Laboratoire

Unité : PRG1	Labo no : 06	Class *Date* et *Personne*

But

Nous désirons créer deux classes *Date* et *Person* qui possèdent, entre autres, les propriétés suivantes :

Date	***Person***

`unsigned day,`
`month,`
`year;`
`bool correct = true;`
	`const std::string lastName;`
	`const std::string firstName;`
	`const Date date;`
	`const unsigned noId;`

Le code principal, qui ne doit en aucun cas être modifié, ainsi que les résultats attendus sont fournis.

Ecrire les classes nécessaires dans quatre fichiers distincts

- `Date.h` et `Date.cpp`
- `Person.h` et `Person.cpp`

Compléments

- Beaucoup de réponses se trouvent dans l'output du programme ;)

Class Date

- Pour faciliter la mise à jour, les éléments linguistiques seront positionnés dans le fichier .h
- Pour les dates erronées (faute d'exception)
  - Les conversions en string, affichage etc.. prévoir « invalide date »
  - Les opérateurs mathématiques ne modifient pas et/ou retourne la même date
  - Les comparaisons avec une ou deux dates erronées retournent *false*
  - Les setters peuvent rendre une date valide ou invalide

Class Person

- Les membres listés ci-dessus sont impérativement constants
- La propriété *noId* est unique pour une personne et est copiée comme telle
- La propriété *nbrePerson* somme le nombre d'objets actifs

Contraintes

- Ne rien modifier dans le programme principal fourni
- Ne rien utiliser qui n'est pas encore étudié en théorie (i.e. exception)
- Factoriser autant que possible le code

```cpp
#include <cstdlib>
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include "Date.h"
#include "Person.h"

using namespace std;

//-------------------------------------------------------------
// collection of persons
using  Book = vector<Person>;
ostream& operator<< (ostream& os, const Book& book);
void show(const Book& book, const PERSON& by, const string& str);

//-------------------------------------------------------------
int main() {

   cout << boolalpha;   // for comparison operators

   cout << "----------------------------------------"                  << endl;
   cout << "   test of class Date"                                     << endl;
   cout << "----------------------------------------"                  << endl;

   cout << "constructors"                                              << endl;
   cout << "Date()                    : " << Date()                    << endl;
   cout << "Date(\"11-03-2019\")        : " << Date("11-03-2019")      << endl;
   Date date(27, 8, 1991);
   cout << "Date(27, 8, 1991)         : " << Date(27, 8, 1991)         << endl;

   cout << "Date(date)                : " << Date(date)                << endl;

   cout << "----------------------------------------"                  << endl
        << "getters and setters"                                       << endl;
   date.setDay(18);
   cout << "setDay(unsigned)          : " << date                      << endl;

   date.setMonth(6);
   cout << "setMonth(unsigned)        : " << date                      << endl;

   date.setMonth("September");
   cout << "setMonth(string)          : " << date                      << endl;

   date.setMonth(Month::MARCH);
   cout << "setMonth(Month)           : " << date                      << endl;

   cout << "date.getDay()             : " << date.getDay()             << endl;
   cout << "date.getMonthNo()         : " << date.getMonthNo()         << endl;
   cout << "(int)date.getMonthEnum()  : " << (int)date.getMonthEnum()  << endl;
   cout << "date.getMonthString()     : " << date.getMonthString()     << endl;
   cout << "date.getYear()            : " << date.getYear()            << endl;
```

```
Date d1(1, 1, 2010);
Date d2(2, 1, 2010);
cout << "---------------------------------------"                  << endl
     << "comparison operators"                                     << endl
     << d1 << " <  " << d2 << " ? " << (d1 <  d2)                  << endl
     << d2 << " <  " << d1 << " ? " << (d2 <  d1)                  << endl
     << d1 << " <= " << d2 << " ? " << (d1 <= d2)                  << endl
     << d1 << " >= " << d2 << " ? " << (d1 >= d2)                  << endl
     << d1 << " == " << d2 << " ? " << (d1 == d2)                  << endl;
--d2;
cout << d1 << " == " << d2 << " ? " << (d1 == d2)                  << endl
     << d1 << " <  " << d2 << " ? " << (d1 <  d2)                  << endl
     << d1 << " >  " << d2 << " ? " << (d1 >  d2)                  << endl
     << d1 << " <= " << d2 << " ? " << (d1 <= d2)                  << endl
     << d1 << " >= " << d2 << " ? " << (d1 >= d2)                  << endl;

cout << "---------------------------------------"                  << endl
     << "compound assignment operators"                            << endl
     << "d1 : " << d1 << endl;
cout << "--d1  : " << d1     << " => " << (--d1)                   << endl;
cout << "++d1  : " << d1     << " => " << (++d1)                   << endl;
cout << "d1--  : " << (d1--) << " => " << d1                       << endl;
cout << "d1++  : " << (d1++) << " => " << d1                       << endl;

cout << "d2 : " << d2 << endl;
cout << "d2-=2 : " << d2 << " => " << (d2-=2)                      << endl;
cout << "d2+=3 : " << d2 << " => " << (d2+=3)                      << endl;

cout << "---------------------------------------"                  << endl
     << "arithmetic operators"                                     << endl;
Date d3(1, 3, 2015);
cout << d3 << " + 10 = " << (d3 + 10)                              << endl;
cout << d3 << " - 42 = " << (d3 - 42)                              << endl;
cout << "10 + " << d3 << " = " << (10 + d3)                        << endl;

cout << "---------------------------------------"                  << endl
     << "assignment operator"                                      << endl;
Date d4;
d4 = d3;
cout << "d4 = d3                    : " << d4                      << endl;

d4 = {11, 03, 1978};
cout << "d4 = {11, 03, 1978}        : " << d4                      << endl;

Date d5;
d5 = {2, 3, 2020};
cout << "string(d5)                 : "  << string(d5)           << endl;
```

```
cout << "----------------------------------------"                    << endl
     << "wrong dates"                                                 << endl;
// in case a Date is wrong, then :
// - when converted to string, displayed, etc => "Invalide Date"
// - when treated with an arithmetic operators => unchanged
// - when treated with an comparison operators => alway false
// - setter are effective and may correct the date to a valid status
Date d6(31, 6, 2020);
cout << "d6(31, 6, 2020)           : " << d6                          << endl;
d6.setDay(30);
cout << "d6(30, 6, 2020)           : " << d6                          << endl;
d6.setMonth(Month::FEBRUARY);
cout << "d6(30, 2, 2020)           : " << d6                          << endl;

cout << "d6.isValid()              : " << d6.isValid()                << endl;
cout << "isValid(31, 6, 2020)      : " << Date::isValid(31, 6, 2020)<< endl;

cout << "----------------------------------------"                    << endl
     << "LeapYear"                                                    << endl;
cout << "d6.isLeapYear()           : " << d6.isLeapYear()             << endl;
cout << "isLeapYear(2020)          : " << Date::isLeapYear(2020)      << endl;

cout << "----------------------------------------"                    << endl
     << "numberDaysInMonth"                                           << endl;
cout << "d6.numberDaysInMonth()    : " << d6.numberDaysInMonth()      << endl;
cout << "numberDaysInMonth(2, 2020) : " << Date::numberDaysInMonth(2, 2020)
     << endl;

cout << endl;
cout << "----------------------------------------"                    << endl;
cout << "   class Person"                                             << endl;
cout << "----------------------------------------"                    << endl;
Person p1("Anna", "Conda", Date(12, 3, 2007));
cout << p1 << endl;
cout << "person counter : " << Person::nbrePerson()                   << endl;

Person p2(p1);
cout << p2 << endl;
cout << "person counter : " << Person::nbrePerson()                   << endl;

p2 = p1;
cout << p2 << endl;
cout << "person counter : " << Person::nbrePerson()                   << endl;
```

```
//-------------------------------------------------------------
// vector of Person
//-------------------------------------------------------------
cout << endl;
cout << "---------------------------------------"                    << endl
     << "3 new persones"                                            << endl;
Person Marilyn  ( "Monroe",  "Marilyn", Date( 1, 6, 1926) );
Person Elvis    ( "Presley", "Elvis",   Date( 8, 1, 1935) );
Person Michael  ( "Jackson", "Michael", Date(29, 8, 1958) );
cout << "nbre personnes : " << Person::nbrePerson()                 << endl;

{
   cout << endl;
   cout << "---------------------------------------"                 << endl
        << "a vector of 3 persons"                                  << endl;
   Book book;
   book.push_back(Marilyn);
   book.push_back(Elvis);
   book.push_back(Michael);
   cout << "person counter : " << Person::nbrePerson()              << endl;

   cout << endl;
   cout << "sorted by noId" << endl;
   sort(book.begin(), book.end(), SortBy(PERSON::NO_ID));
   cout << book << endl;

   cout << "sorted by lastName" << endl;
   sort(book.begin(), book.end(), SortBy(PERSON::LASTNAME));
   cout << book << endl;

   cout << "sorted by firstName" << endl;
   sort(book.begin(), book.end(), SortBy(PERSON::FIRSTNAME));
   cout << book << endl;

   cout << "sorted by date" << endl;
   sort(book.begin(), book.end(), SortBy(PERSON::DATE));
   cout << book << endl;

   cout << "---------------------------------------"                 << endl
        << "find a person by its no_id"                             << endl;
   show(book, PERSON::NO_ID, "2");
   cout << endl;

   cout << "---------------------------------------"                 << endl
        << "find a person by its firstname"                         << endl;
   show(book, PERSON::LASTNAME, "Presley");
   cout << endl;

   cout << "---------------------------------------"                 << endl
        << "find a person by its lastname"                          << endl;
   show(book, PERSON::FIRSTNAME, "Marilyn");
   cout << endl;

   cout << "---------------------------------------"                 << endl
        << "find a person by its date"                              << endl;
   show(book, PERSON::DATE, "29-08-1958");
   cout << endl;
}
```

```
    cout << endl;
    cout << "person counter : " << Person::nbrePerson()                    << endl;

    return EXIT_SUCCESS;
}

//-------------------------------------------------------------
ostream& operator<< (ostream& os, const Book& book) {
    for (const Person& p : book)
        os << p << endl;
    return os;
}

//-------------------------------------------------------------
void show(const Book& book, const PERSON& by, const string& str) {
    Book::const_iterator it = find_if(book.begin(), book.end(), FindBy(by, str));
    if (it != book.end())
        cout << *it;
    else
        cout << "/!\\ - not found";
}
```

```
//-------------------------------------------------------------
// program output
//-------------------------------------------------------------
//               ----------------------------------------
//                  test of class Date
//               ----------------------------------------
//               constructors
//               Date()                      : 01-01-1900
//               Date("11-03-2019")          : 11-03-2019
//               Date(27, 8, 1991)           : 27-08-1991
//               Date(date)                  : 27-08-1991
//               ----------------------------------------
//               getters and setters
//               setDay(unsigned)            : 18-08-1991
//               setMonth(unsigned)          : 18-06-1991
//               setMonth(string)            : 18-09-1991
//               setMonth(Month)             : 18-03-1991
//               date.getDay()               : 18
//               date.getMonthNo()           : 3
//               date.getMonthEnum()         : 3
//               date.getMonthString()       : March
//               date.getYear()              : 1991
//               ----------------------------------------
//               comparison operators
//               01-01-2010 <  02-01-2010 ? true
//               02-01-2010 <  01-01-2010 ? false
//               01-01-2010 <= 02-01-2010 ? true
//               01-01-2010 >= 02-01-2010 ? false
//               01-01-2010 == 02-01-2010 ? false
//               01-01-2010 == 01-01-2010 ? true
//               01-01-2010 <  01-01-2010 ? false
//               01-01-2010 >  01-01-2010 ? false
//               01-01-2010 <= 01-01-2010 ? true
//               01-01-2010 >= 01-01-2010 ? true
//               ----------------------------------------
//               compound assignment operators
//               d1 : 01-01-2010
//               --d1  : 01-01-2010 => 31-12-2009
//               ++d1  : 31-12-2009 => 01-01-2010
//               d1--  : 01-01-2010 => 31-12-2009
//               d1++  : 31-12-2009 => 01-01-2010
//               d2 : 01-01-2010
//               d2-=2 : 01-01-2010 => 30-12-2009
//               d2+=3 : 30-12-2009 => 02-01-2010
//               ----------------------------------------
//               arithmetic operators
//               01-03-2015 + 10 = 11-03-2015
//               01-03-2015 - 42 = 18-01-2015
//               10 + 01-03-2015 = 11-03-2015
//               ----------------------------------------
//               assignment operator
//               d4 = d3                     : 01-03-2015
//               d4 = {11, 03, 1978}         : 11-03-1978
//               string(d5)                  : 02-03-2020
//               ----------------------------------------
//               wrong dates
//               d6(31, 6, 2020)             : Invalide Date
//               d6(30, 6, 2020)             : 30-06-2020
//               d6(30, 2, 2020)             : Invalide Date
//               d6.isValid()                : false
//               isValid(31, 6, 2020)        : false
```

```
//              ---------------------------------------
//              LeapYear
//              d6.isLeapYear()            : true
//              isLeapYear(2020)           : true
//              ---------------------------------------
//              numberDaysInMonth
//              d6.numberDaysInMonth()     : 29
//              numberDaysInMonth(2, 2020) : 29
//
//              ---------------------------------------
//                 class Person
//              ---------------------------------------
//              Anna Conda   12-03-2007 (id=1)
//              person counter : 1
//              Anna Conda   12-03-2007 (id=1)
//              person counter : 2
//              Anna Conda   12-03-2007 (id=1)
//              person counter : 2
//
//              ---------------------------------------
//              3 new persones
//              nbre personnes : 5
//
//              ---------------------------------------
//              a vector of 3 persons
//              person counter : 8
//
//              sorted by noId
//              Monroe Marilyn   01-06-1926 (id=2)
//              Presley Elvis    08-01-1935 (id=3)
//              Jackson Michael  29-08-1958 (id=4)
//
//              sorted by lastName
//              Jackson Michael  29-08-1958 (id=4)
//              Monroe Marilyn   01-06-1926 (id=2)
//              Presley Elvis    08-01-1935 (id=3)
//
//              sorted by firstName
//              Presley Elvis    08-01-1935 (id=3)
//              Monroe Marilyn   01-06-1926 (id=2)
//              Jackson Michael  29-08-1958 (id=4)
//
//              sorted by date
//              Monroe Marilyn   01-06-1926 (id=2)
//              Presley Elvis    08-01-1935 (id=3)
//              Jackson Michael  29-08-1958 (id=4)
//
//              ---------------------------------------
//              find a person by its no_id
//              Monroe Marilyn   01-06-1926 (id=2)
//              ---------------------------------------
//              find a person by its firstname
//              Presley Elvis   08-01-1935 (id=3)
//              ---------------------------------------
//              find a person by its lastname
//              Monroe Marilyn   01-06-1926 (id=2)
//              ---------------------------------------
//              find a person by its date
//              Jackson Michael   29-08-1958 (id=4)
//
//              person counter : 5
```