ECE244 **Programming Fundamentals**

Lab Assignment #7: Inheritance

1. Objectives

The objective of this assignment is to provide you with practice on the use of inheritance in C++ programming. This will be done in the context of re-implementing the simple student-marks database of Assignment 5 to allow the storage and retrieval of records of any type, not only of type `studentRecord`.

2. Problem Statement

In this assignment, you will implement a simple array-based database to store and retrieve records. In the first part of the assignment, you will implement two classes: `Record` and `DB`. The `Record` class will serve as a base class from which other types of record classes can be derived. The `DB` class will be used to create a database of `Record` objects. In the second part of the assignment, you will design and implement the class `studentRecord`, which is derived from the class `Record`. You will test your implementation with the `Driver` you wrote for Assignment 5. However, your implementation of `Record` and `DB` must work for ***any*** class that is derived from `Record`, even without any knowledge on your part of what the derived class does.

2.1 The `Record` Class

The `Record` class has fields to represent the number (key) of an individual. It also has the following member functions associated with it.

- `Record ()`. This is the default constructor. It creates an empty record.
- `virtual ~Record ()`. This is the destructor. It `delete`s all dynamic components of the record.
- `void setNumber (unsigned int Num)`. This function sets the number in the record to `Num`.
- `unsigned int getNumber ()`. This function returns the number in the record.
- `virtual void print () = 0`. This function prints the number to the standard output. It is an abstract function.

2.1 The DB Class

The database has fields to represent the structure of the database and the following member functions associated with it:

- `DB ()`. This is the default constructor. It creates an empty database.
- `~DB ()`. This is the destructor. It deletes all the records in, and the structures of, the database.
- `bool insert (Record * newRecord)`. This method inserts the record pointed to by `newRecord` into the database. If a record with the same number already exists in the database, `false` is returned. The database must not be full when this method is called.
- `bool retrieve (unsigned int Num, Record * & searchRecord)`. This method searches the database for a record with a number `Num`. If the record is found, then `searchRecord` is made to point to it, and `true` is returned. Otherwise, a `false` is returned and `searchRecord` is set to NULL.
- `bool remove (unsigned int Num)`. The method deletes the record with number `Num` from the database. If the record is found and deleted, `true` is returned. Otherwise `false` is returned.
- `void clear ()`. This method clears the database by deleting all the records in the database, effectively returning the database to its initial empty state.
- `bool isEmpty ()`. This method returns `true` if the database is empty, otherwise, it returns `false`.
- `bool isFull ()`. This method returns `true` if the database is full, otherwise it returns `false`.
- `void dump ()`. This method dumps out the database to the standard output, one record at a time (using the `Record`'s virtual `print` function), separated by empty lines, and sorted in ascending order of numbers.

3. Preparation

This assignment builds on Assignment 5. Thus, it pays to examine the handout and your solution to Assignment 5. You must also work on the assignment ***before*** you come to the lab. You should prepare an initial implementation of classes: `Record`, `DB` (Part I below) `studentRecord` and `Driver.cpp` (Parts III and IV below) ***before*** you arrive to the lab session.

4. Procedure

Create a sub-directory called `lab7` in your `ece244` directory, using the `mkdir` command. Make it your working directory. ***You may modify any of the `.h` files that you will download for this assignment only by adding private function members. You may NOT add data members nor public function members***!

4.1 Part I – The `Record` Class

Use a browser to download the file `Record.h`. It contains the definition of the class `Record`, which is intended to hold one person's information. The purpose of the fields and purpose of function members is described in this file. ***You may modify this file only as described above; i.e., only by adding private function members. You may NOT add data members nor public function members***.

Write the implementation of this class in a file called `Record.cpp`.

4.2 Part II – The `DB` Class

In this part, you will use the array-based database implementation from Assignment 5. However, in this case, the array will contain pointer to `Record` objects, not `studentRecord` objects.

Use a browser to download the file `DB.h`. It contains the definition of the class `DB`. ***You may modify this file only as decribed above***. Write the implementation of this class in a file called `DB.cpp`.

4.3 Part III – The `studentRecord` Class

In this part, you must derive the class `studentRecord` from the class `Record`. The class `studentRecord` adds the following data members:

```
unsigned int marks[5];
```

It also adds the following member functions:

- `void setMark (int index, unsigned int mark)`. This method sets the mark at index `index` of the student record to `mark`. The marks are indexed 0 to 4. The method does not check that `index` is in this range; the caller must do so. A mark is between 0 and 100, and the caller must also check this.
- `Unsigned int getMark (int index)`. This method returns the mark at index `index` of the student record. The marks are indexed 0 to 4. The method does not check that `index` is in this range; the caller must do so. A mark is between 0 and 100.
- `void setFirstName (char * firstName)`. This function sets the first name in the record to the value of the string `firstName`.
- `void setLastName (char * lastName)`. This function sets the last name in the record to the value of the string `lastName`.

- `char * getFirstName ()`. This function returns a pointer to the first name in the record.
- `char * getLastName ()`. This function returns a pointer to the last name in the record.
- `void print ()`. This method prints the student record to the standard output, in the following format:

  Student number: `number` ↵
  Student name: `lastname`, `firstname` ↵
  Student marks: `m1`, `m2`, `m3`, `m4`, `m5`↵

The student number should be printed as a 9-digit integer. The ↵ character indicates `newline` (i.e., `endl`).

Write `studentRecord.h` and implement it in `studentRecord.cpp`.

4.4 Part IV – The `Driver` Program

In this part of the assignment, you will re-use (with some minor changes) the array-database driver from Assignment 5 to test your implementation of `studentRecord` with `DB`.

Write a `Makefile` to generate the executable of `Driver`.

5. Deliverables

Submit the following components of your implementations: `Record.h`, `Record.cpp`, `DB.h`, and `DB.cpp`, using the `submitece244f` command as follows:

```
submitece244f 7 Record.h Record.cpp DB.h DB.cpp
```

There is no need to submit the `studentRecord` and `Driver` files. These files were simply for your benefit to test your implementation of `Record` and `DB`. Indeed, your `Record` and `DB` classes will be tested and marked with ***an arbitrary*** class, derived from `Record`, and its associated `Driver`.