Software-ontwikkeling 1
Software-ontwikkeling en object-georiënteerde talen 1
Examen van 31 januari 2006

Ten geleide

Dit examen omvat drie vragen :

- Vraag 1 is de hoofdvraag, waar wellicht de meeste tijd in kruipt
- Bij vraag 2 kan het antwoord erg kort zijn. Als je het niet onmiddellijk ziet, hou ze dan voor het laatste ('t is de "onderscheidingsvraag").
- Verlies ook geen onnodige tijd bij vraag 3 . Een conceptuele oplossing brengt ook punten op, en de examinator is geen compiler ...

De typografische conventies in wat volgt zijn :
- "mensentaal" staat in 12-punt Times New Roman (dit lettertype dus)
- C++-uitdrukkingen, taalwoorden, programmafragmenten staan in `10-punt Courier New`.

Je mag de vragen beantwoorden in de volgorde die je wenst. Lees wel telkens eerst de volledige vraagstelling om verrassingen te voorkomen : de beschikbare tijd kan een kritieke factor worden bij dit examen, maar is voor iedereen dezelfde Bezin dus voor je begint.

Gelieve de hoeveelheid papier die je produceert te beperken tot het essentiële.

Veel succes.

HT

Vraag 1 – De reisorganisatie

Wat volgt is een benaderde omschrijving van een probleemdomein, waarvoor een informatiesysteem moet ontwikkeld worden.

Een reisorganisatie organiseert en verkoopt (uiteraard) reizen.

- Ze doet dit zowel voor leden als voor toevallige klanten. Leden krijgen op geregelde basis brochures, toevallige klanten nemen zelf het initiatief om naar de organsisatie toe te stappen. Leden hebben bovendien een lidnummer en krijgen een korting.
- Zowel leden als klanten hebben een naam, een adres, en eventueel een BTW nummer (als het over zakenreizen gaat – zie verder)
- De reisorganisatie beschikt over een netwerk van kantoren. Elk kantoor heeft een adres.
- Leden ressorteren onder een bepaald kantoor, en kunnen dus ook als toevallige klant naar een ander kantoor stappen. In dat geval kunnen ze wel op de gebruikelijke korting voor leden rekenen.
- De reisorganisatie organiseert zowel privéreizen als zakenreizen. Bij een privéreis wordt een gewone rekening gemaakt en worden de reisdocumenten

eerst opgemaakt na betaling. Bij een zakenreis wordt een factuur bezorgd aan het bedrijf van de reiziger (met BTW nummer).

- De reisorganisatie heeft contracten met touroperatoren die voorverpakte reizen aanbieden. Als ze zulk een reis doorverkoopt treedt ze louter op als tussenpersoon en ontvangt ze daarvoor een commissie
- Reizen worden ook verkocht op individuele basis. In dat geval bestaat een reis uit transport (eigen wagen, vliegtuig, huurauto, schip, ...), verblijf (camping, appartement, villa, hotel – al dan niet met maaltijden) en eventuele extras (bvb. museumcheques).

Ontwerp de nodige C++ klassen die dit kunnen implementeren, in het bijzonder hun attributen en relaties. Doe dit op een zo hoog mogelijk niveau van abstractie en/of genericiteit. Verantwoord uw ontwerpsbeslissingen. Een grafische voorstelling van je ontwerp kan zeer verhelderend zijn, maar hoeft niet. Als je geen UML notatie gebruikt, geef dan een verklaring van de grafische symbolen.

Enkel de attributen van de klassen, die voortvloeien uit bovenstaande gegevens, moeten aangegeven worden. Lidfuncties, andere dan constructoren en destructoren, moeten niet vermeld, noch geïmplementeerd worden. Van de constructoren en destructoren (in zoverre nodig) wordt er wel een implementatie verwacht.

Hulpklassen die beschikbaar zijn in standaard C++ , inclusief STL, mogen (moeten indien nodig !) uiteraard gebruikt worden.

Let wel : er is geen eenduidig antwoord op deze vraag. Bespreek dus je ontwerpsbeslissingen. Probeer er vooral geen dingen bij te fantaseren. Het is vooral van belang dat je aantoont hoe je zulk een probleem aanpakt.

Vraag 2 - ontwerp van klasse-hierarchieën

Beschouw de volgende klassedefinities.

```
class Punt {
      // ... (geen lidfunctie met naam Opp() !!)
      };    // Een punt op het scherm

class Figuur {
      Punt  positie ;
public :
      virtual double Opp() = 0 ; // bereken de oppervlakte
      // ... en het gebruikelijke
      } ;
```

Merk op dat de klasse `Punt` niet kan overerven van de klasse `Figuur` (zie verder bij de vraagstelling).

```
class Cirkel : public Figuur {
      // ... het gebruikelijke
public :
      double Opp () { /* bereken de oppervlakte van een cirkel) */ }
      // ... en het gebruikelijke
      } ;

class Vierkant : public Figuur {
      // ... het gebruikelijke
```

```
public :
      double Opp () { /* bereken de oppervlakte van een vierkant) */ }
      // ... en het gebruikelijke
      } ;
```

In codefragment 1 houden we een lijst van figuren bij en berekenen we de totale oppervlakte. Daar is niets mis mee.

Fragment 1

```
Figuur * lijst[100] ;
int n = 0 ;
double TotaleOppervlakte = 0. ;
lijst[n++] = new Cirkel(/*...*/) ;
lijst[n++] = new Vierkant(/*...*/) ;
for (int i = 0; i < n; i++)
      TotaleOppervlakte += lijst[i]->Opp() ;
```

Achteraf ontdekken we dat een `Punt` veel gemeen heeft met een `Figuur`. Het kan bijvoorbeeld ook op scherm getekend worden. We zouden dus graag ook fragment 2 als mogelijkheid hebben (de wijzigingen t.o.v. fragment 1 staan vetjes).

Fragment 2

```
Figuur * lijst[100] ;
int n = 0 ;
double TotaleOppervlakte = 0. ;
lijst[n++] = new Cirkel(/*...*/) ;
lijst[n++] = new Vierkant(/*...*/) ;
lijst[n++] = new Punt(/*...*/) ;    // mis
for (int i = 0; i < n; i++)
      TotaleOppervlakte += lijst[i]->Opp() ;    // mis
```

Fragment 2 loopt mis op twee punten, namelijk in de twee aangeduide lijnen.

Vragen

a. Waarom kan `Punt` niet overerven van `Figuur`

b. Wat loopt er precies mis in fragment 2.

c. Wat kan eraan gedaan worden en hoe moet de klassestructuur (en eventueel fragment 2) aangepast worden opdat het (d.i. fragment 2) toch zou lukken.

Vraag 3 – De verjaardagskalender

De verjaardagskalender houdt per dag van het jaar bij wie er die dag verjaart. Per dag kunnen er meerdere personen verjaren. Je mag aannemen dat er elk jaar 365 dagen zijn en dus abstractie maken van het bestaan van schrikkeljaren. Er kunnen ook dagen zijn dat er niemand verjaart.
Personen worden geidentificeerd door hun naam (voor de eenvoud een string).

Geef een implementatie die toelaat om :

- Een lege kalender te initialiseren (een constructor ?)
- Een bestaande kalender te vernietigen (een destructor ?)

- Een persoon toe te voegen aan de kalender, liefst op de juiste datum
- Voor een welbepaalde dag een lijst te geven van de personen die dan verjaren.

Je mag dit naar keuze doen in C of C++. Je hoeft geen compleet Datum datatype uit te werken (dat mag, als je tijd genoeg hebt).