

# Semester Project - Part 1

**DUE DATE: Monday, 3/20/2005, 11:59:59 PM**

This is the first installment of a four (or possibly three) part semester project. Throughout the semester you will be writing a program that will allow the user to enter data for family tree and eventually answer questions about family relationships based on this family tree. In each part you will make the program more powerful using the new programming concepts you are learning.

**You are supposed to work on this project in a project team / study group with a total of three students. Once you commit to a group, you will have to stay with that group for the remainder of the semester.**

**Read the problem completely and very carefully before you start!**

**Problem.** The goal of this project is to practice the design and implementation of a class. So the focus will not be on writing a program that can do many things, but rather on the careful design, implementation and testing of a class to represent a specific type of object (members in a family tree).

You are supposed to write a class called `Person` in which you can store data for a person that might appear in a family tree (such as maybe name, date of birth, place of birth, ...). Implement this class in a file called `familyTree.cpp` where you will also test the class.

Below is a fairly detailed sequence of steps that you need to follow to receive credit for the projects. It follows the approach we discussed in class to design a new class:

1. *Class Interface:* informally describe class interface
2. *Class Data:* determine data fields of the class
3. *Class Declaration:* write the formal class declaration
4. *Class Implementation:* implement the member functions
5. *Class Testing:* test if implementation works correctly.
6. *Project Report:* summarize progress and problems

Continued on next page →

The last two steps at the end has been added, so you can be sure that you worked correctly and so you can show that you know how to use objects of your new class.

You will only provide **one submission** of the project for the entire group. That submission will include

- Printed Cover Page, listing all group member
- Academic integrity statement (will be provided), signed by all group members
- Print-out of your final implementation of the program.
- Print-out of the tests that you ran to check the program.
- Write-up (typed) for Part ?? of the project.
- Electronic submission of your project file.

(a) ***Class Interface:*** informally describe interface for class Person

Most of the work for this step is done already. The following operations will definitely have to be supported by your class.

- The default constructor `Person()` that creates and empty `Person()` object.
- A constructor `Person(...)` that receives all the required data as parameters to creat a new person object.
- A function `Void Read()` to let the user enter the data for and empty person object.
- A function `Void Print()` that prints all the information about a person.
- A function `Bool IsMale()` that returns `true` if the person is male.
- A function `Bool IsBornBefore(Date d)` that returns `true` if the person was born before Date `d`.

In addition specify if there are other functions that your class will support.

(b) ***Class Data:*** determine data fields of the class

Note that there are two parts to this step:

- (i) Determine what data you would like to store for each person. For each data item, list a suitable name, its type, as well a brief description. For example:

*Data Field:*    `first_name`  
*Data Type:*    `string`  
*Description:*   The first name of the person

Continued on next page →

Your class must include a datafield to store the date of birth of the person. This datafield should be of the type `Date` that we discussed in class. The declaration and implementation of this `Date` class is contained in the program file `family_tree_step1.cpp` that is provided to you.

- (ii) Determine which data access function your class will provide. to read (accessor) or change (mutator) data values of the object. Decide which of the data should only be read and which, if any, should be possible to be changed. Once you know that, specify the names and types of the data access functions and give a brief description. For instance:

*Access function:*    `string get_last_name() const`  
*Description:*        Returns the last name of the person.  
*Type of function:*   Accessor function, does not change any data.

(c) ***Class Declaration:* write the formal class declaration**

Now its time to enter the class interface into the `dalotto@york.cuny.edu` file `family_tree_step1.cpp`. Add the data fields you specified in Part ??, as well a function header (that is, the name, parameters, return type) for each of the member functions specified in Part ??, as well as data access functions you specified in Part ??.

(d) ***Class Implementation:* implement the member functions**

- (i) Make sure that your class interface compiles, before you start to implement the member functions.
- (ii) You should start with the constructors `Person()` and `Person(...)`, then functions `Read()` and `Print()`.

Test these first with the following `main()` function:

```
int main()
{
    Person p1;
    Person p2;

    p1.Read();
    p2 = Person("Joe","Smith", .....); // you will need to change the parameters
                                         // inside the (...) to match your constructor

    p1.Print();
    p2.Print();
}
```

- (iii) Next, implement your data access functions as well as `IsMale()` and `IsBornBefore`. Make sure that your program compiles and test with the following `main()` function:

```
int main()
{
    Person p1;
    Person p2;

    p1.Read();
    p2 = Person("Joe","Smith", .....); // you will need to change the parameters
                                        // inside the (...) to match your constructor

    p1.Print();
    p2.Print();

    cout << "P1 is male: " << p1.IsMale() << "\n";
    cout << "P1 is born before P2: "
         << p1.IsBornBefore(p2.get_date_of_birth()) << "\n";
}
```

- (e) ***Class Testing: test if implementation works correctly.***

Try to think of additional ways how you can test your class. Make sure that you include test for every member function you implemented. Determine what types of inputs would be good test cases and modify your main program to run these test.

Document the result and discuss if your program behaved as expected. in the next Part ??

- (f) ***Project Summary:***

Provide a write-up answering the following questions:

- (i) How well does your implementation work? Are there known problems? Are there parts that you were not able to implement?
- (ii) How did you test your work? Which test data did you choose? What were the results?
- (iii) What kind of additions could you think of to add to the project?
- (iv) Which parts of the project were easy to implement? Which were harder?
- (v) How was the work load divided between project team members? Who is responsible for which parts? How much time did you invest in this project?