TENTAMEN

Softwaresystemen

vakcode: 201300071
datum: 20 januari 2014
tijd: 8:45 - 12:15

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 5 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.

Opgave 1 (30 punten)

In deze opgave gaan we aantal klassen definiëren om schaatsers en hun trainingsarbeid te modelleren.

Elke schaatser wordt gemodelleerd door een *duurvermogen* en een *explosiviteit*. Beide waarden zijn integers.

a. *(4 punten)* Definier een interface `Skater`. Voor elke schaatser is het mogelijk om zijn duurvermogen en zijn explosiviteit op te vragen en aan te passen. Om duurvermogen en explosiviteit aan te passen wordt als argument meegegeven welke hoeveelheid aan deze waarden *toegevoegd* moeten worden.

b. *(6 punten)* Voeg JML specificaties toe aan de interface `Skater` die het volgende beschrijven:

   - duurvermogen en explosiviteit zijn nooit negatief;
   - explosiviteit kan alleen maar groeien;
   - duurvermogen kan wel afnemen, maar nooit negatief worden.

Er zijn verschillende type schaatsers, zoals langebaanschaatsers en kunstschaatsers. Binnen de langebaanschaatsers kunnen we vervolgens onderscheid maken tussen sprinters en stayers. Langebaanschaatsers trainen door het schaatsen van rondjes en het maken van starts. De effecten van deze oefeningen zijn echter verschillend voor sprinters en stayers.

c. *(5 punten)* Definieer een abstracte klasse `SpeedSkater` (voor langebaanschaatser) die de interface `Skater` implementeert. Voeg aan deze klasse methoden `rounds(int number)` en `starts(int number)` toe, die gebruikt kunnen worden om de effecten van de training te modelleren.

d. *(5 punten)* Implementeer nu een klasse stayer, zodanig dat:

   - elke 5 trainingsrondjes het duurvermogen van een stayer met 1 verhogen; en
   - elke 10 starts de explositiviteit met 1 verhogen, maar het duurvermogen met 1 af laten nemen.

   Let op: het is mogelijk om `starts(int number)` aan te roepen als de stayer te weinig of geen duurvermogen heeft. Maar zorg ervoor dat je implementatie de geërfde specificaties van `Skater` erft.

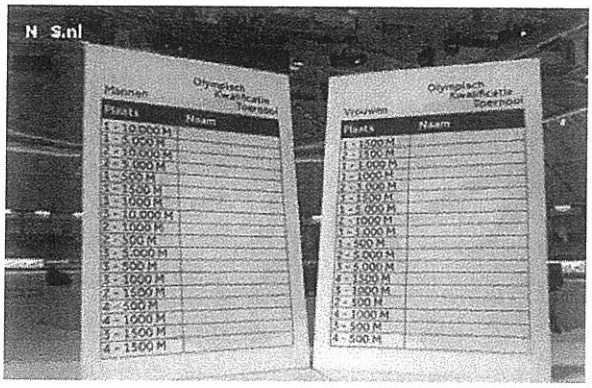
e. *(3 punten)* Geef specificaties voor de methoden `rounds` en `starts`.

f. *(3 punten)* Voor sommige schaatsers is de locatie van hun training erg belangrijk. Definieer een type om de verschillende ijshallen te modelleren. Neem daarin in elk geval de volgende hallen op: Thialf, Salt Lake City, Calgary, Sotsji, Vikingskibet.

g. (4 punten) Sven Kramer is een voorbeeld van een schaatser waarvoor de locatie erg belangrijk is. Definieer een klasse `Sven` die `Stayer` uitbreidt. De `Sven` klasse houdt de trainingslocatie bij. Sven's favoriete trainingslocatie is Thialf. Als Sven daar traint, dan leveren elke 5 trainingsrondjes niet alleen 1 extra duurvermogen op, maar ook 1 extra explosiviteit.

Opgave 2 (30 punten)

In deze opgave kijken we naar kwalificatiecriteria voor de Olympische Spelen. Hiervoor wordt een zogenaamde "matrix" opgesteld. De matrix is een geordende lijst van mogelijke uitslagen per afstand. Op plaats 1 staat bijvoorbeeld de nummer 1 van de 10 km, op plaats 2 staat de nummer 1 van de 5 km enz. Hier zie je als voorbeeld de matrix die eind december gebruikt werd voor de plaatsing van schaatsers voor de Olympische Spelen in Sotsji.



a. *(4 punten)* Definieer een type `Distances` van mogelijke afstanden (500 m, 1000 m, 1500 m, 5.000m en 10.000m) en een type `Ranking` voor de eerste vier plaatsen.

Gegeven is:

- een overzicht van de rankings die elke schaatser gehaald heeft: `Map<Skater, Map<Distance, Ranking>> results` (let op: voor elke schaatser `x` bevat de `Map results.get(x)` alleen de resultaten voor die afstanden waar schaatser `x` één van de eerste vier plaatsen gehaald heeft);
- een matrix schema dat voor elke afstand en plaats een positie op de matrix aangeeft: `Map<Distance,Map<Ranking, Integer>>> schema`; en
- een constante `public static final int MAX = 20;` die het maximaal aantal kwalificatieplekken definieert.

b. *(5 punten)* Schrijf JML specificaties om te specificeren dat voor alle afstanden en alle eerste 4 plaatsen er een positie tussen 1 en `MAX` op de matrix gedefinieerd is.

c. *(8 punten)* Schrijf een method `fillInMatrix` die een array field `Skater [] standing` van lengte `MAX` van schaatsers initialiseert en invult volgens de positie op de matrix.

d. *(3 punten)* Schrijf een methode `matrixComplete` die controleert of alle elementen in `standing` ingevuld zijn.

e. *(3 punten)* Specificeer een loop invariant voor de methode `matrixComplete`.

f. *(3 punten)* Definieer een methode `public Set<Skater> firstN(int n)` die een verzameling met de eerste `n` verschillende schaatsers oplevert (of minder schaatsers, als er niet voldoende verschillende schaatsers in `standings` staan).

g. *(4 punten)* Specificeer een zo volledig mogelijke loop invariant voor de methode `firstN`.

Opgave 3 (15 punten)

Gegeven is de volgende thread-klasse `Stream`:

```
class Stream extends Thread {

    private Lock lock;
    private char token;

    public Stream(char x, Lock l) {
        token = x;
        lock = l;
    }

    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.print(token);
            System.out.print(token);
        }
    }
}
```

Deze klasse declareert een field `Lock l`, maar deze wordt in eerste instantie nog niet gebruikt. Vanuit de volgende `main` methode worden er twee `Stream` threads aangeroepen.

```
public static void main(String[] args) {
    Lock l = new ReentrantLock();
    Stream str1 = new Stream('1', l);
    Stream str2 = new Stream('a', l);
    str1.start();
    str2.start();
}
```

a. *(4 punten)* Beschrijf de mogelijke uitvoer van dit programma.

b. *(4 punten)* Stel dat de lock nu ook echt gebruikt en de body van de `for` loop als volgt wordt aangepast:

```
lock.lock();
System.out.print(token);
System.out.print(token);
lock.unlock();
```

Beschrijf wat nu de mogelijke uitvoer van dit programma is en licht de verandering toe.

c. *(3 punten)* Wat gebeurt er als de aanroep `lock.unlock()` weggelaten wordt? Geef een verklaring voor dit gedrag.

d. *(4 punten)* Wat gebeurt er met het programma als in de `Stream` klasse het `Lock` object wordt aangemaakt, in plaats van dat het meegegeven wordt aan de constructor?

Opgave 4 (15 punten)

a. *(1 punt)* Terwijl je een system aan het analyseren bent, vind je de volgende string: `c2VjcmV0IGl1c3NhZ2U=`. Hoogstwaarschijnlijk is een soort encoding gebruikt. Wat zou je eerste gok zijn?

b. *(2 punten)* Stel, je weet dat een bepaald account beveiligd wordt door een wachtwoord van 4 of 5 karakters en dat enkel 'a', 's', 'd', 'f', 'q', 'w', 'e', 'r', '1', '2', '3', en '4' gebruikt worden. Hoeveel verschillende wachtwoorden zijn er mogelijk?

c. *(4 punten)* Op de marketing pagina van een cloud service vind je de volgende tekst: "This product uses the advanced MD5 public key encryption scheme to protect your data!". Geef (minstens) twee redenen waarom deze marketinguiting niet veel vertrouwen zal geven in de veiligheid van het product.

d. *(4 punten)* Stel je onderschept een vercijferd bericht. Nu weet je dat het onvercijferde bericht moet beginnen met "`<p>`" en dat het vercijferd is met de volgende code:

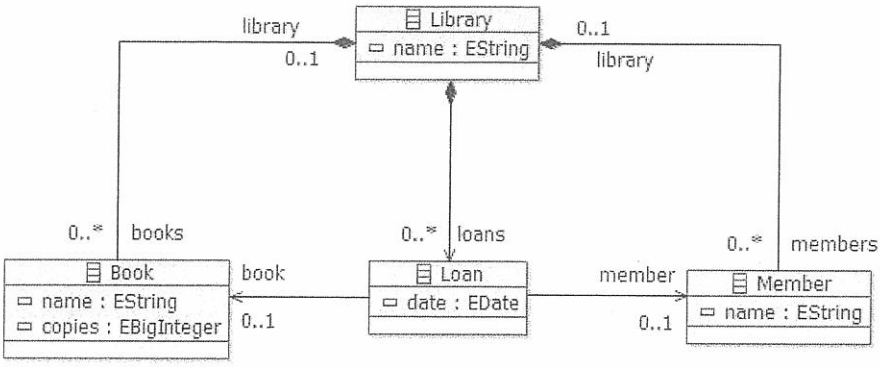
```
/**
 * Make encryption of message
 * @param pincode The pincode to use for encryption. The length of the
 * pincode should be 4 characters and only digits should be used.
 * @param message Message to encrypt.
 * @return
 */
public String encryptMessage(String pincode, String message) {
    MessageDigest md = null;
    try {
        md = MessageDigest.getInstance("SHA-256");
    } catch (NoSuchAlgorithmException e) {
        // Should not happen?
        e.printStackTrace();
    }
    md.update(pincode.getBytes());
    byte[] key = md.digest();
    byte[] ciphertext = secureEncryption(key, message.getBytes());
    return Base64.encodeBase64String(ciphertext);
}
```

Je hebt ook de beschikking tot de bijbehorende functie `decryptMessage()`. Leg uit (in woorden dus) hoe je de plaintext zou kunnen achterhalen.

e. *(4 punten)* Leg uit waarom je voor het opslaan van wachtwoorden beter gebruik kunt maken van BCrypt dan een 'standaard' hash functie als SHA1 of SHA-256.

Opgave 5 (10 punten)

In deze opdracht beschouwen we een UML klassendiagram van een bibliotheeksysteem zoals in de figuur hieronder.



In deze figuur komen de volgende klassen voor:

- `Library`: representeert een bibliotheek, heeft boeken (`books`), open leningen (`loans`) en leden (`members`).
- `Book`: representeert een bepaalde titel, waarvan meerdere fysieke kopieën mogen bestaan.
- `Loan`: representeert een open lening, waarin een kopie van een boek wordt gerelateerd (geleend) aan een lid van de bibliotheek.
- `Member`: representeert een persoon die lid is van de bibliotheek.

Stel dat we een beperking ('constraint') aan dit model willen toevoegen die bepaalt dat er niet meer geleende kopieën van een bepaald boek mogen zijn dan het aantal kopieën die er van dit boek bestaan (attribuut `copies`). Dit is een redelijke aanname, maar het model in de figuur voldoet niet aan deze beperking.

a. *(2 punt)* Wat zou de context van deze constraint moeten zijn, d.w.z. vanuit welke klasse zou deze OCL constraint gedefinieerd moeten worden? Waarom?

b. *(3 punten)* Geef informeel (in tekst) of in OCL de uitspraak in termen van de klassen (objecten) en attributen (referenties) nodig om deze constraint te beschrijven.

Stel dat we een operatie `makeLoan(b Book, m Member)` aan de `Library` klasse willen toevoegen.

c. *(2 punt)* Geef een informele beschrijving van de mogelijke precondities en postcondities van deze operatie.

d. *(2 punt)* Geef een beschrijving van deze operatie in 'Structured English' zoals beschreven in Hoofdstuk 10 van Bennett et al.

e. *(1 punt)* Welke van deze beschrijvingen zijn algoritmisch en niet-algoritmisch?