CSE 202 LAB 6

Modules, External Files

It is often more convenient to work on classes independantly by putting the code in separate files. To do this, you create an interface in a separate file, and the functions in an additional file. For example, lets consider the class Person.

1. Class definitions (interfaces) are stored in a header file, a file with the suffix `.h`

   Create the class definition for the `Person` class in file `Person.h` as follows:

   ```
   #ifndef PERSON_H
   #define PERSON_H
   #include <string>
   using namespace std;

   class Person
   {
   public:
       Person();
       Person(string fname, string lname, string cname);
       string getFirstName();
       string getName();
       string getLastName();
       string getCompany();
   private:
       string firstname;
       string lastname;
       string company;
   }; // Person
   #endif
   ```

2. The methods (member functions) of a class are stored in a file with the same name but with the suffix `.cpp`.

   Complete the member function definitions in the file `Person.cpp` below. This file should begin with:

   ```
   #include "Person.h"
   #include <string>
   using namespace std;

   Person::Person()
   {
       firstname = "";
       lastname = "";
       company = "";
   } // constructor
   ```

   t

   Note the difference between including `string` and `Person.h`

3. You can only compile and not run (link) `Person.cpp` because it does not contain a `main()` function. To compile, use the `-c` option of the compiler which produces an object file with the same name as the source code but with the suffix `.o`

   ```
   g++ -c Person.cpp
   ```

This command produces `Person.o`

4. Now any program that needs to use a Person object can link to `Person.o`

   You may test the correctness of your object file `Person.o` by creating yet another file which only includes the `main()` function and `Person.h`

   Create the file `PersonTest.cpp` to contain:

```
#include "Person.h"
#include <iostream>
#include <string>
using namespace std;

int main()
{
    Person t("Bill", "Gates", "Microsoft");

    cout << t.getName() << endl;
    cout << t.getCompany() << endl;
}
```

   Just compile:

```
g++ -c PersonTest.cpp
```

   Just like before, this produces `PersonTest.o`

   Link and name the executable `PersonTest` :

```
g++ PersonTest.o Person.o -o PersonTest
```

5. Run:

```
PersonTest
```

   Make sure the output is what you expect!

6. Rather than having to compile each module at the command line, it will save time to use a makefile. A makefile lists the different compile and link instructions. The make file for building the above program looks like:

```
   PersonTest: PersonTest.o Person.o
tabkey       g++ PersonTest.o Person.o -o PersonTest

   Person.o: Person.cpp
tabkey       g++ -c Person.cpp

   PersonTest.o: PersonTest.cpp
tabkey       g++ -c PersonTest.cpp
```

   Create and save your make file in the file `PersonTest.make`

   Don't forget the second lines must start with a tab.

   You may re-build the program every time you make any changes using the command:

```
make -f PersonTest.make
```

7. If you have time repeat the exercise by adding the class Location :

```
class Location
  {
  public:
      Location();
      Location(string addr, string bldg, string rm);)
      string getLocationAddress();
      string getLocationBuilding();
      string getLocationRoom();
  private:
      string address;
      string building;
      string room;
  }; // Location
```