UNIVERSITEIT GENT

VAKGROEP INFORMATIETECHNOLOGIE
Sint Pietersnieuwstraat 41, B-9000 Gent, België • www.intec.UGent.be
Telefoon +32 9 2649970 • Fax +32 9 2649960

INTEC

EXAMEN SOFTWAREONTWIKKELING [II]

2DE BA. INFORMATICA
3DE BA. INGENIEURSWETENSCHAPPEN : COMPUTERWETENSCHAPPEN
EERSTE EXAMENPERIODE ACADEMIEJAAR 2008 – 2009

Vrijdag 5 juni, 2009, 8u30

Naam :

V1 :
V2 :
V3 :
V4 :
V5:
V6:
V7:
V8:
V9:

VRAAG 1: SOFTWARE LEVENSCYCLUS

a. Wat wordt bedoeld met de term “software proces” ?

b. Wat is het hoofddoel van de specificatie-fase in het ontwikkelproces ?

c. Geef de voornaamste voordelen van incrementele processen in vergelijking met het klassieke watervalproces.

VRAAG 2: OBJECT-GEORIENTEERDE ANALYSE

Gegeven onderstaande probleem:

"Een kwispakket laat toe om een aantal deelnemers, geïdentificeerd via voornaam, naam en uniek geheel getal, een reeks vragen te laten beantwoorden. Het pakket laat een administrator toe een reeks vragen te definiëren, en uit deze vragen een kwis samen te stellen. De deelnemers zelf beantwoorden alle vragen van een specifieke kwis, waarna hun score volgt. Een antwoord wordt geformuleerd in de vorm van schrifttekst. Elke vraag beschikt over een methode "evalueer", die voor een gegeven antwoord een geheel getal als score berekent. De uiteindelijke score per deelnemer bestaat dan uit de som van alle individuele vraagscores. Om fraude te vermijden, beschikt elke deelnemer over een loginnaam en paswoord. Een deelnemer kan zijn individuele score opvragen voor een welbepaalde kwis. De administrator kan alle scores van alle deelnemers bekijken."

a. Teken een zo nauwkeurig en informatief mogelijk use-case diagram.

b. Zet de tekst zo nauwkeurig mogelijk om in een UML klassendiagram.

VRAAG 3: OBJECT-GEORIENTEERD ONTWERP

a. Argumenteer het gebruik van gelaagde architecturen.

b. Wat bedoelt men met de term “partitioneren in subsystemen” ? Hoe gebeurt dit best (m.a.w. wat zijn de kenmerken van een goede partitionering ?) ?

c. Een UML-klassendiagram kan zowel tijdens de OOA- als tijdens de OOD-fase gemaakt worden. Leg het verschil tussen beide versies uit. Geef ook een **duidelijk** voorbeeld.

Verschillen:

Voorbeeld:

VRAAG 4: OBJECT-GEORIENTEERD PROGRAMMEREN

a. Wat zijn "coding standards" ? Geef enkele voorbeelden.

b. Gegeven onderstaande Java-code voor de methode f(int i, char c) :

```
public static String f(int i,char c) {
      String r="";
      if(i<=0) return r;
      while(i<3) {
            if(c>'A') r+=c;
            else r=c+r;
            i++;
      }
      return r;
}
```

Geef testvectorn <i,c> om takbedekking te realiseren ("branch coverage").

Geef testvectoren <i,c> om padbedekking te realiseren ("path coverage").

VRAAG 5 ONTWERPSPATRONEN

a. Composite-patroon.

(i) Leg uit welk fundamenteel probleem het composite-patroon oplost.

(ii) Geef de algemene structuur van dit patroon.

b. Webshop

Gegeven onderstaande Java skelet-code, die Users toelaat in een Shop zaken te kopen, na authenticatie, en daarna een rekening toegestuurd krijgen. De winkelwagen (shopping cart) wordt hier geïmplementeerd als lijst van Strings.

```
class User {
      protected String name;
      protected String password;
            public User(String n,String p) {
            name=n;
            password=p;
      }
      public boolean isAuthenticated() {
            // …
      }
      public void authenticate() {
            // …
```

```
        }
        public String getInput() {
            // ask input from user
            // "exit" implies stopping the application
            return input;
        }
        public void addToCart(String i) {
            // add item i to shopping cart
        }
        public ArrayList<String> getCart() {
            return cart;
        }
        public void sendBill(double p) {
            // bill received …
        }
}

class Shop {
        public void process(User u) {
            boolean exit=false;
            while(!(u.isAuthenticated())) {
                u.authenticate();
            }
            while(!exit) {
                String input=u.getInput();
                if(input.equals("exit")) exit=true;
                else u.addToCart(input);
            }
            ArrayList<String> cart=u.getCart();
            if(cart.size()==0) {
                return;
            } else {
                double price=calculatePrice(cart);
                u.sendBill(price);
            }
        }
        public double calculatePrice(ArrayList<String> c) {
            // …
        }

}
```

We wensen de code van de klasse Shop te her-ontwerpen (i.h.b. de process()-methode) via het State-patroon, waarbij de toestand van de Shop aangeeft in welk stadium de klant zich bevindt.

i) Teken een toestandsdiagram voor dit systeem.

ii) Geef een UML klassediagram voor dit herontworpen systeem.

iii) Geef de implementatie van de nieuwe **`process()`**-methode van de klasse Shop.

iv) Geef de implementatie van 1 toestandsklasse.

c. Beschouw onderstaande Java-code. In welke zin stelt deze code een probleem i.v.m. uitbreidbaarheir?

```
class A {
	protected int a;
	public A(int ia) {
		a=ia;
	}
	public A(A i){
		a=i.a;
	}
	public String toString() {
		return "A:"+a;
	}
}


class B {
	protected int b;
```

```
        protected A a;
        public B() {}
        public B(int ib,A ia) {
              b=ib;
              a=ia;
        }
        public B deepCopy() {
              B c=new B();
              c.b=b;
              c.a=new A(a);
              return c;
        }
        public String toString() {
              return "B:"+b+" : "+a;
        }
}
```

Hoe zou je dit probleem oplossen ? Teken een aangepast UML klassendiagram.

Geef de code-wijzigingen die je de klasse(n) A en/of B zou aanbrengen.

VRAAG 6: RAAMWERKEN EN INWENDIGE KLASSEN

NIET OP TE LOSSEN VOOR STUDENTEN BI.

a. Beschouw onderstaande Java-code, waarin een rij gehelegetallen in een klasse Series geencapsuleerd wordt.

```
class Series {
      protected int[] r;
      public Series(int s) {
            r=new int[s];
      }
      public void set(int index,int value) {
            r[index]=value;
      }
      public int get(int index) {
            return r[index];
      }
      public int getSum() {
            int sum=0;
            for(int i:r) sum+=i;
            return sum;
      }
      public int getLength() {
            return r.length;
      }
}
```

We wensen de toegang tot deze klasse voor sommige objecten te beperken tot de methoden getSum() en getLength(). Hiertoe gebruiken we het concept “inwendige klasse”.

Teken een UML-diagram van de principiële oplossing.

Geef de Java-implementatie van de aangepaste klasse Series.

b. Wat wordt bedoeld met de term “raamwerk” (“framework”) ?

VRAAG 7: MULTITHREADING

a. Wat is een “Race” ?

b. Leg de relatie uit tussen Java Executors en Threads. Motiveer het gebruik van Executors.

c. Beschouw onderstaande Java-code.

```
class Notifier extends Thread {
	public static final int MAX=10;
	private int goal;
	private Waiter waiter;
	public Notifier(int g,Waiter w) {
		goal=g;
		waiter=w;
		start();
	}
	public void run() {
		while(true) {
			int r=(int)(10.0*Math.random());
			System.out.print(" "+r);
			if(r==goal) {
				System.out.println("");
				waiter.setProceed(true);
				synchronized(waiter) {
					waiter.notifyAll();
				}
```

```
                    return;
                }
            }
        }
    }

class Waiter extends Thread {
    private boolean proceed=false;
    private int t;
    public Waiter() {
        start();
    }
    public synchronized void setProceed(boolean p){
        t++;
        proceed=(t==2);
    }
    public void run() {
        synchronized(this) {
            try{
                while(!proceed) wait();
            } catch(InterruptedException e) {
                System.err.println(e);
            }
        }
        System.out.println("Can proceed now !");
    }
}

public class TestMulti {
    public static void main(String[] args) {
        Waiter w=new Waiter();
        Notifier n1=new Notifier(7,w);
        Notifier n2=new Notifier(3,w);
    }
}
```

i) Leg uit waarom de methode **`setProceed()`** in de klasse **`Waiter`** al dan gesynchoniseerd moet zijn.

ii) Leg uit waarom voor de constructie

```
while(!proceed) wait();
```

gekozen werd, i.p.v. `if(!proceed) wait();`

VRAAG 8 : SERIALISATIE

a. Om een object te persisteren naar een bestand, kan met 2 mechanismen gebruiken: (i) het schrijven van alle datavelden naar het bestand, (ii) gebruik maken van objectserialisatie. Waarin verschillen deze opties fundamenteel ?

b. Beschouw onderstaande Java-code:

```
class A implements Serializable {
     private char c;
     public A(char cc) {
          c=cc;
     }
}


public class TestSimple {
 public static void main(String[] args) throws Exception {
     A[] a={new A('a'),new A('b'),new A('c')};
     ArrayList<A> l=new ArrayList<A>();
```

```
        l.add(new A('b'));
        l.add(a[1]);
        ObjectOutputStream out=new ObjectOutputStream(new
            BufferedOutputStream(new FileOutputStream("t.dat")));
        for(A o:a)
              out.writeObject(o);
        for(A o:l)
              out.writeObject(o);
        out.close();

        int n=0;
        ObjectInputStream in=new ObjectInputStream(new
             BufferedInputStream(new FileInputStream("t.dat")));
        ArrayList<A> i=new ArrayList<A>();
        try{
              while(true) {
                    A o=((A)(in.readObject()));
                    if(!(i.contains(o))) i.add(o);
                    n++;
              }
        } catch(EOFException e) {

        }
        in.close();
        System.out.println(n);
        System.out.println(i.size());
  }
}
```

i) Welke output genereert de regel : `System.out.println(n);` ? Verklaar.

Waarde :
Verklaring :

ii) Welke output genereert de regel : `System.out.println(i.size());` ? Verklaar.

Waarde :

Verklaring :

VRAAG 9 : COMPONENTEN

a. Motiveer het gebruik van componenten.

b. We wensen een JavaBeans component te realiseren, die over 2 bound properties beschikt, namelijk de properties a en b, beide van type int. We wensen ook dat luisteraars zich kunnen registreren om:

- Op de hoogte gebracht te worden van wijzigingen van property a (en niet van b)
- Idem voor wijzigingen van property b (en niet van a)
- Idem voor wijzigingen van beide properties.

Teken een zo volledig mogelijk UML klassendiagram van een systeem dat dit realiseert, zoveel mogelijk gebruik makend van de door Java geboden functionaliteit.