TENTAMEN

Softwaresystemen

vakcode: 201300071
datum: 19 januari 2015
tijd: 8:45 - 11:45

Algemeen

- Bij dit tentamen mag gebruik gemaakt worden van: de handleiding, de slides van de colleges en de boeken die voorgeschreven zijn als studiemateriaal voor de module (of, als je deze niet hebt, een afdruk/kopie van de betreffende pagina's).
  Je mag *geen* gebruik maken van: uitwerkingen van opgaven die op Blackboard gepubliceerd zijn (recommended exercises en oude tentamens) of eigen materiaal (afdrukken van je eigen practicumopgaven, aantekeningen in welke vorm dan ook).
- Het aantal punten voor het tentamen wordt meegenomen in de berekening van het eindcijfer van de module, op de manier zoals aangegeven in de handleiding.
- Dit tentamen bestaat uit 7 opgaven, waarvoor in het totaal 100 punten behaald kunnen worden. Het minimale aantal punten per opgave bedraagt 0 punten. Het eindcijfer van het tentamen wordt bepaald door de optelsom van de punten per opgave.
- Studenten die *alleen* de programmeerlijn volgen, of *alleen* het vak Programmeren 2 herkansen, hoeven Opgave 7 niet te maken. Zij kunnen maximaal 85 punten halen. Het eindcijfer voor het tentamen wordt als volgt berekend: (aantal behaalde punten * 10/85).
- Studenten die *alleen* de ontwerplijn volgen, hoeven alleen Opgave 7 te maken. Zij kunnen maximaal 15 punten halen. Het eindcijfer voor het tentamen wordt als volgt berekent: (aantal behaalde punten * 10/15).

Opgave 1 **(15 punten)**

In dit tentamen gaan we een aantal interfaces, klassen en methoden ontwikkelen die gebruikt kunnnen worden voor een database met koopwoningen.

a. *(5 punten)* Definieer een interface `Woning` met methoden om de volgende kenmerken van een woning op te vragen: prijs, aantal kamers, en aantal verdiepingen. Geef specificaties die het gewenste gedrag van de methoden beschrijven. N.B. er worden niet hele uitgebreide specificaties verwacht, maar wel meer dan alleen `ensures true;`

b. *(5 punten)* Een appartement is de benaming voor een woning van één verdieping (bijvoorbeeld in een flat). Implementeer een klasse `Appartement`. Denk hierbij ook aan de constructor.

c. *(5 punten)* Een studio of eenkamerappartement is een klein appartement bestaande uit één kamer, meestal bedoeld voor één persoon. Implementeer een klasse `Studio` als uitbreiding van `Appartement`. Denk hierbij ook aan de constructor.

Opgave 2 (20 punten)

Hieronder staat een klasse `Aanbod` die een verzameling beschikbare koopwoningen bijhoudt.

```
public class Aanbod {
    //@ private invariant aanbod.size() == aantal;
    private Set<Woning> aanbod= new HashSet<Woning>();
    private int aantal;

    /** Retourneert alle woningen uit het aanbod. */
    /*@ pure */ public Set<Woning> getAanbod() {
        return aanbod;
    }

    /** Retourneert een map die gegeven een aantal kamers k,
      * alle huizen oplevert met k kamers. */
    public Map<Integer,Set<Woning>> groepeerOpAantalKamers() {
        // te implementeren
    }

    /** Retourneert de n huizen uit het aanbod met de laagste prijs, gesorteerd
      * van laag naar hoog. */
    public List<Woning> laagstePrijs(int n) {
        // te implementeren
    }

    public void nieuweWoning(Woning w) {
        aanbod.add(w);
        aantal = aantal + 1;
    }
```

a. *(10 punten)* Implementeer de methode `groepeerOpAantalKamers`. Deze methode levert een map op die gegeven een aantal kamers, alle woningen oplevert met dit aantal kamers. Dus gegeven een key `k`, zal `groepeerOpAantalKamers.get(k)` alle woningen met `k` kamers opleveren.

b. *(10 punten)* Implementeer de methode `laagstePrijs`. Deze levert de `n` goedkoopste huizen uit het aanbod, gesorteerd van laag naar hoog.

Opgave 3 (15 punten)

a. *(5 punten)* Pas de implementatie van `nieuweWoning` aan, zodat deze een zelfgedefinieerde exceptie oplevert als de woning al in het aanbod voorkomt.

b. *(10 punten)* Voeg een methode aan `Aanbod` toe, *inclusief specificatie*

```
public Set<Woning> nieuweWoningen(Set<Woning> ws)
```

die probeert alle woningen in `ws` aan het aanbod toe te voegen, en alle woningen waarvoor dit niet lukt (omdat de woning al in het aanbod zit) teruglevert.

Opgave 4 (10 punten)

De volgende interface bevat functionaliteit om een selectie uit een woningaanbod te maken.

```
public interface Wens {
    /** Retourneert alle aangeboden woningen die voldoen aan deze wens. */
    public Set<Woning> passend(Aanbod a);
}
```

Geef twee implementaties van dit interface:

a. *(5 punten)* Een klasse `PrijsWens`, met velden voor de onder- en bovengrens aan de prijs van de gewenste woningen. Een woning voldoet aan een `PrijsWens` als de prijs van die woning in het opgegeven bereik ligt.

b. *(5 punten)* Een klasse `CombinatieWens`, met een veld `Set<Wens> wensen` waarin deelwensen verzameld zijn; een woning voldoet aan de `CombinatieWens` als hij aan alle deelwensen voldoet.

Opgave 5 (10 punten)

Verschillende makelaars en kopers kunnen tegelijkertijd, in parallel, het aanbod bekijken, door methoden in de klasse `Aanbod` aan te roepen. Makelaars kunnen ook de methode `nieuweWoning` aanroepen, om een woning aan het aanbod toe te voegen. *Ga in deze opgave uit van de oorspronkelijke implementatie van `nieuweWoning` zoals hierboven gegeven, en niet van je eigen versie van opgave 3.*

Met deze implementatie is het mogelijk dat 2 makelaars tegelijkertijd een woning proberen toe te voegen.

a. *(3 punten)* Leg uit wat er dan fout kan gaan

b. *(4 punten)* Illustreer dit aan de hand van een voorbeeldexecutie

c. *(3 punten)* Leg uit hoe dit opgelost kan worden

Opgave 6 (15 punten)

Stel je voor dat bovenstaande woningverkoop-site inderdaad geïmplementeerd is en waar gedurende de tijd vele additionele features aan toegevoegd zijn. Een van de toegevoegde features is accounts (voor makelaars en bezoekers) met een wachtwoord login. Oorspronkelijk werden de wachtwoorden simpelweg opgeslagen, maar later werd in plaats daarvan de cryptografische hash (bijvoorbeeld MD5 of SHA256) van het wachtwoord opgeslagen.

a. *(3 punten)* Stel de site staat enkel toe dat het wachtwoord precies uit 6 karakters bestaat en enkel cijfers, de letters a t/m d, de letters A t/m D en de karakters ";" en ":" gebruikt mogen worden. Hoeveel verschillende mogelijkheden voor wachtwoorden zijn er en ligt je antwoord toe?

b. *(5 punten)* Waarom is het slimmer om de hash van het wachtwoord op te slaan? Wat is het voordeel van een hash functie als scrypt ten opzichte van MD5 of SHA256 voor het opslaan van wachtwoorden?

Nu blijkt er een probleem te zijn met de site en jij wordt gevraagd het op te lossen. Je komt onderstaande broncode tegen. Blijkbaar heeft iemand bedacht om voor de zogenaamde flexibiliteit ergens in het login proces het wachtwoord te voorzien met een string dat beschrijft welke (cryptografische) hash-functie gebruikt moet worden. Dus een wachtwoord "fiets" in combinatie met de hash functie MD5 wordt doorgeven als "MD5:fiets".

```
private Map<String, String> passwordDB;

/**
 * Generates the hex-encoded hash of the password using a very flexible
 * scheme. The hash-function to use is embedded in the password: it is
 * separated by a colon. Example passwords: "MD5:abcd" or "SHA1:s3cr3t".
 * @throws NoSuchAlgorithmException
 */
public String getPWHash(String password) throws NoSuchAlgorithmException {
    String[] r = password.split(":");
    String prefix = r[0];
    String realPassword = r[1];
    MessageDigest md = MessageDigest.getInstance(prefix);
    md.update(realPassword.getBytes());
    byte[] digest = md.digest();
    return Hex.encodeHexString(digest);
}

public boolean login(String username, String password) {
    boolean result = true;
    if (passwordDB.containsKey(username)) {
        try {
            String passwordHash = getPWHash(password);
            if (!passwordDB.get(username).equals(passwordHash)) {
                result = false;
            }
        } catch (Exception e) {
            // Whatever, shouldn't happen, right?
        }
    } else {
        result = false;
    }
    return result;
}
```

De javadoc van de methode `String.split` vermeldt het volgende:

`public String[] split(String regex)`

Splits this string around matches of the given regular expression.

This method works as if by invoking the two-argument split method with the given expression and a limit argument of zero. Trailing empty strings are therefore not included in the resulting array.

The string “boo:and:foo”, for example, yields the following results with these expressions:

Regex	Result
`:`	`{ "boo", "and", "foo" }`
`o`	`{ "b", "", ":and:f" }`

Parameters

regex - the delimiting regular expression

Returns

the array of strings computed by splitting this string around matches of the given regular expression

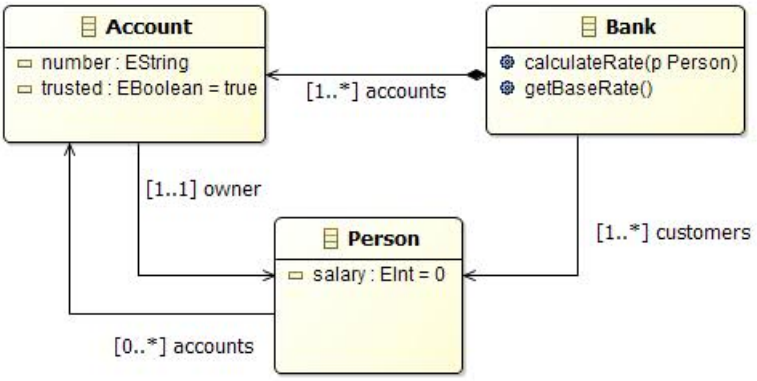
c. *(7 punten)* Stel een aanvaller heeft volledige controle over de inhoud van de variabelen `username` en `password` die meegegeven worden aan de `login`-methode. Er zijn (minstens) twee manieren voor deze aanvaller om de ervoor te zorgen (door manipulatie van de `username` en `password` variabelen) dat `login` true oplevert zonder dat de aanvaller echt een wachtwoord weet. Geef er minstens 1. Geef ook aan hoe je dit probleem (eenvoudig) zou kunnen verhelpen.

Opgave 7 (15 punten)

Deze opgave hoeft alleen gemaakt te worden door studenten die de designlijn volgen, dus niet door studenten die alleen de programmeerlijn doen, of alleen Programmeren 2 herkansen.

In deze opgave wordt op twee plekken een OCL-constraint gevraagd. Deze hoeven niet syntactisch helemaal te kloppen, maar in grote lijnen moeten ze voldoen aan het formaat van OCL — zie §10.5 van Bennett et al.

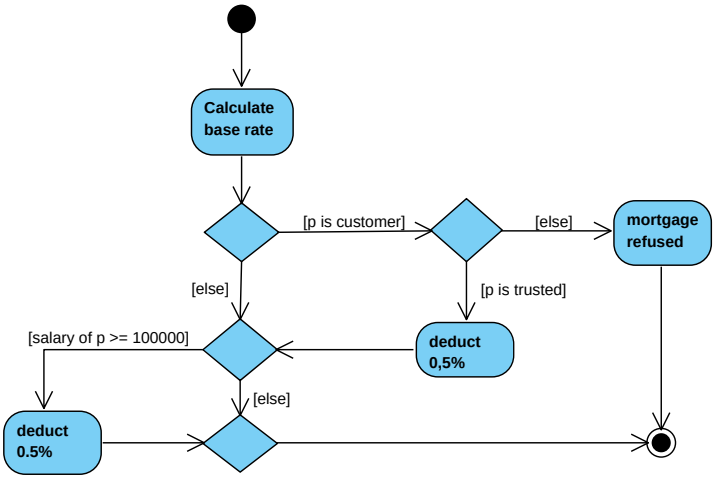
Om een hypotheek te verkrijgen laat een persoon zijn rentepercentage uitrekenen. De klassen die hierbij komen kijken zijn hieronder grafisch weergegeven.



a. (*3 punten*) Welke Java-representatie (m.a.w., welk type) zou je kiezen voor de vier associaties in het diagram?

b. (*3 punten*) Geef een model constraint voor het bovenstaande diagram in OCL.

Het volgende activity diagram specificeert de operatie `calculateRate(p:Person)` van de klasse `Bank`, waarin de rente van een hypotheek wordt berekend.



c. (*3 punten*) Geef een equivalente specificatie in de vorm van een decision table (zie Fig. 10.1 van Bennett et al.)

d. (*3 punten*) Geef een equivalente specificatie in de vorm van pseudo-code (zie §10.4 van Bennett et al.)

e. (*3 punten*) Geef een equivalente specificatie in de vorm van een OCL-constraint