Problema 1: Especificació i implementació de la classe `Date` (Solució)

Josep M. Ribó

4 d'octubre de 2006

1 Objectius

- Comprendre què és una classe i les passes de definició de la mateixa (especificació-representació-implementació)
- Aprendre a usar una classe. En particular, a crear objectes d'una classe i aplicar operacions de la classe sobre ells

Per fer aquest problema, cal que us llegiu:
- Els apunts (apartats 1.1-1.5)
- El document titulat: *Manaments de disseny de classes*

A cada apartat del problema trobareu entre parèntesi i en negreta els apartats dels apunts on s'explica la teoria que necessiteu per resoldre'l.

2 Especificació de la classe `Date`

La classe `Date` està dissenyada per tal de modelitzar dates de l'estil 1-2-2003 a partir de l'any 1900.

1. En una classe apareixen tres famílies d'operacions. Quines són? **(APUNTS: 1.3, 1.4.3)**
   Operacions constructores, modificadores i consultores
2. En el cas de la classe `Date`: Quines operacions de cadascuna de les famílies anteriors ha d'oferir?
   - **Constructora:**
     ```
     bool createDate(int pday, int pmonth, int pyear);
     ```
   - **Modificadores:**
     ```
     bool setDay(int pday);
     bool setMonth(int pmonth);
     bool setYear(int pyear);
     void copy(Date d);
     void addDays(int ndays);
     ```
   - **Consultores:**

```
unsigned int getDay();
unsigned int getMonth();
unsigned int getYear();
bool equals(Date d);
unsigned int daysinBetween(Date d);
```

3. Què significa *especificar una classe*? **(APUNTS: 1.2.2, 1.4, 1.5.5)**

   Essencialment significa proposar una precondició i una postcondició per cadascuna de les operacions de la classe.

4. Per què a l'especificació d'una classe se l'anomena *contracte*? Entre quines dues parts s'estableix aquest contracte? A què es compromet cadascuna? **(APUNTS: 1.4.1)**

   L'especificació d'una classe es pot veure com un contracte entre l'usuari de la classe i el seu implementador. L'usuari es compromet a fer servir les operacions públiques de la classe de la manera indicada per l'especificació i l'implementador a implementar-les respectant escrupulosament la mateixa especificació.

5. En quin fitxer posaràs aquesta especificació? **(APUNTS: 1.5.4, 1.5.5)**

   Al fitxer `Date.txt` *o* `Date.html` *o* `Date.pdf`.

   En tot cas, es tracta d'un fitxer de text, no de codi.

6. Després d'haver contestat les preguntes anteriors, ara ja pots especificar la classe `Date`

   Fitxer: `Date.txt`**:**

   - `bool createDate(int pday, int pmonth, int pyear)` *(creador)*
     - **Crida:** `er=dat.createDate(pday,pmonth,pyear);`
     - **Pre:** -
     - **Post:** `dat` conté la data `pday-pmonth-pyear` i `er=false`.
       Si `pday-pmonth-pyear` no és una data correcta posterior a `1-01-1900`, `dat` és `1-01-1900` i `er=true`
   - `int getDay()` *(consultor)*
     - **Crida:** `d=dat.getDay();`
     - **Pre:** `dat` és una data correctament creada
     - **Post:** `d` és el dia de la data `dat`. `dat=dat'`.
   - `bool setDay(int pday)` *(modifier)*
     - **Crida:** `er=dat.setDay(pday);`
     - **Pre:** `dat` és una data correctament creada
     - **Post:** `dat` té el mateix mes i any que `dat'`. `pday` és el nou dia de la data `dat` i `er=fals`.
       Si la data `pday-dat.getMonth()-dat.getYear()` no és correcta, `dat=dat'` i `er=true`
   - `void copy(Date d);` *(modificador)*
     - **Crida:** `dat.copy(dat2);`
     - **Pre:** -
     - **Post:** `dat` obté el valor de `dat2` (`dat=dat2`)
   - `bool equals(Date d);` *(consultor)*
     - **Crida:** `b=dat.equals(dat2);`
     - **Pre:** `dat` és una data correctament creada
     - **Post:** `b` és cert si `dat` té el mateix valor de `dat2`. `dat` no ha sigut modificat

- `void addDays(unsigned int ndays)` *(modificador)*
  - **Crida:** `dat.addDays(ndays);`
  - **Pre:** `ndays` $\geq$ `0` i `dat` és una data correctament creada
  - **Post:** `dat` és la data `dat'` a la que s'ha afegit `ndays` dies.

3 Representació de la classe `Date`

1. Què significa *representar una classe*? **(APUNTS: 1.2.2, 1.5)**

   Representar una classe significa triar els atributs que permetran emmagatzemar internament (i.e., a la memòria de la màquina) els objectes de la classe que s'està representant.

   Aquests atributs solen ser de tipus predefinits proporcionats pel llenguatge de programació (int, char, char[]...) o bé d'altres classes dissenyades prèviament (vegeu problema 2).

   La representació d'una classe no pot ser vista, sota cap circumstància, per l'usuari de la classe.

2. En quin fitxer es posa la representació de la classe `Date`? **(APUNTS: 1.5.4, 1.5.5)**

   Al fitxer `Date.h`

3. Quina etiqueta es posa abans de la definició dels elements que constitueixen la representació de la classe? **(APUNTS: 1.5.2)**

   L'etiqueta és `private:`

   Si no es posa aquesta etiqueta no passa res perquè, per defecte, els membres (atributs i operacions) d'una classe són privats. Però compte!! En aquest cas, els membres privats han de ser els primers de la definició de la classe. No poden aparéixer després dels membres delcarats com a `public:`

4. Com es diuen els elements que usem per representar una classe? **(APUNTS: 1.2.2, 1.5.2)**

   Els membres que usem per representar una classe s'anomenen atributs.

5. El fitxer `Date.h` conté alguns elements que no formen part pròpiament de la representació de la classe. Quins són? **(APUNTS: 1.5.4)**

   A més a més dels atributs privats que serveixen per representar una classe, el fitxer `Date.h` *contindrà també la capcelera de les operacions públiques de la classe. O sigui, aquelles operacions que poden ser utilitzades per l'usuari de la classe i que han sigut especificades al fitxer* `Date.txt`*. En tot cas, aquestes operacions no formen part de la representació de la classe.*

6. La precondició i la postcondició de cada operació (i.e., el que hem anomenat *l'especificació de la classe* i que hem completat a la secció anterior), poden contenir atributs de la representació de la classe? **(APUNTS: 1.4.2, 1.4.3)**

   No, no i no i mil vegades no. És un greu error incloure aspectes de la representació de la classe (i, per tant, privats) en la seva especificació. Un dels pilars de la POO és la separació entre l'especificació i la representació-implementació d'una classe. L'usuari simplement no pot accedir a aquests darrers. D'aquesta manera:

   - *No s'ha de preocupar per entendre com estan representats internament els objectes de les classes que utilitza*
   - *No ha de modificar el codi dels seus programes si, en un futur, aquestes classes canvien de representació per fer-la més eficient.*

7. Després de contestar les preguntes anteriors, ara ja pots representar la classe `Date`

   Fitxer `Date.h`:

```
#ifndef DATA_H
#define DATA_H

#include <iostream>

using namespace std;

class Date{
private:
  unsigned int day;
  unsigned int month;
  unsigned int year;

  bool correctDate(unsigned int pday, unsigned int pmonth,
                   unsigned int pyear);
  int leap(unsigned int y);
public:
  bool createDate(unsigned int pday, unsigned int pmonth,
                  unsigned int pyear);

  bool setDay(unsigned int pday);
  bool setMonth(unsigned int pmonth);
  bool setYear(unsigned int pyear);

  unsigned int getDay() const;
  unsigned int getMonth() const;
  unsigned int getYear() const;

  void copy(Date d);
  bool equals(Date d) const;

  void addDays(unsigned int ndays);
};
#endif
```

4 Implementació de les operacions de la classe `Date`

1. En què consisteix la implementació de les operacions d'una classe? **(APUNTS: 1.4.1, 1.5.4, 1.5.5)**

   La implementació de les operacions d'una classe consisteix en implementar cadascuna de les operacions públiques de la classe:

   - *Usant la representació decidida a l'apartat anterior*
   - *Respectant escrupulosament l'especificació de la classe*

2. En quin fitxer has de posar la implementació de les operacions de la classe? **(APUNTS: 1.5.4)**

   Al fitxer `Date.cc`

3. Quin fitxer has d'incloure per tal de poder implementar les operacions de la classe `Date`? **(APUNTS: 1.5.4)**

   ```
   #include "Date.h"
   ```

4. Què ha de tenir especialment en compte l'implementador? **(APUNTS: 1.4.1, 1.5.5)**

   Ser fidel a l'especificació de cada operació que implementi.

5. Implementa les operacions de la classe `Date`

Fitxer *Date.cpp*

```
#include <iostream>
#include "Date.h"

using namespace std;

int Date::leap(unsigned int pyear)
{
  if (pyear%4==0 && pyear%100!=0) return 1;
  else return 0;
}

bool Date::correctDate(unsigned int pday,
                       unsigned int pmonth,
                       unsigned int pyear)
{
  unsigned int max;

  if (pmonth==1 || pmonth==3 || pmonth==5 ||
      pmonth==7 || pmonth==8 || pmonth==10 ||
      pmonth==12){ max=31;}
  else if (pmonth==2){ max=28+leap(pyear);}
  else max=30;

  return (pday<=max && pmonth<=12 && pyear>=1900) ;
}

bool Date::createDate(unsigned int pday,
                      unsigned int pmonth,
                      unsigned int pyear)
{
  bool error;

  if (correctDate(pday, pmonth, pyear)){
    error=false;
    day=pday;
    month=pmonth;
    year=pyear;
  }
  else{
    error=true;
    day=1;
    month=1;
    year=1900;
 }

  return error;
}

bool Date::setDay(unsigned int pday)
{
  bool err;

  if (correctDate(pday,month,year)){
    day=pday;
    err=false;
```

```
  }
  else err=true;

  return err;
}

bool Date::setMonth(unsigned int pmonth)
{
  bool err;
  if (correctDate(day,pmonth,year)){
    month=pmonth;
    err=false;
  }
  else err=true;

  return err;
}

bool Date::setYear(unsigned int pyear)
{
  bool err;
    if (correctDate(day,month,pyear)){
      year=pyear;
      err=false;
    }
    else err=true;
    return err;
}

unsigned int Date::getDay() const
{
 return day;
}

unsigned int Date::getMonth() const
{
  return month;
}

unsigned int  Date::getYear() const
{
 return year;
}

void Date::copy(Date d)
{
  day=d.day;
  month=d.month;
  year=d.year;
}

bool Date::equals(Date d) const
{
  return (day==d.day && year==d.year&& month==d.month);
}

void Date::addDays(unsigned int ndays)
{
```

```
  //....
}
```

Nota important: Sovint és útil definir operacions privades per tal de simplificar la implementació de les operacions públiques. Dues situacions típiques són les següents:

- S'implementa en una operació privada una funcionalitat que es necessita en la implementació de diverses operacions públiques (és el cas de `correctDate`, que es crida des de diferents operacions)
- S'implementa en una operació privada una funcionalitat no trivial, que requereix unes quantes lí nies de codi i pensar-hi una mica. També és el cas de `correctDate`.

Les operacions privades no poden ser vistes ni usades per l'usuari de la classe.

5 Ús de la classe `Date`

1. En quin fitxer has de posar un programa que usi la classe `Date`? **(APUNTS: 1.5.4, 1.5.5)**

   Al fitxer `usuari.cc`. *El nom, importa poc. Simplement ha de ser un fitxer de codi C++ diferent de* `Date.cc`.

2. Quin fitxer has d'incloure per tal de poder usar la classe `Date`? **(APUNTS: 1.5.4)**

   ```
   #include "Date.h"
   ```

3. Quins membres de la classe `Date` podran ser accedits pel programa usuari d'aquesta classe? **(APUNTS: 1.2.2, 1.5.2, 1.5.5)**

   Només els membres públics de la classe. Mai els privats.

4. Com es declaren els objectes de la classe `Date`? I com s'aplica una operació sobre ells? **APUNTS: 1.6.1**

   - *Declaració d'un objecte:*
     `Date d;`
   - *Crida d'una operació sobre un objecte:*
     `d.createDate(10,10,1980);`

5. Escriu un petit programa que usi la classe `Date`

   Fitxer: `usuari.cc`

   ```
   #include "Date.h"
   #include <iostream>

   using namespace std;

   int main()
   {
     bool err;
     unsigned int d,m,y;
     Date today;

     cin>>d>>m>>y;
     err=today.createDate(d,m,y);

     if (!err)  cout<<today.getDay()<<"-"<<today.getMonth()<<"-"
   ```

```
                <<today.getYear()<<endl;
  else cout<<"error"<<endl;
  return 0;
}
```

6. Què cal fer per tal d'executar aquest programa? **(APUNTS: 1.5.4)**

```
$ g++ -c Data.cc
$ g++ -c usuari.cc
$ g++ usuari.o Data.o -o usuari
```