Tentamen Object Georiënteerd Programmeren TI1206
29 oktober 2014, 9.00-11.00
Afdeling SCT, Faculteit EWI, TU Delft

Bij dit tentamen mag je *geen* gebruik maken van hulpmiddelen zoals boek of slides. Digitale middelen zoals telefoon, tablet, ... stop je in je boekentas/rugzak.

Ter informatie:

- Elke vraag heeft 1 juist antwoord
- Er zijn 25 vragen, elke vraag heeft hetzelfde gewicht.
- Je vult je antwoorden in op een apart blad dat automatisch verwerkt zal worden

Vraag 1 Als we 2 logische waarden A en B willen combineren kunnen we gebruik maken van een lazy operator. Welke uitspraak over een een lazy operator is waar?

a. & → deze operator vergelijkt B niet meer als A false is.
b. && → deze operator vergelijkt B niet meer als A false is.
c. | → deze operator vergelijkt B niet meer als A false is.
d. || → deze operator vergelijkt B niet meer als A false is.

Vraag 2 Deze lus-constructie

```
do
{
        // statements
} while (x > 0)
```

a. Compileert niet.
b. Geeft een runtime fout.
c. Zal altijd 1 keer worden uitgevoerd.
d. Zal alleen worden uitgevoerd als x groter is dan 0.

Vraag 3 Voor de methode met signatuur *public int max(int x, int y)* geldt:

a. Compileert niet.
b. x en y zijn formale parameters
c. x en y zijn actuele parameters

Vraag 4 Een breakpoint is een punt in je programma waar:

a. Een fout zit
b. Waar een break statement staat (om bijvoorbeeld een case van een switch te eindigen).
c. Waar de debugger even stopt.
d. Waar een iteratie van een lus-structuur eindigt.

Vraag 5 Welke uitspraak over een constructor is waar:

a. Je bent niet verplicht om een constructor te definieren.
b. Je kan slechts 1 constructor definieren.
c. Je kan een constructor maken met een returnwaarde.
d. Volgende 2 constructors kunnen in dezelfde klasse Adres gedefinieerd worden:
    - public Adres(String straatnaam, int huisnummer, String postcode, String gemeente)
    - public Adres(String straatnaam, int huisnummer, String gemeente, String postcode)

Vraag 6 Ga uit van een klasse Punt met 2 attributen x en y, allebei van het type int. Gegeven deze equals methode:

```
public boolean equals(Punt other){
        if (other instanceof Punt){
                Punt that = (Punt)other;
                return (this.x == that.x) && (this.y == that.y);
        }
        return false;
}
```

In een testmethode staat nu dit:

```
Punt a = new Punt(3,4);
Punt b = new Punt(3,4);
assertEquals(a, b);
```

Deze code:

a. Zal niet compileren.
b. Zal aanleiding geven tot een runtime fout.
c. Geeft als resultaat een geslaagde test. (“pass”)
d. Geeft als resultaat een falende test (“fail”).

Vraag 7 Voor volgende stuk code geldt:

```
private static double fRente = 3.15;

public static double geefRente(double correctiefactor)
{
        return this.rente * correctiefactor;
}
```

a. Deze compileert niet.
b. Deze zal een runtime fout opleveren.
c. De returnwaarde is 6.30 als de parameter waarde 2.0 heeft.
d. Geen van de bovenstaande.

Vraag 8 Neem aan dat klasse A een publiek attribuut x heeft. Wat is het gevolg van volgende code:

```
A a = new A();
a.x = 6;
if(a.x = 5)
        System.out.println(“Gelijk aan 5”);
else if(a.x > 5)
        System.out.println(“Groter dan 5”);
else
        System.out.println(“Kleiner dan 5”);
```

a. Deze compileert niet.
b. Op het scherm verschijnt “Gelijk aan 5”.
c. Op het scherm verschijnt “Groter dan 5”.
d. Op het scherm verschijnt “Kleiner dan 5”.

Vraag 9 Wat is het gevolg van volgende code:

```
public int[] copy(int[] rij){
        int n = rij.length();
        int[] res = new int[n];
        for(int i = 0; i < n; i = i + 1)
        res[i] = rij[i];
        return res;
}
```

a. Deze compileert niet.
b. Deze geeft een ArrayOutOfBoundsException.
c. Deze geeft een ArrayIndexOutOfBoundsException.
d. Deze werkt naar behoren.

Vraag 10 Er bestaat een basisklasse Persoon, daarvan afgeleid zijn Student en Docent. Er moet een gemeenschappelijke lijst van Studenten en Docenten komen. Er bestaat een toString methode in de klasse Persoon die stopt enkel de naam van de Persoon in een String.

Nu wordt volgende code uitgevoerd.

1. ArrayList personenlijst = new ArrayList();
2. personenlijst.add(new Docent("Andy"));
3. personenlijst.add(new Student("Tim"));
4. boolean check = personenlijst.get(0).equals("Andy");
5. Persoon p = personenlijst.get(1);

De compiler geeft een fout aan op lijn:
a. 1
b. 2 en 3
c. 4
d. 5

Vraag 11 Welke uitspraak over het Liskov substitutiebeginsel is het meest waar?
a. Als A een superklasse van B is, dan is overal waar een object van type A gevraagd wordt, een object van type B ook acceptabel.
b. Als A een superklasse van B is, dan is overal waar een object van type A gevraagd wordt, een object van type B ook acceptabel, zolang alle methoden die in A staan gedefinieerd ook expliciet worden gedefinieerd in B.
c. Als A een subklasse van B is, dan is overal waar een object van type A gevraagd wordt, een object van type B ook acceptabel.
d. Als A een subklasse van B is, dan is overal waar een object van type A gevraagd wordt, een object van type B ook acceptabel, zolang alle methoden die in B staan gedefinieerd ook expliciet worden gedefinieerd ook in A staan.

Vraag 12 Welke uitspraak is niet waar: als je goede testen wil schrijven dan moet je...

a. Elke methode minstens 1 keer testen
b. Zoveel mogelijk asserts uitvoeren per test-methode
c. Extra peek en poke methodes schrijven
d. Extra get en set methodes schrijven

Vraag 13 Welke uitspraak is waar? Gebruik maken van het keyword "synchronized" kan...

a. Ervoor zorgen dat threads altijd netjes samenwerken
b. Race conditions tot gevolg hebben
c. Deadlock tot gevolg hebben
d. Geen van bovenstaande

Vraag 14 Als een klasse A de interface Runnable implementeert dan:
a. Moet klasse A een methode *public static void main(String[] args)* implementeren
b. Moet klasse A een methode *public void start()* definieren
c. Moet klasse A een methode *public void run()* definieren
d. Moet klasse A een attribuut van het type Thread hebben.

Vraag 15 Een MouseAdapter is een:
a. Een klasse die niet in de standaard Java API zit.
b. Een interface die toelaat om te luisteren naar MouseEvents.
c. Een klasse die het makkelijk maakt om met MouseEvents om te gaan.
d. Geen van bovenstaande.

Vraag 16 Een grafische user interface maken kan door middel van het Model View Controller design pattern. Welke uitspraak is (het meest) correct:
a. Wat de gebruiker ziet (d.m.v. de controller) is altijd de meest recente data.
b. De gebruiker altijd in staat is om makkelijk de data in het model aan te passen d.m.v. de view.
c. De view en het model gescheiden zijn van elkaar, er meerdere views op die data kunnen bestaan.
d. **De view en het model gescheiden zijn van elkaar, er meerdere views op die data kunnen bestaan en sommige van die views aangepast kunnen worden door de gebruiker.**

Vraag 17 Om dit Model View Controller pattern te implementeren biedt java ondersteuning door middel van:
a. De interfaces Model en View.
b. De klassen Model en View.
c. De interfaces Model, View en Controller.
d. De interfaces Observer en Observable.

Vraag 18 Een attribuut static declareren houdt in dat:
a. De waarde van dit attribuut niet meer gewijzigd kan worden na initialisatie.
b. Dit attribuut gemeenschappelijk is voor alle instanties van de klasse.
c. Dit attribuut niet toegankelijk is voor subklassen.
d. Dit attribuut enkel wijzigbaar is door methoden van de klasse.

Vraag 19 Wat is het verschil tussen een error, checked exception en unchecked exception?
a. Een error is een fout van de gebruiker, een checked exception *moet* opgevangen worden en mag niet doorgegeven worden, terwijl een unchecked exception opgevangen mag worden of mag worden doorgegeven.
b. Een error is een fout van de gebruiker, een checked exception *moet* opgevangen of doorgegeven worden, terwijl een unchecked exception opgevangen mag worden of mag worden doorgegeven.
c. Een error is een systeemfout, een checked exception *moet* opgevangen worden en mag niet doorgegeven worden, terwijl een unchecked exception opgevangen mag worden of mag worden doorgegeven.
d. Een error is een systeemfout, een checked exception *moet* opgevangen of doorgegeven worden, terwijl een unchecked exception opgevangen mag worden of mag worden doorgegeven.

Vraag 20 Welke uitspraak is (het meest) waar?
a. Een BufferedReader is een soort Reader (extends Reader), maar is typisch efficienter omdat hij grotere stukken in een buffer inleest en dan beschikbaar stelt.

b. Een BufferedReader is een soort Reader en heeft ook een Reader als attribuut, maar is typisch efficienter omdat hij grotere stukken in een buffer inleest en dan beschikbaar stelt.
c. Een BufferedReader is een soort Reader en heeft ook een Reader als attribuut (in design pattern terminologie heeft dit een "Decorator"), maar is typisch efficienter omdat hij grotere stukken in een buffer inleest en dan beschikbaar stelt.
d. Een BufferedReader is een soort Reader (extends Reader) en heeft een Reader als attribuut (in design pattern terminologie heet dit een "Decorator"), maar is typisch efficienter omdat hij grotere stukken in een buffer inleest en dan beschikbaar stelt.

Vraag 21 Bekijk de code in bijlage A. Gegeven de code in bijlage A, wat is het gevolg van volgende 3 lijnen code?

```
Id multiplier = new Multiplier(7);
Functie f = multiplier.afgeleide();
double x = f.apply(2);
```

a. Een compile-time fout
b. Een run-time fout
c. Geen fout, x krijgt de waarde 2 na afloop.
d. Geen fout, x krijgt de waarde 14 na afloop.

Vraag 22 Bekijk de code in bijlage A. Gegeven de code in bijlage A, wat is het gevolg van volgende 3 lijnen code?

```
Functie functie  = new Multiplier(6);
Constant constant = (Constant)functie;
double x = constant.apply(2);
```

a. Een compile-time fout
b. Een run-time fout
c. Geen fout, x krijgt de waarde 12 na afloop.
d. Geen fout, x krijgt de waarde 2 na afloop.

Vraag 23 Dynamische binding. Welke uitspraak is fout?
a. Dynamische binding noemt men ook wel late binding.
b. Dynamische binding betekent dat de java virtuele machine beslist over welke methode wordt uitgevoerd.
c. Dynamische binding kan niet werken als er een abstracte klasse betrokken partij is.
d. Als software-ontwikkelaar kan je het gedrag van dynamische binding beinvloeden.

Vraag 24 Cyclomatische complexiteit. Welke uitspraak is fout?
a. De cyclomatische complexiteit drukt het aantal lineair onafhankelijke paden uit binnen een stuk code.
b. De cyclomatische complexiteit bereken je door het aantal beslissingspunten (condities) te tellen.
c. De cyclomatische complexiteit is een hulpmiddel bij het schrijven van testen.
d. De cyclomatische complexiteit is uitgevonden door McCabe en wordt dus soms ook wel McCabe complexiteit genoemd.

Vraag 25 Welke uitspraak is fout? Encapsulatie houdt in dat:
a. Je de interne details van de klasse (de implementatie) verbergt voor de buitenwereld.
b. Je de programmeur beschermt tegen zichzelf.
c. Je het moeilijker maakt om te testen.
d. Je het pass-by-value principe in Java mogelijk maakt.

Bijlage A

```
public abstract class Functie{
   public double apply(double x){
      return x;
   }
}

public class Id extends Functie{
}

public class Constant extends Functie{

   public Constant (double c){
      this.c = c;
   }

   public double apply(double x){
      return c;
   }

   private double c;
}

public class Multiplier extends Id{
   public Multiplier(double f){
      factor = f;
   }

   public double apply(double k){
      return k * factor;
   }

   public Functie afgeleide(){
      return new Constant(factor);
   }

   private double factor;
}

class Compose extends Functie{
    public Compose (Functie first, Functie second){
       this.first = first;
       this.second = second;
    }

    public double apply(double x){
       double y = first.apply(x);
       return second.apply(y);
    }

    private Functie first, second;
}
```