CS22510 Assignment

Samuel B Sherar (sbs1)

March 22, 2013

Contents

1 Event Creation

1.1 Code

1.1.1 main.cpp

```
#include <iostream>
#include <vector>
#include "Event.h"
#include "List.h"
#include "Competitors.h"
#include "CLI.h"
#include "FileWriter.h"

using namespace std;

void event_menu();
void entrants_menu();
void courses_menu();
void print_files();

CLI cli;
Event * event;
FileWriter * file_io;
vector<Competitor> competitors;
vector<Course> courses;

int main(int argc, char** argv) {
  char choice = 'A';
  do {
    cli.write_main_menu();
    choice = cli.get_input("");
    switch(choice) {
      case '1':
        event_menu();
        break;
      case '2':
        entrants_menu();
        break;
      case '3':
        courses_menu();
        break;
      case '4':
        print_files();
        break;
    }
  } while(choice != 'q');
}

/**
 * Handles all the submenus for the event, including
 * creating and showing the current created event
 */
```

```
void event_menu() {
  char e_choice = 'A';
  do {
    cli.write_event_menu();
    e_choice = cli.get_input("");
    if('1' == e_choice) {
      string name = cli.get_input_string("Please enter the event
         name");
      int day = cli.get_input_int("Please enter a day");
      int month = cli.get_input_int("Please enter a month");
      int year = cli.get_input_int("Please enter a year");
      int hour = cli.get_input_int("Please enter the start hour");
      int minute = cli.get_input_int("Please enter the start minute");
      event = new Event(name, day, month, year, hour, minute);
    } else if('2' == e_choice) {
      if(NULL == event) {
        cli.write_screen("No event has beem created");
      } else {
        cli.write_screen(event->format_for_screen());
      }
    }
  } while(e_choice != 'q');
}

/**
 * Handles all the submenus for the entrants, including
 * Adding entrants and listing alrady created entrants
 */
void entrants_menu() {
  char en_choice = 'A';
  do {
    cli.write_entrant_menu();
    en_choice = cli.get_input("");
    if('1' == en_choice) {
      string name = cli.get_input_string("Please enter the competitor
         name");
      char course_id = cli.get_input("Please enter the course id");
      Competitor tmp (course_id, name);
      competitors.push_back(tmp);
    } else if('2' == en_choice) {
      for(unsigned int i = 0; i < competitors.size(); i++) {
        char formatted[100];
        Competitor tmp = competitors[i];
        sprintf(formatted, "%02d \t %c \t",
              i + 1,
              tmp.course_id);
        cout << formatted << tmp.name << endl;
      }
    }
  } while(en_choice != 'q');
}
```

```
/**
 * Handles all the submenus for the courses, including
 * creating and listing of current courses created.
 */
void courses_menu() {
  char c_choice = 'A';
  do {
    cli.write_courses_menu();
    c_choice = cli.get_input("");
    if('1' == c_choice) {
      char course_id = cli.get_input("Please enter the course id");
      Course c (course_id);
      int cv_choice = 0;
      do {
        cv_choice = cli.get_input_int("Please enter the checkpoint
           number (0 to exit)");
        if(0 != cv_choice) {
          c.add_node(cv_choice);
        }
      } while(cv_choice > 0);
      courses.push_back(c);
    } else if('2' == c_choice) {
      for(unsigned int i = 0; i < courses.size(); i++) {
        courses[i].format_for_screen();
      }
    }
  } while(c_choice != 'q');
}

/**
 * Checks if all data is available to print out
 * to the file, and then writes the new file
 * to the directory specified by the user
 */
void print_files() {
  if(NULL == event ||
    competitors.size() == 0 ||
    courses.size() == 0) {

    cli.write_screen("One or more items are not created. Please create
       an Event, a Competitor and a Course");
  } else {
    string path = cli.get_input_string("Please enter the folder path
       to create the files");
    file_io = new FileWriter(path);
    file_io->write_event_file(event);
    file_io->write_competitor_file(competitors);
    file_io->write_courses_file(courses);
  }
}
```

1.1.2 CLI.h

```
class CLI {
  public:
    CLI();
    static void write_main_menu();
    static void write_event_menu();
    static void write_entrant_menu();
    static void write_courses_menu();
    static char get_input(std::string);
    static std::string get_input_string(std::string);
    static int get_input_int(std::string);
    static void write_screen(std::string);
};
```

1.1.3 CLI.cpp

```
#include <iostream>
#include "CLI.h"

using namespace std;

CLI::CLI() {}

/**
 * Writes out the main menu to screen
 */
void CLI::write_main_menu() {
  cout << endl;
  cout << " Welcome to Event Creation " << endl;
  cout << endl;
  cout << " 1) Create Event Info" << endl;
  cout << " 2) Add Entrants " << endl;
  cout << " 3) Add Courses " << endl;
  cout << " 4) Print to file" << endl;
  cout << " q) Quit" << endl;
  cout << endl;
  cout << "Please enter your choice > ";
}

/**
 * Writes out the event menu to screen
 */
void CLI::write_event_menu() {
  cout << endl;
  cout << " Event: " << endl;
  cout << " 1) Add new Event" << endl;
  cout << " 2) Print out current event" << endl;
  cout << " q) Quit back to main menu" << endl;
  cout << endl;
  cout << "Please enter your choice > ";
}

/**
 * Writes out the entrants menu to the screen
```

```
 */
void CLI::write_entrant_menu() {
  cout << endl;
  cout << " Entrants:" << endl;
  cout << " 1) Add Entrant" << endl;
  cout << " 2) List Entrants" << endl;
  cout << " q) Quit back to main menu" << endl;
  cout << endl;
  cout << "Please enter your choice > ";
}

/**
 * Writes out the courses menu to the screen
 */
void CLI::write_courses_menu() {
  cout << endl;
  cout << " Courses:" << endl;
  cout << " 1) Add new Course " << endl;
  cout << " 2) List Courses " << endl;
  cout << " q) Quit back to main menu" << endl;
  cout << endl;
  cout << "Please enter your choice > ";
}

/**
 * Writes out the value given with a newline character appended
 * @param val the message
 */
void CLI::write_screen(string val) {
  cout << val << endl;
}

/**
 * Gets a character from the input, with an overloaded question
 * if the question has already been written out to the screen
 * (such if being used with any CLI::write_X_menu()
 * @param question the optional question to be written
 * @returns the chosen character
 */
char CLI::get_input(string question) {
  if(!question.empty()) {
    cout << question << " > ";
  }
  char input;
  cin >> input;
  return input;
}

/**
 * Gets input string from the user
 * @param question the question to be asked
 * @returns string from the user
```

```
 */
string CLI::get_input_string(string question) {
  cout << question << " > ";
  string input;
  cin.ignore();
  getline(cin, input);
  return input;
}

/**
 * Gets a integer input from the user
 * @param question the question to be asked
 * @returns int from the user
 */
int CLI::get_input_int(string question) {
  cout << question << " > ";
  int input;
  cin >> input;
  return input;
}
```

1.1.4 Competitors.h

```
#ifndef COMPETITORS_H
#define COMPETITORS_H

using namespace std;


class Competitor {

  public:
    char course_id;
    string name;
    Competitor(char, string);
    //~Competitor();
    void format_for_file();

};

#endif
```

1.1.5 Competitor.cpp

```
#include <iostream>
#include <vector>
#include "Competitors.h"

using namespace std;

/**
 * Constructor for the competitor, which sets the name
 * and course identifier
```

```
 * @param course_id the course identifier
 * @param name the name of the competitor
 */
Competitor::Competitor(char course_id, string name) {
  this->course_id = course_id;
  this->name = name;
}
```

1.1.6 List.h

```
#include <vector>
#ifndef LIST_H
#define LIST_H

class Course;
class CP_Node;
using namespace std;
class Course {
  private:
    CP_Node * head;

  public:
    vector<int> list;
    char course_id;
    Course(char);
    ~Course();
    void add_node(int);
    void format_for_screen();
    string format_for_file();

};

class CP_Node {
  public:
    int node_id;
    CP_Node * next;
    CP_Node(int);
};

#endif
```

1.1.7 Course.cpp

```
#include "List.h"
#include <iostream>
#include <vector>
#include <sstream>

using namespace std;

/**
 * Creates a course with a course identifier
 */
```

```
Course::Course(char id) {
  this->course_id = id;
}

/**
 * Deconstructor for the course class
 */
Course::~Course() {

}

/**
 * Adds node to the vector
 */
void Course::add_node(int node_id) {
  this->list.push_back(node_id);
}
/**
 * Outputs the course identifier with the list
 * nodes afterwards
 */
void Course::format_for_screen() {
  cout << this->course_id << " ";
  for(unsigned int i = 0; i < list.size(); i++) {
    cout << list[i] << " ";
  }
  cout << endl;
}

/**
 * Formats the course to a string with format
 * ID NumberNodes Nodes**
 * @returns formatted string
 */
string Course::format_for_file() {
  ostringstream os;
  os << this->course_id << " ";
  os << this->list.size() << " ";
  for(unsigned int i = 0; i < list.size(); i++) {
    os << this->list[i] << " ";
  }
  return os.str();
}
```

1.1.8 Event.h

```
#include <string>

#ifndef EVENT_H
#define EVENT_H
class Event {
  std::string event_name;
  int day, month, year, hour, minute;
```

```
  public:
    Event(std::string, int, int, int, int, int);
    //Event();
    char* format_printing_file();
    std::string format_for_screen();
    std::string get_name();
    std::string get_date();
};
#endif
```

1.1.9 Event.cpp

```
#include <iostream>
#include "Event.h"
#include <string>
#include <stdio.h>

using namespace std;

/**
 * Creates an event
 * @param info  information of the event
 * @param day the day
 * @param month the month
 * @param year the year
 * @param hour the hour
 * @param minute the minute
 */
Event::Event(string info, int day, int month, int year, int hour, int
   minute) {
  this->event_name = info;
  this->day = day;
  this->month = month;
  this->year = year;
  this->hour = hour;
  this->minute = minute;
}
/**
 * Creates a formatted string to be outputted to
 * the screen
 * @returns the formatted string
 */
string Event::format_for_screen() {
  string ret;
  if(this->event_name.empty() ||
    0 == this->day ||
    0 == this->month ||
    0 == this->year ||
    0 == this->hour ||
    0 == this->minute) {
      ret = "No event has been created\n";
  } else {
    ret.append("Event: ");
```

```
    ret.append(this->event_name);
    ret.append("\n");
    char formatted_time[100];
    sprintf(formatted_time, "on %02d/%02d/%04d at %02d  :%02d",
          this->day,
          this->month,
          this->year,
          this->hour,
          this->minute);
    ret.append(formatted_time);
    ret.append("\n");
  }
  return ret;

}

/**
 * Returns the name of the event
 * @return the name of the event
 */
string Event::get_name() {
  return this->event_name;
}

/**
 * Returns the formatted date
 * @return formatted string with date/time
 */
string Event::get_date() {
  char formatted_time[100];
  sprintf(formatted_time, "on %02d/%02d/%04d at %02d:%02d",
        this->day,
        this->month,
        this->year,
        this->hour,
        this->minute);
  return formatted_time;
}
```

1.1.10 FileWriter.h

```
#include <vector>
#include "Event.h"
#include "List.h"
#include "Competitors.h"

using namespace std;

class FileWriter {
  private:
    string path;

  public:
```

```
    FileWriter(string);
    void write_event_file(Event*);
    void write_courses_file(vector<Course>);
    void write_competitor_file(vector<Competitor>);
};
```

1.1.11 FileWriter.cpp

```
#include <fstream>
#include <vector>
#include <iostream>
#include "FileWriter.h"
#include "Event.h"

using namespace std;

/**
 * Constructor which sets the path for all the files
 * to be written out to
 */
FileWriter::FileWriter(string path) {
  this->path = path;
}

/**
 * Writes the event model to a file
 */
void FileWriter::write_event_file(Event * event) {
  string file_path = this->path.c_str();
  file_path.append("event.txt");
  cout << file_path << endl;
  ofstream file (file_path.c_str());
  if(file.is_open()) {
    file << event->get_name() << "\n";
    file << event->get_date() << "\n";
  }
  file.close();
}

/**
 * Writes the competitors to the file
 */
void FileWriter::write_competitor_file(vector<Competitor> competitors)
   {
  string file_path = this->path.c_str();
  file_path.append("comp_data.txt");
  ofstream file(file_path.c_str());
  if(file.is_open()) {
    for(unsigned int i = 0; i < competitors.size(); i++) {
      file << (i + 1) << " ";
      file << competitors[i].course_id << " ";
      file << competitors[i].name << "\n";
    }
```

```
  }
  file.close();
}

/**
 * Writes the courses out to the file
 */
void FileWriter::write_courses_file(vector<Course> courses) {
  string file_path = this->path.c_str();
  file_path.append("courses.txt");
  ofstream file(file_path.c_str());
  if(file.is_open()) {
    for(unsigned int i = 0; i < courses.size(); i++) {
      file << courses[i].format_for_file() << "\n";
    }
  }
  file.close();
}
```

1.1.12 Makefile

```
all:
  g++ -g -Wall main.cpp Event.cpp Course.cpp Competitor.cpp CLI.cpp
     FileWriter.cpp -o bin/run

clean:
  rm -rf bin/*

run:
  bin/run
```

1.2 Compiler Output

```
event-creation(master*): make clean && make
rm -rf bin/*
g++ -g -Wall main.cpp Event.cpp Course.cpp Competitor.cpp CLI.cpp
   FileWriter.cpp -o bin/run
```

1.3 Terminal Output

```
 Welcome to Event Creation

 1) Create Event Info
 2) Add Entrants
 3) Add Courses
 4) Print to file
 q) Quit

Please enter your choice > 1

 Event:
 1) Add new Event
 2) Print out current event
```

```
 q) Quit back to main menu

Please enter your choice > 1
Please enter the event name > Bob Oreillys Amazing Race
Please enter a day > 2
Please enter a month > 2
Please enter a year > 2013
Please enter the start hour > 7
Please enter the start minute > 30

 Event:
 1) Add new Event
 2) Print out current event
 q) Quit back to main menu

Please enter your choice > 2
Event: Bob Oreillys Amazing Race
on 02/02/2013 at 07  :30


 Event:
 1) Add new Event
 2) Print out current event
 q) Quit back to main menu

Please enter your choice > q

 Welcome to Event Creation

 1) Create Event Info
 2) Add Entrants
 3) Add Courses
 4) Print to file
 q) Quit

Please enter your choice > 2

 Entrants:
 1) Add Entrant
 2) List Entrants
 q) Quit back to main menu

Please enter your choice > 1
Please enter the competitor name > Bugs Duggan
Please enter the course id > D

 Entrants:
 1) Add Entrant
 2) List Entrants
 q) Quit back to main menu

Please enter your choice > 1
```

```
Please enter the competitor name > Trevor Nelson
Please enter the course id > E

 Entrants:
 1) Add Entrant
 2) List Entrants
 q) Quit back to main menu

Please enter your choice > 1
Please enter the competitor name > Bob Oreilly
Please enter the course id > A

 Entrants:
 1) Add Entrant
 2) List Entrants
 q) Quit back to main menu

Please enter your choice > 2
01   D  Bugs Duggan
02   E  Trevor Nelson
03   A  Bob Oreilly

 Entrants:
 1) Add Entrant
 2) List Entrants
 q) Quit back to main menu

Please enter your choice > q

 Welcome to Event Creation

 1) Create Event Info
 2) Add Entrants
 3) Add Courses
 4) Print to file
 q) Quit

Please enter your choice > 3

 Courses:
 1) Add new Course
 2) List Courses
 q) Quit back to main menu

Please enter your choice > 1
Please enter the course id > A
Please enter the checkpoint number (0 to exit) > 1
Please enter the checkpoint number (0 to exit) > 2
Please enter the checkpoint number (0 to exit) > 3
Please enter the checkpoint number (0 to exit) > 4
Please enter the checkpoint number (0 to exit) > 5
Please enter the checkpoint number (0 to exit) > 9
```

```
Please enter the checkpoint number (0 to exit) > 10
Please enter the checkpoint number (0 to exit) > 14
Please enter the checkpoint number (0 to exit) > 7
Please enter the checkpoint number (0 to exit) > 3
Please enter the checkpoint number (0 to exit) > 2
Please enter the checkpoint number (0 to exit) > 1
Please enter the checkpoint number (0 to exit) > 0

 Courses:
 1) Add new Course
 2) List Courses
 q) Quit back to main menu

Please enter your choice > 1
Please enter the course id > D
Please enter the checkpoint number (0 to exit) > 1
Please enter the checkpoint number (0 to exit) > 2
Please enter the checkpoint number (0 to exit) > 3
Please enter the checkpoint number (0 to exit) > 4
Please enter the checkpoint number (0 to exit) > 10
Please enter the checkpoint number (0 to exit) > 3
Please enter the checkpoint number (0 to exit) > 2
Please enter the checkpoint number (0 to exit) > 1
Please enter the checkpoint number (0 to exit) > 0

 Courses:
 1) Add new Course
 2) List Courses
 q) Quit back to main menu

Please enter your choice > 1
Please enter the course id > E
Please enter the checkpoint number (0 to exit) > 1
Please enter the checkpoint number (0 to exit) > 6
Please enter the checkpoint number (0 to exit) > 9
Please enter the checkpoint number (0 to exit) > 4
Please enter the checkpoint number (0 to exit) > 3
Please enter the checkpoint number (0 to exit) > 2
Please enter the checkpoint number (0 to exit) > 1
Please enter the checkpoint number (0 to exit) > 0

 Courses:
 1) Add new Course
 2) List Courses
 q) Quit back to main menu

Please enter your choice > 2
A 1 2 3 4 5 9 10 14 7 3 2 1
D 1 2 3 4 10 3 2 1
E 1 6 9 4 3 2 1

 Courses:
```

```
 1) Add new Course
 2) List Courses
 q) Quit back to main menu

Please enter your choice > q

 Welcome to Event Creation

 1) Create Event Info
 2) Add Entrants
 3) Add Courses
 4) Print to file
 q) Quit

Please enter your choice > 4
Please enter the folder path to create the files > data/
data/event.txt

 Welcome to Event Creation

 1) Create Event Info
 2) Add Entrants
 3) Add Courses
 4) Print to file
 q) Quit

Please enter your choice > q
```

1.4 Files Created

1.4.1 Event

```
Bob Oreillys Amazing Race
on 02/02/2013 at 07:30
```

1.4.2 Competitors

```
1 D Bugs Duggan
2 E Trevor Nelson
3 A Bob Oreilly
```

1.4.3 Courses

```
A 12 1 2 3 4 5 9 10 14 7 3 2 1
D 8 1 2 3 4 10 3 2 1
E 7 1 6 9 4 3 2 1
```

2 Checkpoint Manager

2.1 Code

2.1.1 uk/co/samsherar/cs22510/Run.java

```
package uk.co.samsherar.cs22510;

import uk.co.samsherar.cs22510.Controller.FileParser;
import uk.co.samsherar.cs22510.Controller.Manager;
/**
 * Main entry point into the program
 * @author Samuel B Sherar <sbs1@aber.ac.uk>
 */
public class Run {
  /**
   * The main method, which checks command line args and then
   * either exits with an error with printing usage, or creates the GUI
   * @param args the commandline arguments
   */
  public static void main(String[] args) {
    if(args.length != 5) {
      printUsage();
      System.exit(1);
    }

    if(FileParser.appendLog(args[4], "Started Process") == 1) {
      System.out.println("Log file is currently locked. Please try
         again later");
      System.exit(1);
    }

    Manager m =  Manager.getInstance();

    m.setFiles(args);
    m.runGUI();
    FileParser.appendLog(args[4], "Ended Process");
  }

  /**
   * Prints out the usage to the commandline
   */
  private static void printUsage() {
    System.out.println("Usage: CheckpointManager [Entrants File] [
       Courses File] [Checkpoints file] [Times File] [Log File] ");
  }
}
```

2.1.2 uk/co/samsherar/cs22510/Controller/Manager.java

```
package uk.co.samsherar.cs22510.Controller;

import uk.co.samsherar.cs22510.View.*;
```

```
import uk.co.samsherar.cs22510.Model.*;
import java.util.*;

/**
 * The overall manager for the whole program. It knows _everything_
 * @author Samuel B Sherar <sbs1@aber.ac.uk>
 */
public class Manager {
  /**
   * Singleton variable
   */
  private static Manager m = null;
  /**
   * The frame for the GUI
   */
  private MainFrame frame = null;
  /**
   * List of entrants
   */
  private LinkedList<Entrant> entrants;
  /**
   * List of courses
   */
  private LinkedList<Course> courses;
  /**
   * List of checkpoints
   */
  private LinkedList<Integer> checkpoints;
  /**
   * Time filename for later appending
   */

  private String timeFilename;

  /**
   * the path to the log file
   */
  private String logFile;

  /**
   * Singleton protected Constructor
   */
  protected Manager() {

  }

  /**
   * Checks if the GUI is already running, and if not
   * create it!
   */
  public void runGUI() {
    if(this.frame == null) {
```

```
    frame = new MainFrame();
    frame.populateEntrants(entrants);
    frame.setCheckpoints(checkpoints);
  }
}

/**
 * Parses all the data from the files inserted
 * @param filenames the array of filenames from the commandline
 */
public void setFiles(String filenames[]) {
   this.entrants = FileParser.parse_entrants(filenames[0]);
   this.courses = FileParser.parseCourses(filenames[1]);
   this.checkpoints = FileParser.parseCheckpoints(filenames[2]);
   this.populateEntrantCourses();
   FileParser.parseTimes(filenames[3], this.entrants);
   this.timeFilename = filenames[3];
   Entrant e = this.entrants.get(0);
   this.logFile = filenames[4];
}

/**
 * Singleton method to get an instance of Manager
 * @return Manager
 */
public static Manager getInstance() {
  if(m == null) {
    m = new Manager();
  }
  return m;
}

/**
 * Links a course to the entrants
 */
private void populateEntrantCourses() {
  for(Entrant entrant : this.entrants) {
    Course course = null;
    for(Course c : this.courses) {
      if(entrant.getCourseID() == c.getCourseID()) {
        course = c;
        break;
      }
    }
    if(course != null) {
      entrant.setCourse(course);
    }
  }
}

/**
 * Find entrants from their name
```

```
 * @param name the name of the entrant
 * @return the Entrant, null otherwise
 */
private Entrant findEntrant(String name) {
  Entrant ret = null;
  for(int i = 0; i < this.entrants.size(); i++) {
    if(this.entrants.get(i).getName().equals(name)) {
      ret = this.entrants.get(i);
      break;
    }
  }
  return ret;
}

/**
 * Validates the time entered from the GUI and appends it correctly
 * @param en the entrant
 * @param cp the checkpoint
 * @param arrival the arrival time
 * @param depart the depart time (normally null)
 * @return an error code (3 if the entrant has already been excluded)
 * @see uk.co.samsherar.cs22510.Controller.FileParser#appendExcluded(
    file, cp, enID, arrival)
 * @see uk.co.samsherar.cs22510.Controller.FileParser#appendStandard(
    file, cp, enID, arrival)
 * @see uk.co.samsherar.cs22510.Controller.FileParser#
    appendMedicalfile, cp, enID, arrival, depart)
 */
public int addTime(Object en, Object cp, String arrival, String
   depart) {
  int ret = 0;
  Entrant entrant = this.findEntrant((String) en);
  Integer checkpoint = (Integer) cp;
  if(entrant.isExcluded()) {
    ret = 3;
  } else {
    entrant.appendVisited(checkpoint);
    if(!entrant.onPath()) {
      entrant.setExcluded(true);
      ret = FileParser.appendExcluded(this.timeFilename, checkpoint,
         entrant.getId(), arrival);
    } else {
      if(depart.length() == 0) {
         ret = FileParser.appendStandard(this.timeFilename,
            checkpoint, entrant.getId(), arrival);
      } else {
        ret = FileParser.appendMedical(this.timeFilename, checkpoint,
            entrant.getId(), arrival, depart);
      }
    }
  }
  return ret;
```

```
  }

  public int appendLog(String action) {
    return FileParser.appendLog(this.logFile, action);
  }
}
```

2.1.3 uk/co/samsherar/cs22510/Controller/FileParser.java

```
package uk.co.samsherar.cs22510.Controller;

import java.io.*;
import java.nio.channels.FileLock;
import java.util.*;
import java.util.regex.*;
import uk.co.samsherar.cs22510.Model.*;
/**
 * A static class which parses and writes files out.
 * @author Samuel B Sherar <sbs1@aber.ac.uk>
 */
public class FileParser {
  /**
   * Regex for parsing the entrants file
   */
  private static String REGEX_ENTRANT = "([0-9]+) ([A-Z]) ([a-zA-Z ]+)
      ";
  /**
   * Regex for parsing the courses file
   */
  private static String REGEX_COURSES = "([A-Z]) [0-9]+ (.*)";
  /**
   * Regex for parsing the checkpoint file, which only accepts CP, and
       not JN
   */
  private static String REGEX_CP = "([0-9]+) CP";
  /**
   * Regex for parsing the times file
   */
  private static String REGEX_TIMES = "([A-Z]) ([0-9]+) ([0-9]+)
      ([0-9]+\\:[0-9]+)";

  /**
   * Escapes all whitespace characters, as {@link Java.util.regex.
       Pattern#Pattern()}
   * dislikes spaces for no apparent reason.
   */
  static {
    REGEX_ENTRANT.replace(" ", "\\ ");
    REGEX_COURSES.replace(" ", "\\ ");
    REGEX_CP.replace(" ", "\\ ");
    REGEX_TIMES.replace(" ", "\\ ");
  }
```

```java
/**
 * Parses entrants and returns a List of {@link uk.co.samsherar.
    cs22510.Model.Entrant}
 * to be manipulated
 * @param entrantsPath the path to the entratns file
 * @return list of Entrants
 */
public static LinkedList<Entrant> parse_entrants(String entrantsPath)
    {
  LinkedList<Entrant> ret = new LinkedList<Entrant>();
  BufferedReader br = null;
  try {
    br = new BufferedReader(new FileReader(entrantsPath));
    String currentLine;
    Pattern p = Pattern.compile(REGEX_ENTRANT);
    while((currentLine = br.readLine()) != null) {
      Matcher match = p.matcher(currentLine);
      if(match.find()) {
        int id = Integer.parseInt(match.group(1));
        char course = match.group(2).toCharArray()[0];
        String name = match.group(3);
        ret.add(new Entrant(course, name, id));
      }

    }
  } catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } finally {
    if(br != null) {
      try {
        br.close();
      } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
    }
  }

  return ret;
}

/**
 * Parses courses and returns a List of {@link uk.co.samsherar.
    cs22510.Model.Courses}
 * to be manipulated
 * @param coursePath the path to the courses file
 * @return list of courses
 */
```

```
public static LinkedList<Course> parseCourses(String coursePath) {
  LinkedList<Course> ret = new LinkedList<Course>();
  BufferedReader br = null;
  try {
    br = new BufferedReader(new FileReader(coursePath));
    String currentLine;
    Pattern p = Pattern.compile(REGEX_COURSES);
    while((currentLine = br.readLine()) != null) {
      Matcher match = p.matcher(currentLine);
      if(match.find()) {
        char course = match.group(1).toCharArray()[0];
        Course c = new Course(course);
        String[] split = match.group(2).split("\\ ");
        for(int i = 0; i < split.length; i++) {
          int tmp = Integer.parseInt(split[i]);
          c.addCheckpoint(tmp);
        }
        ret.add(c);
      }
    }
  } catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (NumberFormatException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
  return ret;
}

/**
 * Parses entrants and returns a List of {@link uk.co.samsherar.
    cs22510.Model.Entrant}
 * to be manipulated
 * @param checkpointPath the path to the checkpoint file
 * @return list of checkpoints
 */
public static LinkedList<Integer> parseCheckpoints(String
   checkpointPath) {
  LinkedList<Integer> ret = new LinkedList<Integer>();
  BufferedReader br = null;
  try {
    br = new BufferedReader(new FileReader(checkpointPath));
    String currentLine;
    Pattern p = Pattern.compile(REGEX_CP);
    while((currentLine = br.readLine()) != null) {
      Matcher match = p.matcher(currentLine);
      if(match.find()) {
        int tmp = Integer.parseInt(match.group(1));
```

```
        ret.add(tmp);
      }
    }
  } catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }

  return ret;
}
/**
 * Parses the times file, and adds the visited node to the specified
    entrant
 * @param timeFilename the filepath for the times file
 * @param entrants the list of entrants
 */
public static void parseTimes(String timeFilename, LinkedList<Entrant
   > entrants) {
  BufferedReader br = null;
  try {
    br = new BufferedReader(new FileReader(timeFilename));
    String currentLine;
    Pattern p = Pattern.compile(REGEX_TIMES);
    while((currentLine = br.readLine()) != null) {
      Matcher match = p.matcher(currentLine);
      if(match.find()) {
        char type = match.group(1).charAt(0);
        int cpID = Integer.parseInt(match.group(2));
        int entrantID = Integer.parseInt(match.group(3));
        String time = match.group(4);
        Entrant e = FileParser.findEntrant(entrants, entrantID);
        if(type == 'I') {
          e.setExcluded(true);
        } else if(e != null) {
          e.appendVisited(cpID);
        }
      }
    }
    br.close();
  } catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
}

/**
```

```
 * Finds the Entrant with a specific id in a List of entrants
 * @param entrants the list of entrants
 * @param id the id of the entrant
 * @return Entrant object if exists, null otherwise
 */
private static Entrant findEntrant(LinkedList<Entrant> entrants, int
   id) {
  for(int i = 0; i < entrants.size(); i++) {
    if(entrants.get(i).getId() == id) {
      return entrants.get(i);
    }
  }
  return null;
}

/**
 * Appends a time formatted string to the times file with the flag
 * to say that the entrant has been excluded from the rase
 * @param filename the times filename
 * @param checkpoint the checkpoint in question
 * @param entrantId the entrant id
 * @param arrival the arrival time
 * @return 1 if lock was unsuccessfull, 2 otherwise
 */
public static int appendExcluded(String filename, Integer checkpoint,
    int entrantId, String arrival) {
  try {
    FileOutputStream fos = new FileOutputStream(filename, true);
    FileLock fl = fos.getChannel().lock();
    if(fl == null) {
      return 1;
    }
    FileWriter fw = new FileWriter(fos.getFD());
    StringBuilder sb = new StringBuilder();
    sb.append("I ");
    sb.append(checkpoint + " ");
    sb.append(entrantId + " ");
    sb.append(arrival + "\n");
    fw.write(sb.toString());
    fl.release();
    fw.close();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
  return 2;

}

/**
 * Appends a time to the time file with a standard flag
 * @param filename the time filepath
```

```
 * @param checkpoint the checkpoint id
 * @param entrantId the entrant id
 * @param arrival the arrival time
 * @return 1 if the lock failed, 0 if successful
 */
public static int appendStandard(String filename, Integer checkpoint,
    int entrantId, String arrival) {
  try {
    FileOutputStream fos = new FileOutputStream(filename, true);
    FileLock fl = fos.getChannel().lock();
    if(fl == null) {
      return 1;
    }
    FileWriter fw = new FileWriter(fos.getFD());
    StringBuilder sb = new StringBuilder();
    sb.append("T ");
    sb.append(checkpoint + " ");
    sb.append(entrantId + " ");
    sb.append(arrival + "\n");
    fw.write(sb.toString());
    fl.release();
    fw.close();
  } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
  }
  return 0;

}

/**
 * Appends 2 strings to the times file, one with Arrival time, and
 * one with the departed time
 * @param filename the time filename
 * @param checkpoint the checkpoint id
 * @param entrantId the entrant id
 * @param arrival the arrival time
 * @param depart the departure time
 * @return 1 if lock failed, 0 otherwise
 */
public static int appendMedical(String filename, Integer checkpoint,
    int entrantId, String arrival, String depart) {
  try {
    FileOutputStream fos = new FileOutputStream(filename, true);
    FileLock fl = fos.getChannel().lock();
    if(fl == null) {
      return 1;
    }
    FileWriter fw = new FileWriter(fos.getFD());
    StringBuilder sbArrival = new StringBuilder();
    sbArrival.append("A ");
    sbArrival.append(checkpoint + " ");
```

```
      sbArrival.append(entrantId + " ");
      sbArrival.append(arrival + "\n");
      fw.write(sbArrival.toString());

      StringBuilder sbDepart = new StringBuilder();
      sbDepart.append("D ");
      sbDepart.append(checkpoint + " ");
      sbDepart.append(entrantId + " ");
      sbDepart.append(depart + "\n");
      fw.write(sbDepart.toString());
      fl.release();
      fw.close();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return 0;
  }

  public static int appendLog(String filename, String action) {
    try {
      FileOutputStream fos = new FileOutputStream(filename, true);
      FileLock fl = fos.getChannel().lock();
      if(fl == null) {
        return 1;
      }
      FileWriter fw = new FileWriter(fos.getFD());
      Date date = new Date();
      StringBuilder sb = new StringBuilder();
      sb.append("Checkpoint-Manager [");
      sb.append(date.toGMTString());
      sb.append("] ");
      sb.append(action);
      sb.append("\n");
      fw.write(sb.toString());
      fl.release();
      fw.close();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return 0;
  }
}
```

2.1.4 uk/co/samsherar/cs22510/Model/Course.java

```
package uk.co.samsherar.cs22510.Model;

import java.util.*;

/**
 * The course model
```

```
 * @author Samuel B Sherar <sbs1@aber.ac.uk>
 */
public class Course {
  /**
   * The course id
   */
  private char courseID;
  /**
   * List of checkpoints associated to the course id
   */
  private LinkedList<Integer> checkpoints;

  /**
   * Constructor: assigns course id and instantiates everything
   * @param courseID
   */
  public Course(char courseID) {
    this.courseID = courseID;
    this.checkpoints = new LinkedList<Integer>();
  }

  /**
   * Adds checkpoint to the list of the nodes
   * @param node the node id
   */
  public void addCheckpoint(int node) {
    this.checkpoints.add(node);
  }

  /**
   * Gets the list of checkpoints
   * @return list of checkpoints
   */
  public LinkedList<Integer> getCheckpoints() {
    return this.checkpoints;
  }

  /**
   * Gets course id
   * @return course identifier
   */
  public char getCourseID() {
    return this.courseID;
  }

}
```

2.1.5 uk/co/samsherar/cs22510/Model/Entrant.java

```
package uk.co.samsherar.cs22510.Model;

import java.util.LinkedList;
```

```java
/**
 * Model for the Entrants
 * @author Samuel B Sherar <sbs1@aber.ac.uk>
 */
public class Entrant {
  /**
   * The course identifier for the entrants
   */
  private char courseID;

  /**
   * the name of the entrant
   */
  private String name;

  /**
   * the unique identifier for the entrant
   */
  private int id;

  /**
   * the list of nodes which the entrant will visit
   */
  private Course course;

  /**
   * List of visitied nodes
   */
  private LinkedList<Integer> visited;

  /**
   * Is the entrant excluded? Time will only tell...
   */
  private boolean excluded = false;

  /**
   * Constructor: Instantiates the class variables
   * @param courseID the couse id
   * @param name the name of the entrant
   * @param id the identifier
   */
  public Entrant(char courseID, String name, int id) {
    this.courseID = courseID;
    this.name = name;
    this.visited = new LinkedList<Integer>();
  }

  public char getCourseID() {
    return courseID;
  }

  public void setCourseID(char courseID) {
```

```
  this.courseID = courseID;
}

public String getName() {
  return name;
}

public boolean isExcluded() {
  return excluded;
}

public void setExcluded(boolean excluded) {
  this.excluded = excluded;
}

public int getId() {
  return id;
}

public void setId(int id) {
  this.id = id;
}

public Course getCourse() {
  return course;
}

public void setCourse(Course course) {
  this.course = course;
}

public void setName(String name) {
  this.name = name;
}

public void appendVisited(int node) {
  this.visited.add(node);
}

/**
 * Compares the list of visited nodes to the course the entrant is on
    ,
 * and if there is any deviation, we can exclude them
 * @return if there is any deviation
 */
public boolean onPath() {
  boolean ret = true;
  LinkedList<Integer> checkpoints = this.course.getCheckpoints();
  for(int i = 0; i < this.visited.size(); i++) {
    if(this.visited.get(i) != checkpoints.get(i)) {
      ret = false;
    }
```

```
    }
    return ret;
  }

}
```

2.1.6 uk/co/samsherar/cs22510/Model/EventInfo.java

```
package uk.co.samsherar.cs22510.Model;
import java.util.*;


public class EventInfo {
  private String name;
  private Date date;
  private String time;

  /**
   *
   */
  public EventInfo() {
  }

  /**
   * @return the name
   */
  public String getName() {
    return name;
  }

  /**
   * @param name the name to set
   */
  public void setName(String name) {
    this.name = name;
  }

  /**
   * @return the date
   */
  public Date getDate() {
    return date;
  }

  /**
   * @param date the date to set
   */
  public void setDate(Date date) {
    this.date = date;
  }

  /**
   * @return the time
```

```
   */
  public String getTime() {
    return time;
  }

  /**
   * @param time the time to set
   */
  public void setTime(String time) {
    this.time = time;
  }

}
```

2.1.7 uk/co/samsherar/cs22510/View/MainFrame.java

```
package uk.co.samsherar.cs22510.View;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;

import uk.co.samsherar.cs22510.Controller.Manager;
import uk.co.samsherar.cs22510.Model.*;

import javax.swing.*;

/**
 * The main GUI. Nothing more, nothing less
 * @author Samuel B Sherar <sbs1@aber.ac.uk>
 */
public class MainFrame extends JFrame {
  /**
   * The medical checkbox
   */
  private final JCheckBox medical;

  /**
   * the departure field
   */
  private final JTextField depart;
  /**
   * the entrants combobox
   */
  private final JComboBox entrants;

  /**
   * the checkpoints combobox
   */
  private final JComboBox checkpoints;
```

```
/**
 * Draws the Frame to the screen
 */
public MainFrame() {
  this.setLayout(new GridLayout(0,2));
  setTitle("Checkpoint Manager");
  setSize(300,400);
  setDefaultCloseOperation(EXIT_ON_CLOSE);
  this.add(new JLabel("Entrants: "));
  entrants = new JComboBox();
  this.add(entrants);

  this.add(new JLabel("Checkpoints: "));
  checkpoints = new JComboBox();
  this.add(checkpoints);

  this.add(new JLabel("Medical Checkpoint? "));
  medical = new JCheckBox();
  medical.addActionListener(new ActionListener() {
    /*
     * Allows for the departure field to be enabled/disabled at click
     * @see java.awt.event.ActionListener#actionPerformed(java.awt.
        event.ActionEvent)
     */
    @Override
    public void actionPerformed(ActionEvent arg0) {

      depart.setEnabled(medical.isSelected());
    }

  });
  this.add(medical);

  this.add(new JLabel("Arrival: "));
  final JTextField arrival = new JTextField();
  this.add(arrival);

  this.add(new JLabel("Depart: "));
  depart = new JTextField();
  depart.setEnabled(false);
  this.add(depart);

  JButton add = new JButton("Add");
  add.addActionListener(new ActionListener() {
    /*
     * Scrapes the data and tries to save it using the Manager
     * @see uk.co.samsherar.cs22510.Controller.Manager#getInstance()
     * @see java.awt.event.ActionListener#actionPerformed(java.awt.
        event.ActionEvent)
     */
    @Override
```

```
    public void actionPerformed(ActionEvent arg0) {
      Manager m = Manager.getInstance();
      int ret = m.addTime(entrants.getSelectedItem(), checkpoints.
         getSelectedItem(),
            arrival.getText(), depart.getText());
      if(ret > 0) {
        String message = "";
        if(ret == 1) {
          message = "File lock didn't work. Please try again later";
        } else if (ret == 2) {
          message = "File written, but competitor is now excluded";
        } else if (ret == 3) {
          message = "Competitor is excluded";
        } else {
          message = "Time added successfully";
        }
        m.appendLog(message);
        JOptionPane.showMessageDialog(null, message);
      }
      arrival.setText("");
      depart.setText("");
    }

  });
  JButton exit = new JButton("Exit");
  exit.addActionListener(new ActionListener() {
    /*
     * Exits out of the program
     * @see java.awt.event.ActionListener#actionPerformed(java.awt.
        event.ActionEvent)
     */
    @Override
    public void actionPerformed(ActionEvent arg0) {
      System.exit(0);
    }

  });

  this.add(add);
  this.add(exit);

  this.pack();

  this.repaint();
  this.validate();
  this.setVisible(true);
}

/**
 * Populates the entrants combobox
 * @param entrants the list of entrants
 */
```

```
  public void populateEntrants(LinkedList<Entrant> entrants) {
    this.entrants.addItem("");
    for(Entrant e : entrants) {
      this.entrants.addItem(e.getName());
    }
  }

  /**
   * Populates the checkpoint combobox
   * @param checkpoints the list of checkpoints
   */
  public void setCheckpoints(LinkedList<Integer> checkpoints) {
    this.checkpoints.addItem("");
    for(int node : checkpoints) {
      this.checkpoints.addItem(node);
    }
  }
}
```

2.2 Compiler Output

```
checkpoint-manager(master*)$ avac -verbose -d bin -classpath bin -
   sourcepath src src/uk/co/samsherar/cs22510/Run.java > output.txt
[javac]\[parsing started src/uk/co/samsherar/cs22510/Run.java]
[parsing completed 7ms]
[search path for source files: src]
[search path for class files: /System/Library/Java/JavaVirtualMachines
   /1.6.0.jdk/Contents/Classes/jsfd.jar,/System/Library/Java/
   JavaVirtualMachines/1.6.0.jdk/Contents/Classes/classes.jar,/System/
   Library/Frameworks/JavaVM.framework/Frameworks/JavaRuntimeSupport.
   framework/Resources/Java/JavaRuntimeSupport.jar,/System/Library/Java
   /JavaVirtualMachines/1.6.0.jdk/Contents/Classes/ui.jar,/System/
   Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes/laf.jar
   ,/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Classes
   /sunrsasign.jar,/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/
   Contents/Classes/jsse.jar,/System/Library/Java/JavaVirtualMachines
   /1.6.0.jdk/Contents/Classes/jce.jar,/System/Library/Java/
   JavaVirtualMachines/1.6.0.jdk/Contents/Classes/charsets.jar,/System/
   Library/Java/Extensions/AppleScriptEngine.jar,/System/Library/Java/
   Extensions/dns_sd.jar,/System/Library/Java/Extensions/j3daudio.jar,/
   System/Library/Java/Extensions/j3dcore.jar,/System/Library/Java/
   Extensions/j3dutils.jar,/System/Library/Java/Extensions/jai_codec.
   jar,/System/Library/Java/Extensions/jai_core.jar,/System/Library/
   Java/Extensions/mlibwrapper_jai.jar,/System/Library/Java/Extensions/
   MRJToolkit.jar,/System/Library/Java/Extensions/QTJava.zip,/System/
   Library/Java/Extensions/vecmath.jar,/System/Library/Java/
   JavaVirtualMachines/1.6.0.jdk/Contents/Home/lib/ext/apple_provider.
   jar,/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
   /lib/ext/dnsns.jar,/System/Library/Java/JavaVirtualMachines/1.6.0.
   jdk/Contents/Home/lib/ext/localedata.jar,/System/Library/Java/
   JavaVirtualMachines/1.6.0.jdk/Contents/Home/lib/ext/sunjce_provider.
   jar,/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
   /lib/ext/sunpkcs11.jar,bin]
```

```
[loading src/uk/co/samsherar/cs22510/Controller/FileParser.java]
[parsing started src/uk/co/samsherar/cs22510/Controller/FileParser.java
   ]
[parsing completed 8ms]
[loading src/uk/co/samsherar/cs22510/Controller/Manager.java]
[parsing started src/uk/co/samsherar/cs22510/Controller/Manager.java]
[parsing completed 4ms]
[loading java/lang/Object.class(java/lang:Object.class)]
[loading java/lang/String.class(java/lang:String.class)]
[loading java/nio/channels/FileLock.class(java/nio/channels:FileLock.
   class)]
[loading java/util/LinkedList.class(java/util:LinkedList.class)]
[loading src/uk/co/samsherar/cs22510/Model/Entrant.java]
[parsing started src/uk/co/samsherar/cs22510/Model/Entrant.java]
[parsing completed 1ms]
[loading src/uk/co/samsherar/cs22510/Model/Course.java]
[parsing started src/uk/co/samsherar/cs22510/Model/Course.java]
[parsing completed 0ms]
[loading java/lang/Integer.class(java/lang:Integer.class)]
[loading src/uk/co/samsherar/cs22510/View/MainFrame.java]
[parsing started src/uk/co/samsherar/cs22510/View/MainFrame.java]
[parsing completed 2ms]
[loading java/awt/BorderLayout.class(java/awt:BorderLayout.class)]
[loading java/awt/Color.class(java/awt:Color.class)]
[loading java/awt/GridLayout.class(java/awt:GridLayout.class)]
[loading java/awt/event/ActionEvent.class(java/awt/event:ActionEvent.
   class)]
[loading java/awt/event/ActionListener.class(java/awt/event:
   ActionListener.class)]
[loading javax/swing/JFrame.class(javax/swing:JFrame.class)]
[loading javax/swing/WindowConstants.class(javax/swing:WindowConstants.
   class)]
[loading javax/accessibility/Accessible.class(javax/accessibility:
   Accessible.class)]
[loading javax/swing/RootPaneContainer.class(javax/swing:
   RootPaneContainer.class)]
[loading javax/swing/TransferHandler.class(javax/swing:TransferHandler.
   class)]
[loading javax/swing/TransferHandler$HasGetTransferHandler.class(javax/
   swing:TransferHandler$HasGetTransferHandler.class)]
[loading java/io/Serializable.class(java/io:Serializable.class)]
[loading java/awt/Frame.class(java/awt:Frame.class)]
[loading java/awt/MenuContainer.class(java/awt:MenuContainer.class)]
[loading java/awt/Window.class(java/awt:Window.class)]
[loading java/awt/Container.class(java/awt:Container.class)]
[loading java/awt/Component.class(java/awt:Component.class)]
[loading java/awt/image/ImageObserver.class(java/awt/image:
   ImageObserver.class)]
[loading javax/swing/JCheckBox.class(javax/swing:JCheckBox.class)]
[loading javax/swing/JTextField.class(javax/swing:JTextField.class)]
[loading javax/swing/JComboBox.class(javax/swing:JComboBox.class)]
[checking uk.co.samsherar.cs22510.Run]
```

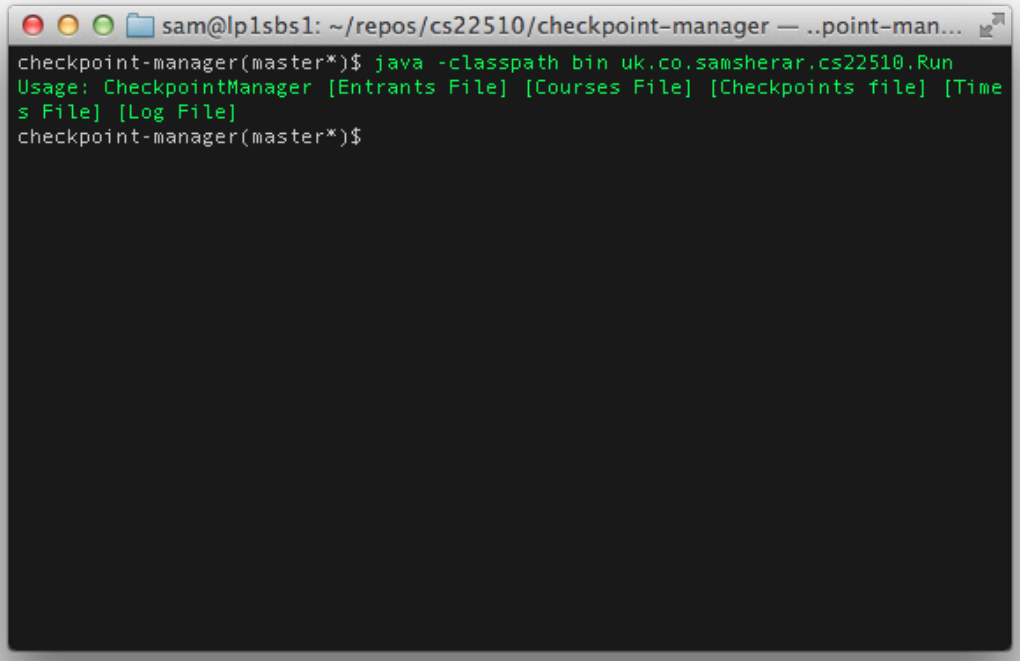
```
[loading java/lang/System.class(java/lang:System.class)]
[loading java/io/PrintStream.class(java/io:PrintStream.class)]
[loading java/io/FilterOutputStream.class(java/io:FilterOutputStream.
   class)]
[loading java/io/OutputStream.class(java/io:OutputStream.class)]
[wrote bin/uk/co/samsherar/cs22510/Run.class]
[checking uk.co.samsherar.cs22510.Controller.FileParser]
[loading java/lang/CharSequence.class(java/lang:CharSequence.class)]
[loading java/lang/Comparable.class(java/lang:Comparable.class)]
[loading java/util/Collection.class(java/util:Collection.class)]
[loading java/util/AbstractSequentialList.class(java/util:
   AbstractSequentialList.class)]
[loading java/util/AbstractList.class(java/util:AbstractList.class)]
[loading java/util/AbstractCollection.class(java/util:
   AbstractCollection.class)]
[loading java/io/BufferedReader.class(java/io:BufferedReader.class)]
[loading java/io/FileReader.class(java/io:FileReader.class)]
[loading java/io/FileDescriptor.class(java/io:FileDescriptor.class)]
[loading java/io/File.class(java/io:File.class)]
[loading java/io/InputStreamReader.class(java/io:InputStreamReader.
   class)]
[loading java/io/InputStream.class(java/io:InputStream.class)]
[loading java/io/Reader.class(java/io:Reader.class)]
[loading java/util/regex/Pattern.class(java/util/regex:Pattern.class)]
[loading java/util/regex/Matcher.class(java/util/regex:Matcher.class)]
[loading java/lang/Number.class(java/lang:Number.class)]
[loading java/io/FileNotFoundException.class(java/io:
   FileNotFoundException.class)]
[loading java/io/IOException.class(java/io:IOException.class)]
[loading java/lang/Exception.class(java/lang:Exception.class)]
[loading java/lang/Throwable.class(java/lang:Throwable.class)]
[loading java/lang/NumberFormatException.class(java/lang:
   NumberFormatException.class)]
[loading java/lang/IllegalArgumentException.class(java/lang:
   IllegalArgumentException.class)]
[loading java/lang/RuntimeException.class(java/lang:RuntimeException.
   class)]
[loading java/io/FileOutputStream.class(java/io:FileOutputStream.class)
   ]
[loading java/nio/channels/FileChannel.class(java/nio/channels:
   FileChannel.class)]
[loading java/nio/channels/ByteChannel.class(java/nio/channels:
   ByteChannel.class)]
[loading java/nio/channels/ReadableByteChannel.class(java/nio/channels:
   ReadableByteChannel.class)]
[loading java/nio/channels/Channel.class(java/nio/channels:Channel.
   class)]
[loading java/io/Closeable.class(java/io:Closeable.class)]
[loading java/nio/channels/WritableByteChannel.class(java/nio/channels:
   WritableByteChannel.class)]
[loading java/nio/channels/GatheringByteChannel.class(java/nio/channels
   :GatheringByteChannel.class)]
```

```
[loading java/nio/channels/ScatteringByteChannel.class(java/nio/
   channels:ScatteringByteChannel.class)]
[loading java/nio/channels/spi/AbstractInterruptibleChannel.class(java/
   nio/channels/spi:AbstractInterruptibleChannel.class)]
[loading java/nio/channels/InterruptibleChannel.class(java/nio/channels
   :InterruptibleChannel.class)]
[loading java/io/FileWriter.class(java/io:FileWriter.class)]
[loading java/io/OutputStreamWriter.class(java/io:OutputStreamWriter.
   class)]
[loading java/io/Writer.class(java/io:Writer.class)]
[loading java/lang/StringBuilder.class(java/lang:StringBuilder.class)]
[loading java/lang/AbstractStringBuilder.class(java/lang:
   AbstractStringBuilder.class)]
[loading java/lang/StringBuffer.class(java/lang:StringBuffer.class)]
[loading java/util/Date.class(java/util:Date.class)]
[loading java/lang/Error.class(java/lang:Error.class)]
[loading java/lang/Byte.class(java/lang:Byte.class)]
[loading java/lang/Character.class(java/lang:Character.class)]
[loading java/lang/Short.class(java/lang:Short.class)]
[loading java/lang/Long.class(java/lang:Long.class)]
[loading java/lang/Float.class(java/lang:Float.class)]
[loading java/lang/Double.class(java/lang:Double.class)]
[loading java/lang/Boolean.class(java/lang:Boolean.class)]
[loading java/lang/Void.class(java/lang:Void.class)]
[wrote bin/uk/co/samsherar/cs22510/Controller/FileParser.class]
[checking uk.co.samsherar.cs22510.Model.Entrant]
[wrote bin/uk/co/samsherar/cs22510/Model/Entrant.class]
[checking uk.co.samsherar.cs22510.Model.Course]
[wrote bin/uk/co/samsherar/cs22510/Model/Course.class]
[checking uk.co.samsherar.cs22510.Controller.Manager]
[loading java/lang/Iterable.class(java/lang:Iterable.class)]
[loading java/util/Iterator.class(java/util:Iterator.class)]
[wrote bin/uk/co/samsherar/cs22510/Controller/Manager.class]
[checking uk.co.samsherar.cs22510.View.MainFrame]
[loading java/awt/LayoutManager.class(java/awt:LayoutManager.class)]
[loading java/awt/Dimension.class(java/awt:Dimension.class)]
[loading javax/swing/JLabel.class(javax/swing:JLabel.class)]
[loading javax/swing/Icon.class(javax/swing:Icon.class)]
[loading javax/swing/JComponent.class(javax/swing:JComponent.class)]
[loading java/awt/PopupMenu.class(java/awt:PopupMenu.class)]
[loading javax/swing/JToggleButton.class(javax/swing:JToggleButton.
   class)]
[loading javax/swing/AbstractButton.class(javax/swing:AbstractButton.
   class)]
[loading java/util/EventListener.class(java/util:EventListener.class)]
[loading java/lang/Override.class(java/lang:Override.class)]
[loading java/lang/annotation/Annotation.class(java/lang/annotation:
   Annotation.class)]
[loading java/lang/annotation/Target.class(java/lang/annotation:Target.
   class)]
[loading java/lang/annotation/ElementType.class(java/lang/annotation:
   ElementType.class)]
```

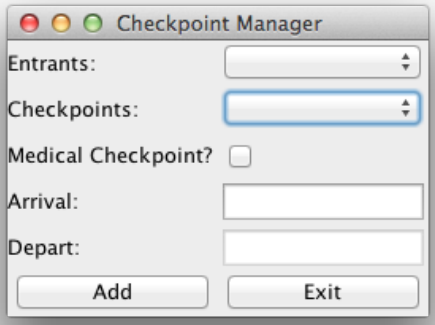
```
[loading java/lang/annotation/Retention.class(java/lang/annotation:
   Retention.class)]
[loading java/lang/annotation/RetentionPolicy.class(java/lang/
   annotation:RetentionPolicy.class)]
[loading javax/swing/text/JTextComponent.class(javax/swing/text:
   JTextComponent.class)]
[loading javax/swing/JButton.class(javax/swing:JButton.class)]
[loading javax/swing/Action.class(javax/swing:Action.class)]
[loading javax/swing/JOptionPane.class(javax/swing:JOptionPane.class)]
[loading java/awt/HeadlessException.class(java/awt:HeadlessException.
   class)]
[loading java/lang/UnsupportedOperationException.class(java/lang:
   UnsupportedOperationException.class)]
[loading java/util/Set.class(java/util:Set.class)]
[loading java/lang/Class.class(java/lang:Class.class)]
[wrote bin/uk/co/samsherar/cs22510/View/MainFrame$1.class]
[wrote bin/uk/co/samsherar/cs22510/View/MainFrame$2.class]
[wrote bin/uk/co/samsherar/cs22510/View/MainFrame$3.class]
[wrote bin/uk/co/samsherar/cs22510/View/MainFrame.class]
[total 476ms]
Note: src/uk/co/samsherar/cs22510/Controller/FileParser.java uses or
   overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

2.3 Screen Shots

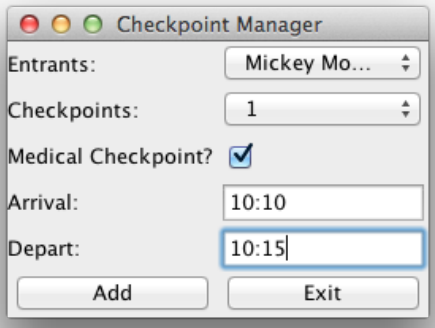
2.3.1 No inputs



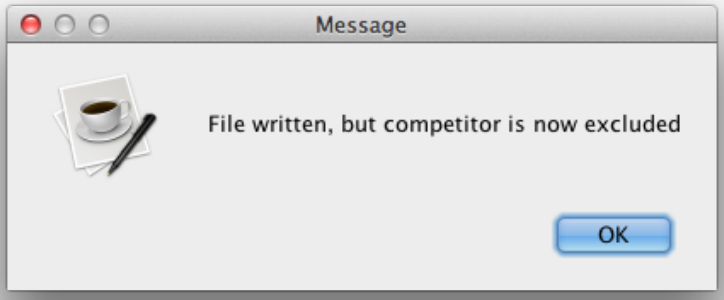
2.3.2 Main GUI



2.3.3 Medical Checkpoint



2.3.4 Feedback



2.3.5 File locking

```
checkpoint-manager(master*)$ chflags uchg ../event-manager/data/log.txt
checkpoint-manager(master*)$ java -classpath bin uk.co.samsherar.
    cs22510.Run data/comp_data.txt data/courses.txt data/node.txt data/
    cp_times_1.txt ../event-manager/data/log.txt
Log file is currently locked. Please try again later
checkpoint-manager(master*)$
```

[1]

[1] I have used chflags to keep the file locked for the whole time I am needing it, so it's easier for me to test that it is working

3 Event Manager

3.1 Compiler Output

```
event-manager(master*)$ make clean && make
rm -rf bin/*
gcc -g -Wall  fileio.c main.c node.c log.c -o bin/run
fileio.c: In function  load_courses_file :
fileio.c:156: warning: assignment makes integer from pointer without a
   cast
fileio.c:160: warning: assignment from incompatible pointer type
fileio.c:161: warning: assignment from incompatible pointer type
fileio.c:165: warning: assignment from incompatible pointer type
fileio.c: In function  load_time_file :
fileio.c:219: warning: passing argument 1 of
    insert_checkpoint_data  from incompatible pointer type
main.c: In function   menu  :
main.c:110: warning: comparison between pointer and integer
main.c:113: warning: comparison between pointer and integer
main.c:114: warning: comparison between pointer and integer
main.c:117: warning: format   %s   expects type  char *   , but
   argument 3 has type  int *
main.c:117: warning: format   %s   expects type  char *   , but
   argument 3 has type  int *
main.c:156: warning: implicit declaration of function
    find_disq_cp
main.c:160: warning: implicit declaration of function
    find_disq_medical
main.c:175: warning: passing argument 1 of
    insert_checkpoint_data_manually  from incompatible pointer type
main.c: In function  find_not_started :
main.c:206: warning: comparison between pointer and integer
main.c: In function  find_running :
main.c:224: warning: comparison between pointer and integer
main.c:225: warning: comparison between pointer and integer
main.c: In function  find_finished :
main.c:243: warning: comparison between pointer and integer
main.c:244: warning: comparison between pointer and integer
main.c: In function  find_disq_cp :
main.c:260: warning: statement with no effect
main.c: In function  find_disq_medical :
main.c:277: warning: statement with no effect
main.c: In function   startup  :
main.c:301: warning: implicit declaration of function
    load_info_file
main.c:293: warning: unused variable  log_filename
main.c: In function  print_competitor :
main.c:372: warning: format   %-15s   expects type  char *   , but
   argument 5 has type  int *
main.c:372: warning: format   %-15s   expects type  char *   , but
   argument 6 has type  int *
main.c:372: warning: format   %-15s   expects type  char *   , but
```

```
   argument 5 has type   int *
main.c:372: warning: format    %-15 s    expects type   char *    , but
   argument 6 has type   int *
node.c: In function  check_next_empty :
node.c:80: warning: comparison between pointer and integer
node.c: In function  find_current_node :
node.c:99: warning: comparison between pointer and integer
node.c: In function  insert_checkpoint_data :
node.c:132: warning: passing argument 1 of  find_node_head  from
   incompatible pointer type
node.c:134: warning: passing argument 1 of  check_next_empty  from
   incompatible pointer type
node.c: In function  insert_checkpoint_data_manually :
node.c:156: warning: passing argument 1 of  find_node_head  from
   incompatible pointer type
node.c: In function  calc_total_time :
node.c:172: warning: comparison between pointer and integer
node.c:172: warning: comparison between pointer and integer
node.c:179: warning: passing argument 2 of  __builtin___strcpy_chk
    from incompatible pointer type
node.c:179: warning: passing argument 2 of  __inline_strcpy_chk
   from incompatible pointer type
node.c:180: warning: passing argument 2 of  __builtin___strcpy_chk
    from incompatible pointer type
node.c:180: warning: passing argument 2 of  __inline_strcpy_chk
   from incompatible pointer type
node.c:193: warning: function returns address of local variable
node.c: In function  find_current_track :
node.c:221: warning: control reaches end of non-void function
node.c: In function  find_track :
node.c:234: warning: statement with no effect
node.c:241: warning: control reaches end of non-void function
node.c: In function  find_comp_index :
node.c:266: warning: statement with no effect
```

3.2 Output Generated & Results

```
event-manager(master*)$ bin/run
Please enter the file for the logging > data/log.txt
Please enter the file for the event information > data/event_info.txt
Please enter the file for the node type > data/node.txt
Please enter the file for the competitors > data/comp_data.txt
Please enter the file for the courses > data/courses.txt
Please enter the file for the tracks > data/tracks.txt


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
```

```
  5)  List competitors with times
  q)  Exit the application

 > 2


  1)  Query how many which haven't started
  2)  Query how many people are on the courses
  3)  Query how many people have finished
  4)  Query how many have been disqualified for wrong checkpoint
  5)  Query how many have been disqualified for medical reasons

 > 1
Number of competitors not started: 14


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > 4


Please enter the data file for the checkpoints > data/cp_times_1.txt
Node 1 was hit at 07:30 by competitor 0
Node 1 was hit at 07:35 by competitor 1
Node 1 was hit at 07:39 by competitor 2
Node 1 was hit at 07:43 by competitor 3
Node 1 was hit at 07:47 by competitor 4
Node 1 was hit at 07:51 by competitor 5
Node 1 was hit at 07:56 by competitor 6
Node 1 was hit at 08:01 by competitor 7
Node 1 was hit at 08:05 by competitor 8
Node 4 was hit at 08:09 by competitor 0
Node 1 was hit at 08:10 by competitor 9
Node 4 was hit at 08:11 by competitor 1
Node 1 was hit at 08:14 by competitor 10
Node 1 was hit at 08:18 by competitor 11
Node 9 was hit at 08:20 by competitor 2
Node 1 was hit at 08:22 by competitor 12
Node 5 was hit at 08:22 by competitor 0
Node 5 was hit at 08:25 by competitor 1
Node 1 was hit at 08:26 by competitor 13
Node 9 was hit at 08:33 by competitor 3
Node 4 was hit at 08:35 by competitor 6
Node 9 was hit at 08:35 by competitor 4
```

```
Node 9 was hit at 08:37 by competitor 5
Node 4 was hit at 08:42 by competitor 7
Node 9 was hit at 08:46 by competitor 0
Node 5 was hit at 08:48 by competitor 6
Node 9 was hit at 08:49 by competitor 1
Node 9 was hit at 08:49 by competitor 8
Node 9 was hit at 08:54 by competitor 9


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > 2


  1)  Query how many which haven't started
  2)  Query how many people are on the courses
  3)  Query how many people have finished
  4)  Query how many have been disqualified for wrong checkpoint
  5)  Query how many have been disqualified for medical reasons

 > 1
Number of competitors not started: 0


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > 2


  1)  Query how many which haven't started
  2)  Query how many people are on the courses
  3)  Query how many people have finished
  4)  Query how many have been disqualified for wrong checkpoint
  5)  Query how many have been disqualified for medical reasons

 > 2
```

```
Number of competitors running: 14


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > 5


Name                            ID          Course ID       Start Time
        End Time        Total Time
Donald Duck                     1           D               07:30
Mickey Mouse                    2           D               07:35
Jemima Julieta Mouse            3           E               07:39
Minnie Duck                     4           F               07:43
Minnie Mouse                    5           E               07:47
Minnie Mouse Junior             6           E               07:51
Deputy Doug                     7           D               07:56
Deputy Duck                     8           D               08:01
Bewick Swan                     9           F               08:05
Black Swan                      10          F               08:10
Albert Einstein                 11          E               08:14
Albert Mouse                    12          D               08:18
Donald Duck Senior              13          E               08:22
Egbert Einstein                 14          F               08:26


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > 4


Please enter the data file for the checkpoints > data/cp_data_2.txt
Node 13 was hit at 08:56 by competitor 3
Node 9 was hit at 08:56 by competitor 10
Node 13 was hit at 08:57 by competitor 2
Node 5 was hit at 08:57 by competitor 7
```

```
Node 4 was hit at 08:59 by competitor 11
Node 9 was hit at 09:10 by competitor 12
Node 13 was hit at 09:12 by competitor 8
Node 13 was hit at 09:13 by competitor 5
Node 5 was hit at 09:14 by competitor 11
Node 9 was hit at 09:14 by competitor 6
Node 13 was hit at 09:15 by competitor 4
Node 9 was hit at 09:15 by competitor 13
Node 13 was hit at 09:17 by competitor 9
Node 9 was hit at 09:23 by competitor 7
Node 1 was hit at 09:31 by competitor 0
Node 1 was hit at 09:34 by competitor 1
Node 13 was hit at 09:37 by competitor 10
Node 13 was hit at 09:37 by competitor 13
Node 1 was hit at 09:42 by competitor 2
Node 9 was hit at 09:42 by competitor 11
Node 13 was hit at 09:45 by competitor 12
Node 1 was hit at 09:46 by competitor 3
Node 1 was hit at 09:56 by competitor 8
Node 1 was hit at 09:58 by competitor 5
Node 1 was hit at 10:02 by competitor 6
Node 1 was hit at 10:05 by competitor 4
Node 1 was hit at 10:09 by competitor 9
Node 1 was hit at 10:11 by competitor 7
Node 1 was hit at 10:20 by competitor 13
Node 1 was hit at 10:23 by competitor 11
Node 1 was hit at 10:26 by competitor 10
Node 1 was hit at 10:33 by competitor 12


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > 5


Name                            ID          Course ID       Start Time
        End Time        Total Time
Donald Duck                     1           D               07:30
             09:31           02:01
Mickey Mouse                    2           D               07:35
             09:34           01:59
Jemima Julieta Mouse            3           E               07:39
             09:42           02:03
Minnie Duck                     4           F               07:43
```

```
             09:46           02:03
Minnie Mouse                   5          E                07:47
             10:05           02:18
Minnie Mouse Junior            6          E                07:51
             09:58           02:07
Deputy Doug                    7          D                07:56
             10:02           02:06
Deputy Duck                    8          D                08:01
             10:11           02:10
Bewick Swan                    9          F                08:05
             09:56           01:51
Black Swan                     10         F                08:10
             10:09           01:59
Albert Einstein                11         E                08:14
             10:26           02:12
Albert Mouse                   12         D                08:18
             10:23           02:05
Donald Duck Senior             13         E                08:22
             10:33           02:11
Egbert Einstein                14         F                08:26
             10:20           01:54


Welcome to Endurance Horse Race - Beginners Event on 26th June 2012
   07:30!

  1)  Query Location of Competitor
  2)  Query status of competitors
  3)  Supply times for individual competitor
  4)  Read checkpoint data from file
  5)  List competitors with times
  q)  Exit the application

 > q
```

3.3 Log File

```
[Thu Mar 21 21:25:12 2013] Hello World
[Thu Mar 21 22:17:55 2013] Hello World
[Thu Mar 21 22:19:11 2013] Hello World
Checkpoint-Manager [22 Mar 2013 04:51:49 GMT] Started Process
Checkpoint-Manager [22 Mar 2013 04:51:50 GMT] Ended Process
Checkpoint-Manager [22 Mar 2013 04:52:03 GMT] Started Process
Checkpoint-Manager [22 Mar 2013 04:52:03 GMT] Ended Process
Checkpoint-Manager [22 Mar 2013 04:52:12 GMT] Started Process
Checkpoint-Manager [22 Mar 2013 04:52:13 GMT] Ended Process
Checkpoint-Manager [22 Mar 2013 04:52:48 GMT] Competitor is excluded
Checkpoint-Manager [22 Mar 2013 04:53:03 GMT] File written, but
   competitor is now excluded
Checkpoint-Manager [22 Mar 2013 05:12:25 GMT] Started Process
Checkpoint-Manager [22 Mar 2013 05:12:26 GMT] Ended Process
Checkpoint-Manager [22 Mar 2013 05:13:27 GMT] Competitor is excluded
Checkpoint-Manager [22 Mar 2013 05:26:59 GMT] Started Process
```

```
Checkpoint-Manager [22 Mar 2013 05:26:59 GMT] Ended Process
Event-Manager: [Fri Mar 22 05:50:04 2013] Started Process
Event-Manager: [Fri Mar 22 05:50:13 2013] Querying Status of
   Competitors
Event-Manager: [Fri Mar 22 05:51:39 2013] Ended Process
Event-Manager: [Fri Mar 22 05:57:19 2013] Started Process
Event-Manager: [Fri Mar 22 05:57:22 2013] Querying Status of
   Competitors
Event-Manager: [Fri Mar 22 06:00:14 2013] Started Process
Event-Manager: [Fri Mar 22 06:00:20 2013] Querying Status of
   Competitors
Event-Manager: [Fri Mar 22 06:00:32 2013] Loading time file
Event-Manager: [Fri Mar 22 06:00:35 2013] Querying Status of
   Competitors
Event-Manager: [Fri Mar 22 06:00:39 2013] Querying Status of
   Competitors
Event-Manager: [Fri Mar 22 06:01:00 2013] Loading time file
Event-Manager: [Fri Mar 22 06:01:04 2013] Ended Process
```

4 Descriptions

4.1 Event Creator

For the Event Creator, I decided to use C++. This was because of the great libraries I could use for parsing data on the commandline, while not having to rewrite the Event Manager in a different language. If I were to start from afresh, I would have written the Event Creator in C, mainly because it isn't as powerful as C++.

I also designed it to use a Model-View-Controller meta-pattern in this program to an extent, so it's easy to edit and modify if/when an update needs to be rolled out.

4.2 Event Manager

For the Event Manager, I decided to not reinvent the wheel and keep it in C. However, I created a new file for the logging and file locking, so there won't be any conflicts with the current code. For the file locking, I have used the example given to us, but with an added method using `F_GETLK` , which will not set the lock, but returns -1 if there is a lock currently on the file.

4.3 Checkpoint Manager

For the Checkpoint Manager, I decided to use Java. This is because I am comfortable creating good GUIs in little time, which will be consistant

I also decided to use the Model-View-Control meta-pattern as it seems the most appropiate for a GUI based application, and it can easily be updated upon in the future. I also have used a Singleton design pattern for the main Controller of the program, which will allow for a single static datasource between all different packages. If the program was any larger, I would have opted for an Observable pattern across, but it seemed a little overpowered for what was needed for this assignment.