Background

The Mine Safety and Health Administration (MSHA) has been reorganized and re-designated as the Job Site Safety and Health Administration (JS^{2}HA). The organization is now responsible for analysis of all work related accidents, so our existing information processing systems need to be updated. You will reuse your `Date` and `Time` classes in this project. If they were working correctly for Project #1 then no changes should be required. The `Accident` classes will need to be completely rewritten.

As before, accident data are received in text files. The mining accident data file has the following new layout:

```
accidentNumber, accidentDate, accidentTime, FIPS_StateCode, mineID, injuryCode, numberOfInjuries,
daysRestricted, daysLost, narrative
```

All mine accident data are stored in the file `mineAccidentsTenYear.dat`, a sample of this data is shown below and the complete file may be downloaded from Blackboard (some narrative fields have been shortened to present each record of the sample data on one line).

```
220081430017 2008-05-15 15:45 29 2300781 06 1 0 0 Miner was stepping down off a trailer mounted water pump. Miner stepped on and slipped on a wet stone. Miner attempted…
220083080028 2008-10-23 07:30 29 2300781 03 1 0 8 Miner picked up a piece of steel I-beam approx 2ft long weighing about 100 lbs. He carried it for approx. 5 feet and set it …
220111860042 2011-06-25 09:30 42 3609407 00 0 0 0 Roof fall was in the 3rd Northwest Butt Room section in #23 room 45' inby spad # 2082 The fall was approx. 55' L. in #23 room …
220071690009 2007-06-08 13:00 18 1200066 04 1 71 32 EE WAS USING VAC TRUCK IN THE ABC TUNNEL. HOSE CAUGHT IN PILE AND JERKED FORWARD PUSHING HER THUMB BACKWARDS.
220060810024 2006-03-14 08:30 13 0901139 04 1 6 5 Employee was changing the drive belt on the screen deck. The lifting guide broke out of the top of the motor causing the …
```

The railroad accident data file has the following layout:

```
accidentNumber, accidentDate, accidentTime, FIPS_StateCode, railroadID, nearestCityOrTown,
carsCarryingHAZMAT, personsEvacuated, personsInjured, personsKilled, narrative
```

All rail accident data are stored in the file `rrAccidentsTenYear.dat`, a sample of this data is shown below and the complete file may be downloaded from Blackboard (some narrative fields have been shortened to present each record of the sample data on one line).

```
118430-000083996 2011-01-12 08:29 36 ATK DUNKIRK 0 0 2 1 TRAIN 48 OPERATING WITH LOCOMOTIVES E/118-E/188 AND 14 CARS STRUCK A TRACTOR-TRAILER AT MP39.38…
118455-118455 2011-01-12 14:45 11 ATK WASHINGTON,_DC 0 0 0 0 AN AMTRAK CONTRACTOR WAS OPERATING A COMMISSARY TRACTOR OVER THE CROSSING ON 18 TRACK …
1185-1185 2009-09-03 03:58 17 IHB RIVERDALE 2 0 0 0 RUN 598 HUMPING PAIR OF CARS (KCS 129335) FOR BC21, FOLLOWED BY SINGLE CAR (GATX 66787) FOR BC18. PA IR OF …
118544-118544 2011-01-18 07:22 09 ATK WALLINGFORD 0 0 1 0 TRAIN 141 OPERATING WITH LOCOMOTIVE E/40 AND 7 CARS STRUCK A TRACTOR-TRAILER TRUCK AT MP10.57, …
118546-118546 2011-01-18 01:45 25 ATK BOSTON 0 0 0 0 TRAIN 194S EQUIPMENT WAS OPERATED INTO A PLATE ORDER WITH PANTOGRAPH RAISED AND INVERTED PANTOGRAPH …
```

You will also need to use a file containing Federal Information Processing Standards (FIPS) Codes for states. The FIPS Codes are sored in a file named `FIPS_StateCodes.dat` having the following layout:

```
numericCode, alphaCode, stateName
```

A sample of this data is shown below and the complete file may be downloaded from Blackboard.

```
46 SD South Dakota
47 TN Tennessee
48 TX Texas
49 UT Utah
50 VT Vermont
51 VA Virginia
```

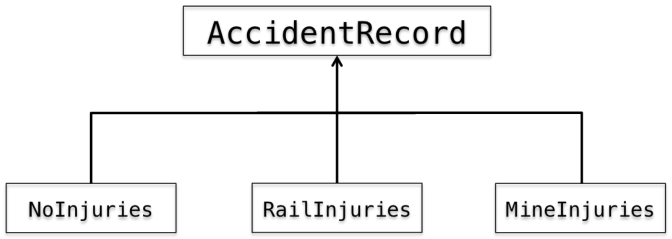
Requirements

To begin, I recommend that you implement the functions necessary to read the input data files. These will be stand-alone functions in your driver program (not class member functions). When fully implemented, the function prototypes will be:

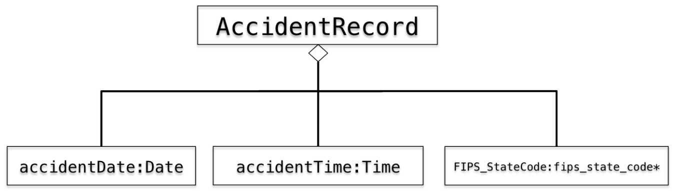
```
    void loadDataFIPS(string, vector<fips_state_code>&);

    void loadDataRail(string, vector<fips_state_code>&,
                        vector<RailInjuries>&, vector<NoInjuries>&);

    void loadDataMine(string, vector<fips_state_code>&,
                        vector<MineInjuries>&, vector<NoInjuries>&);
```

However, it may be prudent to first implement the functions without the vector parameters. The string parameter in each function is the file name. These three input file names should be passed to your program as command line arguments in the same order as the function prototypes are listed above. Implementing the code to simply open and read the files first is a good way to minimize the "moving parts" while you incrementally code and test small parts of your functions. Adding the vector parameters will be no problem after the classes whose objects they store have been declared.

Next you should implement the classes for this project as illustrated in the following inheritance hierarchy:



Note that object composition and aggregation via pointers are also required for the `AccidentRecord`:



Specification details for all classes shall be placed in file `AccidentRecord.h` and implementation details in the file `AccidentRecord.cpp`. UML diagrams for these classes (and the `struct` for FIPS State Code objects) are provided in an appendix.

Using class inheritance as shown above will improve the quality of our software. By abstracting common data and including those data members in the parent class we reduce storage requirements and code duplication. Different types of accidents also require different reports. Our object-oriented solution will address that by providing custom output for each different class via overloading of their steam insertion operators.

Output for any type of accident involving no injury or death shall follow the format below. (Note that the `NoInjuries` class does not have an overloaded stream insertion operator. It uses the operator inherited from `AccidentRecord`.)

```
Accident Record Report (no injuries or fatalities):
000000766-000000766
2004-02-07 00:14
Rail CSX  Maryland
TRAIN Q40605 OPERATING EASTWARD IN LIGHT DYNAMIC BRAKE WHEN TRAIN IN EMERGENCY,
AFTER WHICH ENGINEER ACTUATED THE INDEPENDENT BRAKE. OF THE CARS DERAILED, 3 ARE
HAZMAT - RPBX 17142/8800GAL RELEASED; R PBX 17108/100GAL RELEASED; WACX
15179/150GAL RELEASED
```

Output for a railroad accident with persons evacuated, injured, or killed shall follow the format below.

```
Railroad Acident with Injuries Report:
000027149-000027149
2006-11-27 09:00
Rail CSX HARRIS,  North Carolina
HAZMAT cars involved: 0
Persons evacuated: 0 Persons injured: 0 Persons killed: 1
U31521 STRUCK AN AUTOMOBILE AT CROSSING THAT RESULTED IN A FATALITY AND
DAMAGE TO CROSSING WARNING D EVICE.
```

Finally, the output for a mining accident with persons injured or killed shall follow the format below.

```
Mining Acident with Injuries Report:
220122140014
2012-07-19 19:00
1519097 Mine  Kentucky
Injury Code: 03 Number of Injuries: 1
Days Restructed Duty: 0 Work Days Lost: 82
Employee was walking back from putting up bolts to move roof bolter to next
row when a piece of draw rock 3.5'x 3' x 2 fell striking him on the back of
the head and neck.
```

Once your classes have been implemented, you may resume work on the functions that load data from the input files. You must also incorporate exception handling in your project. At a minimum, include `try/catch` blocks in your driver program and in each function that loads file data. The driver program shall throw a string exception if there are not enough command line arguments. The functions that read file data shall throw a string exception if the data file fails to open. Additionally, those functions shall re-throw the exception so that the driver program can gracefully exit if necessary.

To demonstrate the functionality of your software, the driver program should output the count of objects in each vector. Additionally, "several" (5 or 10) reports from each accident type should be displayed on the screen.

Programming Skills

The programming skills required to complete this assignment include:

- Class inheritance
- Exception handling
- Object oriented design
- Class composition
- Aggregation via pointers
- Function overloading
- Operator overloading

Submission Details

What to submit:	One compressed file containing all source code and any other files associated with this project. The file name shall be `<netID>P2.zip`. You must separate your class specification details from your class implementation details. Ensure that your `.h` files contain sufficient comments for each data member and class method. Additionally, you must provide another `.cpp` file that contains function `main()`. This "driver" program is where class objects are instantiated and functionality of the software is demonstrated. Use the following file names: `DateTime.h` `DateTime.cpp` `AccidentRecord.h` `AccidentRecord.cpp` `<netID>P2.cpp`
Due date/time:	Tuesday, 18 FEB 2014, no later than 11:00 am. Late submissions will be penalized 2.5 points for each 15 minutes late. If you are over 10 hours late you may turn in the project to receive feedback but the grade will be zero. In general requests for extensions will not be considered.
Point Value:	100 points

Academic Integrity

This is an individual project and all work must be your own. Refer to the guidelines specified in the *Academic Honesty* section of this course syllabus or contact me if you have any questions.

Include the following comments at the start of your program:

```
/*
 * <FileName>.<file extension>
 *
 *  COSC 052 Spring 2014
 *  Project #2
 *
 *  Due on: FEB 18, 2014
 *  Author: <your name>
 *  netID:  <your netID>
 *
 *
 *  In accordance with the class policies and Georgetown's
 *  Honor Code, I certify that, with the exception of the
 *  class resources and those items noted below, I have neither
 *  given nor received any assistance on this project.
 */
```

Grading

This graded assignment is worth 100 points and will be counted as part of the *Programming Projects* category for the course. Your final score is based on common deductions, as well as, a detailed rubric of points. The table below lists common deductions.

Common Deductions	
Program does not compile	-35.00
Program compiles but has warnings (deduction varies depending on how bad, value listed is max)	-15.00
Program crashes during execution (deduction varies depending on how bad, value listed is max)	-25.00
Filenames do not follow conventions specified (deduction varies depending, value listed is max)	-30.00
Uses any global variables	-20.00
Required comments and honor statement not included at start of file exactly as specified	-35.00
Late penalty for each 15 minutes late	-2.50

The table below contains the detailed rubric of specific points for this project.

```
GRADE RUBRIC TO BE PUBLISHED SEPARATELY
```

Appendix A – UML Diagrams

```
                              AccidentRecord

   - accidentNumber : string
   - accidentDate : Date
   - accidentTime : Time
   - FIPS_StateCode : *fips_state_code
   - narrative : string
   - facilityCategory : string
   - facilityID : string

   + AccidentRecord(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                    fips_state_code *sc = NULL, string narr = "", string cat = "",
                    string id = "")

   + AccidentRecord(const AccidentRecord&)

   + ~AccidentRecord()

   + setDataMembers(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                    fips_state_code *sc = NULL, string narr = "", string cat = "",
                    string id = "") : void

   + operator =(const AccidentRecord&) : const AccidentRecord&

   + getAccidentNumber() const : string
   + getAccidentDate() const : Date
   + getAccidentTime() const : Time
   + getFIPS_stateCode() const : fips_state_code*
   + getNarrative() const : string
   + getCategory() const : string
   + getID() const : string

   + operator <<(ostream&, const AccidentRecord&) : ostream&    //non-member friend function
```

```
                                NoInjuries

   + NoInjuries(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                    fips_state_code *sc = NULL, string narr = "", string cat = "Unknown",
                    string id = "")

   + NoInjuries(const NoInjuries&)

   + setDataMembers(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                    fips_state_code *sc = NULL, string narr = "", string cat = "Unknown",
                    string id = "") : void

   + ~NoInjuries()
```

```
                                 RailInjuries

   - nearestTownOrCity : string
   - carsCarringHAZMAT : integer
   - personsEvacuated : integer
   - personsInjured : integer
   - personsKilled : integer

   + RailInjuries(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                  fips_state_code *sc = NULL, string narr = "",
                  string cat = "Rail", string id = "", string city = "",
                  int hazmat = 0, int evac = 0, int inj = 0, int kld = 0)

   + RailInjuries(const RailInjuries&)
   + ~RailInjuries()

   + setDataMembers(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                  fips_state_code *sc = NULL, string narr = "",
                  string cat = "Rail", string id = "", string city = "",
                  int hazmat = 0, int evac = 0, int inj = 0, int kld = 0): void

   + getNearestCityOrTown() const : string
   + getCarsCarryingHAZMAT() const : integer
   + getPersonsEvacuated() const : integer
   + getTotalInjured() const : integer
   + getTotalKilled () const : integer

   + operator =(const RailInjuries&) : const RailInjuries&
   + operator <<(ostream&, const RailInjuries&) : ostream&  //non member friend function
```

```
                                 MineInjuries

   - injuryCode : string
   - numberOfInjuries : integer
   - daysRestricted : integer
   - daysLost : integer

   + MineInjuries(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                  fips_state_code *sc = NULL, string narr = "",
                  string cat = "Mine", string id = "", string ic = "",
                  int nI = 0, int dR = 0, int dL = 0)

   + MineInjuries(const MineInjuries&)
   + ~MineInjuries()

   + setDataMembers(string aNum = "", Date = Date(1923, 1, 1), Time = Time(00,01),
                  fips_state_code *sc = NULL, string narr = "",
                  string cat = "Mine", string id = "", string ic = "",
                  int nI = 0, int dR = 0, int dL = 0): void

   + getInjuryCode() const : string
   + getNumberOfInjuries() const : integer
   + getDaysRestricted() const : integer
   + getDaysLost() const : integer

   + operator =(const MineInjuries&) : const MineInjuries&
   + operator <<(ostream&, const MineInjuries&) : ostream&  //non member friend function
```

Note: This UML diagram represents a `struct`, not a `class`.

```
fips_state_code
----------------------------------------
+ numericCode : string
+ alphaCode : string
+ stateName : string
----------------------------------------
// no member functions for this struct
```

Appendix B – Data Sets

Mining Accident Data

Source: http://www.data.gov/energy/datasets/accident-injuries-data-set?comm=3

The Department of Labor maintains mining accident data. The file we are using is extracted from data containing information about mining accidents and injuries since 1983. For detailed information about the fields in this data set, refer to the Project #1 Appendix or the link above.

Rail Accident Data

Source: http://safetydata.fra.dot.gov/OfficeofSafety/Default.aspx

The Federal Railroad Administration maintains rail accident data. The file we are using is an extract from consolidated annual "Accident Data On Demand" downloads.

Field descriptions for the complete file can be found in Document Number 6180.54 at:
http://safetydata.fra.dot.gov/OfficeofSafety/publicsite/downloadFStructure.aspx

Federal Information Processing Standards (FIPS) State Codes

The National Institute of Standards and Technology (NIST) maintains FIPS publications. The file of state codes that we are using is an extract of codes listed at the source link below.

Source: http://itl.nist.gov/fipspubs/fip5-2.htm