVersSumme	3000EUR	<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 **HausratZusatzdeckung** an.

Definieren Sie die Formelsignatur für die Berechnung des Jahresbasisbeitrags am **HausratZusatzdeckungstyp**. 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.

Übung zum Kapitel 6

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 `Deckung` an und geben Sie diese als Oberklasse von `HausratGrunddeckung` und `HausratZusatzdeckung` an. Legen Sie die abstrakte Klasse `Vertrag` als Oberklasse von `HausratVertrag` an. Legen Sie vom `Vertrag` eine Beziehung `Deckungen` zur `Deckung` an und markieren Sie diese als `derived union`. Tragen Sie diese Beziehung dann bei den bestehenden Beziehungen von `HausratVertrag` als die Beziehung ein, deren Subset diese Beziehungen jetzt sind. Verahren 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 `Vertrag` an und implementieren Sie diese Methode so, dass sie über alle `Deckungen` iteriert und deren `Basisbeitragsberechnung` anstößt und aufaddiert.

Übungen zum Kapitel 8

Übung 1

Erzeugen Sie im Hausratmodell Projekt eine neue Test-Klasse HausratGrunddeckungTest mit einer Test-Methode testGetBeitrag.

Um die Beitragsberechnung ohne produktive Produktdaten test zu können, sind einige Vorbereitungen notwendig. Gehen Sie wie folgt vor

- Erstellen Sie die Klasse TestContent mit einem Attribut vom Typ InMemoryRuntimeRepository
- Um das InMemoryRuntimeRepository der TestContent Klasse mit Testdaten füllen zu können, müssen einige Klassen abgeleitet werden:
 - Tarifzontentabelle: 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 TarifzontentabelleRow gefüllt.
 - TariftabelleHausrat: Ableitung analog zur Tarifzontentabelle
 - HaushaltGrunddeckungsTypAnpStufe: In einer Ableitung dieser Klasse kann der Name der Tariftabelle gesetzt werden, die dann beim Aufruf des getters aus dem Repository geladen wird

```
/**
 * Die Ableitung der Tarifzontentabelle ist notwendig um sie mit
 * Beispieldaten zu füllen.
 */
private static class TestTarifzontentabelle extends Tarifzontentabelle {
    @SuppressWarnings({ "unchecked", "rawtypes" })
    public TestTarifzontentabelle() {
        super();
        rows = new ArrayList();
        rows.add(new TarifzontentabelleRow("10000", "19999", "I"));
        rows.add(new TarifzontentabelleRow("20000", "29999", "II"));
        rows.add(new TarifzontentabelleRow("30000", "39999", "III"));
        initKeyMaps();
    }
}

/**
 * Die Ableitung der Tariftabelle ist notwendig um sie mit
 * Beispieldaten zu füllen.
 */
private static class TestTariftabelle extends TariftabelleHausrat {
    @SuppressWarnings({ "unchecked", "rawtypes" })
    public TestTariftabelle() {
        super();
        rows = new ArrayList();
        rows.add(new TariftabelleHausratRow("I", Decimal.valueOf("0.5")));
        rows.add(new TariftabelleHausratRow("II", Decimal.valueOf("0.7")));
        rows.add(new TariftabelleHausratRow("III", Decimal.valueOf("0.9")));
        initKeyMaps();
    }
}

private class TestHausratGrunddeckungsTypAnpStufe extends HausratGrunddeckungsTypAnpStufe {
    private String tabellenName;

    public TestHausratGrunddeckungsTypAnpStufe(HausratGrunddeckungsTyp productCmpt) {
        super(productCmpt);
    }

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

    @Override
    public TariftabelleHausrat getTariftabelle() {
        return (TariftabelleHausrat) localRepository.getTable(tabellenName);
    }
}
```

- Implementieren Sie eine 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.putProductCmptAnpStufeeration(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
    // Testtarifabelle 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 der Enum-Werte für die Zahlweise zum Repository
    List<Zahlweise> zahlweisen = new ArrayList<Zahlweise>();
    zahlweisen.add(new Zahlweise("J", "Jährlich", 1));
    zahlweisen.add(new Zahlweise("H", "Halbjährlich", 2));
    localRepository.putEnumValues(Zahlweise.class, zahlweisen);
}
```

- Implementieren Sie eine getter-Methode im TestContent, die ein initialisiertes InMemoryRepository liefert

Implementieren Sie nun die Methode testGetBeitrag am HausratGrunddeckungTest entsprechend der test-Methode in HausratTestMitProdukt aus dem Hausratprodukte Projekt, jedoch auf Basis der Daten des Repositories aus TestContent.

Übung 2

Binden Sie Mockito in den Java Build Path des Hausratmodell Projektes ein. Erstellen Sie im Testfall HausratGrunddeckungTest die Test-Methode testGetBeitragWithMocks.

Implementieren Sie die Test-Methode. Verwenden Sie Mocks für HausratVertrag, HausratGrunddeckungstypAnpStufe, TariftabelleHausrat, TariftabelleHausratRow

Übung 3

- 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:

Testfalltyp: BeitragsberechnungsTest

Struktur des Testfalltyps

- Testfalltyp
 - HausratVertrag - Eingabe und erwartetes Ergebnis (P)
 - HausratGrunddeckung - Eingabe und erwartetes Ergebnis (P)
 - HausratZusatzdeckung - Eingabe und erwartetes Ergebnis (P)

Neu...
Löschen
Auf
Ab

Testparameter

HausratVertrag - Eingabe und erwartetes Ergebnis (P)
Name: HausratVertrag
Typ: Eingabe und erwartetes Ergebnis
Benötigt Produktbaustein: ☒
Minimum Instanzen: 1
Maximum Instanzen: 1

Name im Testfall	Typ	Attribut	Datentyp
plz	Eingabe	plz	String
versSumme	Eingabe	versSumme	Money
wirksamAb	Eingabe	wirksamAb	GregorianCalendar
wohnflaeche	Eingabe	wohnflaeche	Integer
zahlweise	Eingabe	zahlweise	String
tarifzone	Erwartetes Ergebnis		String

Neu...
Löschen
Auf
Ab

Attribute Details

- Hinzufügen einer HausratGrunddeckung zum HausratVertrag (benötigt produktbaustein: ja) und setzen der Attribute entsprechend dem Bildausschnitt:

Testfalltyp: BeitragsberechnungsTest

Struktur des Testfalltyps

- Testfalltyp
 - HausratVertrag - Eingabe und erwartetes Ergebnis (P)
 - HausratGrunddeckung - Eingabe und erwartetes Ergebnis (P)
 - HausratZusatzdeckung - Eingabe und erwartetes Ergebnis (P)

Neu...
Löschen
Auf
Ab

Testparameter

HausratGrunddeckung - Eingabe und erwartetes Ergebnis (P)
Name: HausratGrunddeckung
Typ: Eingabe und erwartetes Ergebnis
Benötigt Produktbaustein: ☒
Minimum Instanzen: 0
Maximum Instanzen: 1

Name im Testfall	Typ	Attribut	Datentyp
jahresbasisbeitrag	Erwartetes Ergebnis	jahresbasisbeitrag	Money

Neu...
Löschen
Auf
Ab

Attribute Details

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

Testfalltyp: BeitragsberechnungsTest

Struktur des Testfalltyps

- Testfalltyp
 - HausratVertrag - Eingabe und erwartetes Ergebnis (P)
 - HausratGrunddeckung - Eingabe und erwartetes Ergebnis (P)
 - HausratZusatzdeckung - Eingabe und erwartetes Ergebnis (P)

Neu...
Löschen
Auf
Ab

Testparameter

HausratZusatzdeckung - Eingabe und erwartetes Ergebnis (P)
Name: HausratZusatzdeckung
Typ: Eingabe und erwartetes Ergebnis
Benötigt Produktbaustein: ☒
Minimum Instanzen: 0
Maximum Instanzen: 1

Name im Testfall	Typ	Attribut	Datentyp
jahresbasisbeitrag	Erwartetes Ergebnis	jahresbasisbeitrag	Money
versSumme	Erwartetes Ergebnis		Money

Neu...
Löschen
Auf
Ab

Attribute Details

Übung 4

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

- Implementierung der executeBusinessLogic-Methode

```
/**
 * Führt die zu testende Geschäftslogik aus.
 *
 * @restrainedmodifiable
 */
@Override
public void executeBusinessLogic() {
    // begin-user-code
    inputHausratVertrag.berechneJahresbasisbeitrag();
    // end-user-code
}
```

- Implementierung der executeAsserts-Methode

```
/**
 * Führt die Prüfungen (Asserts) aus, d.h. vergleicht die erwarteten Werten mit den tatsächlichen Ergebnissen.
 *
 * @restrainedmodifiable
 */
public void executeAsserts(IpsTestResult result) {
    // begin-user-code
    String tarifzoneErw = (String) getExtensionAttributeValue(erwartetHausratVertrag,
        TESTATTR_HAUSRATVERTRAG_TARIFZONE);
    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");
    }
    // end-user-code
}
```

Übung 5

- 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.