

Lösungen

In dem Sourcecode 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 Tarifzone

Klasse: Hausratvertrag

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

Ausschnitt 3: Ermittlung Jahresbasisbeitrag Grunddeckung

```
private Money berechneJahresbasisbeitragInternal() {
    HausratGrunddeckungstypGen gen = (HausratGrunddeckungstypGen)
        getHausratGrunddeckungstypGen();
    if (gen == null) {
        return Money.NULL;
    }
    TariftabelleHausrat tabelle = gen.getTariftabelle();
    TariftabelleHausratRow row =
        tabelle.findRow(getHausratVertrag().getTarifzone());
    if (row == null) {
        return Money.NULL;
    }
    Money vs = getHausratVertrag().getVersSumme();
    Decimal beitragsatz = row.getBeitragsatz();
    return vs.divide(1000, BigDecimal.ROUND_HALF_UP).multiply(beitragsatz,
        BigDecimal.ROUND_HALF_UP);
}
```

Ausschnitt 4: 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 5: Delegation an abgeleitete Objekte

Klasse: HausratDeckung

```
public void berechneJahresbasisbeitrag() {
    jahresbasisbeitrag = berechneJahresbasisbeitragInternal();
}

abstract protected Money berechneJahresbasisbeitragInternal();
```

Klasse: HausratGrunddeckung

```
protected Money berechneJahresbasisbeitragInternal() {
    [...]
    return vs.divide(1000, BigDecimal.ROUND_HALF_UP).multiply(beitragssatz,
        BigDecimal.ROUND_HALF_UP);
}
```

Klasse: HausratVertrag

```
public void berechneJahresbasisbeitrag() {
    Money beitrags = Money.euro(0);
    for (IDeckung deckung : getDeckungen()) {
        deckung.berechneJahresbasisbeitrag();
        beitrags = beitrags.add(deckung.getJahresbasisbeitrag());
    }
    jahresbasisbeitrag = beitrags;
}
```

Ausschnitt 6: JUnit Testfall

```
import java.util.GregorianCalendar;

import junit.framework.TestCase;

import org.faktorips.runtime.ClassloaderRuntimeRepository;
import org.faktorips.runtime.IProductComponent;
import org.faktorips.runtime.IProductComponentGeneration;
import org.faktorips.runtime.IRuntimeRepository;
import org.faktorips.tutorial.hausrat.hausrat.IHausratGrunddeckung;
import org.faktorips.tutorial.hausrat.hausrat.IHausratGrunddeckungstyp;
import org.faktorips.tutorial.hausrat.hausrat.IHausratProdukt;
import org.faktorips.tutorial.hausrat.hausrat.IHausratProduktGen;
import org.faktorips.tutorial.hausrat.hausrat.IHausratVertrag;
import org.faktorips.tutorial.hausrat.hausrat.IHausratZusatzdeckung;
import org.faktorips.tutorial.hausrat.hausrat.IHausratZusatzdeckungstyp;
import org.faktorips.tutorial.hausrat.internal.hausrat.HausratProdukt;
import org.faktorips.values.Money;

public class TutorialTest extends TestCase {

    private IRuntimeRepository repository;
    private IHausratProdukt kompaktProdukt;
    private IHausratProduktGen kompaktGen;

    public void setUp() {
        // wenn das toc-file nicht gelesen werden kann, wird eine runtime exception geworfen.
        repository = ClassloaderRuntimeRepository
            .create("org/faktorips/tutorial/produktdaten/internal/faktorips-repository-
toc.xml");

        // Referenz auf das Kompaktprodukt aus dem Repository holen
        IProductComponent pc = repository.getProductComponent("hausrat.HR-Kompakt 2009-01");

        // Juengste Productgeneration holen (wir wissen es gibt nur eine).
        IProductComponentGeneration pcGen = pc.getLatestProductComponentGeneration();

        // Auf die eigenen Modellklassen casten
        kompaktProdukt = (HausratProdukt) pc;
        kompaktGen = (IHausratProduktGen) pcGen;
    }

    public void testProduktdatenLesen() {
        System.out.println("Produktname: " + kompaktGen.getProduktname());
        System.out.println("Vorschlag Vs pro lqm: " + kompaktGen.getVorschlagVersSummeProQm());
        System.out.println("Default Zahlweise : " + kompaktGen.getDefaultZahlweise());
        System.out.println("Erlaubte Zahlweisen: " + kompaktGen.getAllowedValuesForZahlweise(null));
        System.out.println("Default Vs: " + kompaktGen.getDefaultVersSumme());
        System.out.println("Bereich Vs: " + kompaktGen.getRangeForVersSumme(null));
        System.out.println("Default Wohnflaeche: " + kompaktGen.getDefaultWohnflaeche());
        System.out.println("Bereich Wohnflaeche: " + kompaktGen.getRangeForWohnflaeche(null));
    }

    public void testGetTarifzone() {
        // Erzeugen eines hausratvertrags mit der Factorymethode des Produktes
        IHausratVertrag vertrag = kompaktProdukt.createHausratVertrag();

        // Wirksamkeitsdatum des Vertrages setzen, damit die Produktgeneration gefunden wird!
        // Dies muss nach dem Gueltigkeitsbeginn der Generation liegen!
        vertrag.setWirksamAb(new GregorianCalendar(2010, 0, 1));

        vertrag.setPlz("45525");
        assertEquals("III", vertrag.getTarifzone());
    }
}
```

```
public void testBerechneBeitrag() {
    // Erzeugen eines hausratvertrags mit der Factorymethode des Produktes
    IHausratVertrag vertrag = kompaktProdukt.createHausratVertrag();

    // Wirksamkeitsdatum des Vertrages setzen, damit die Produktgeneration gefunden wird!
    // Dies muss nach dem Gueltigkeitsbeginn der Generation liegen!
    vertrag.setWirksamAb(new GregorianCalendar(2010, 0, 1));

    // Vertragsattribute setzen
    vertrag.setPlz("45525"); // => tarifzone 3
    vertrag.setVersSumme(Money.euro(60000));
    vertrag.setZahlweise(new Integer(2)); // halbjahrlich

    // Grunddeckungstyp holen, der dem Produkt in der Generation zugeordnet ist.
    IHausratGrunddeckungstyp deckungstyp = kompaktGen.getHausratGrunddeckungstyp();

    // Grunddeckung erzeugen und zum Vertrag hinzufügen
    IHausratGrunddeckung deckung = vertrag.newHausratGrunddeckung(deckungstyp);

    // Beitrag berechnen und Ergebniss prüfen
    vertrag.berechneBeitrag();

    // tarifzone 3 => beitragsatz = 1.21
    // jahresbasisbeitrag = versicherungssumme / 1000 * beitragsatz = 60000 / 1000 * 1,21 = 72,60
    assertEquals(Money.euro(72, 60), deckung.getJahresbasisbeitrag());

    // Jahresbasisbeitrag vertrag = Jahresbasisbeitrag deckung
    assertEquals(Money.euro(72, 60), vertrag.getJahresbasisbeitrag());

    // NettobeitragZw = 72,60 / 2 * 1,03 (wg. Ratenzahlungszuschlag von 3%) = 37,389 => 37,39
    assertEquals(Money.euro(37, 39), vertrag.getNettobeitragZw());

    // BruttobeitragZw = 37,39 * Versicherungssteuersatz = 37,39 * 1,19 = 44,49
    assertEquals(Money.euro(44, 49), vertrag.getBruttobeitragZw());
}

public void testBerechneJahresbasisbeitragFahrraddiebstahl() {
    // Erzeugen eines hausratvertrags mit der Factorymethode des Produktes
    IHausratVertrag vertrag = kompaktProdukt.createHausratVertrag();

    // Wirksamkeitsdatum des Vertrages setzen, damit die Produktgeneration gefunden wird!
    // Dies muss nach dem Gueltigkeitsbeginn der Generation liegen!
    vertrag.setWirksamAb(new GregorianCalendar(2010, 0, 1));

    // Vertragsattribute setzen
    vertrag.setVersSumme(Money.euro(60000));

    // Zusatzdeckungstyp Fahrraddiebstahl holen
    // Der Einfachheit halber, nehmen wir hier an, der erste ist Fahrraddiebstahl
    IHausratZusatzdeckungstyp deckungstyp = kompaktGen.getHausratZusatzdeckungstyp(0);

    // Zusatzdeckung erzeugen
    IHausratZusatzdeckung deckung = vertrag.newHausratZusatzdeckung(deckungstyp);

    // Jahresbasisbeitrag berechnen und testen
    deckung.berechneJahresbasisbeitrag();

    // Versicherungssumme der Deckung = 1% von 60.0000, max 5.000 => 600
    // Beitrag = 10% von 600 = 60
    assertEquals(Money.euro(60, 0), deckung.getJahresbasisbeitrag());
}

}

public void testBerechneJahresbasisbeitragVertrag() {
    IHausratVertrag hausratVertrag = ((IHausratProdukt)
repository.getProductComponent("hausrat.HR-Optimal 2009-01")).createHausratVertrag();
    hausratVertrag.setWirksamAb(new GregorianCalendar(2009, Calendar.JANUARY, 1));
    hausratVertrag.setVersSumme(Money.euro(60000));
    hausratVertrag.setPlz("12345");
}
```

```
        IHausratGrunddeckungstyp deckungstyp =
hausratVertrag.getHauseatProduktGen().getHauseatGrunddeckungstyp();
        hausratVertrag.addHauseatGrunddeckung(deckungstyp.createHauseatGrunddeckung());

        for (IHauseatZusatzdeckungstyp zusatzdeckungsTyp :
hausratVertrag.getHauseatProduktGen().getHauseatZusatzdeckungstypen()) {
            IHauseatZusatzdeckung zusatzdeckung = zusatzdeckungsTyp.createHauseatZusatzdeckung();
            hausratVertrag.addHauseatZusatzdeckung(zusatzdeckung);
        }
        hausratVertrag.berechneJahresbasisbeitrag();
        // Beitrag = Grundbeitrag(48€) + Fahrradbeitrag(60€) + Überspannungsbeitrag(100€)
        assertEquals(Money.euro(208, 0), hausratVertrag.getJahresbasisbeitrag());
    }
```