CS22510 Assignment 1
Runners and Riders
"Out and About"

Chris Savill
chs17@aber.ac.uk

March 20, 2013

Contents

1 Description of three programs

1.1 Event Creation Program

Implemented in C++, this program was the simplest program to implement. Designing the program was quite easy as well due to how well this program suited an object oriented programming language such as C++. As the program was required to create an event with courses and competitors it made sense to make each of these a class with their related data members and member functions encapsulated together within their respective classes.

As the program's main purpose involved getting the user to input the details of the event, the courses, and the competitors, I decided to make the input processes robust by using checks and getting the user to double check their inputs. Although this may make the process long-winded I think the purpose they serve is worth the time. An example of a check I implemented involves the validating of the hours and minutes of a time entered to make sure the hours are between 0 and 23 inclusive and the minutes are between 0 and 59 inclusive then asking the user if they are happy with the final time.

Another feature to improve robustness was to make sure that a user could not create a competitor without at least one course, otherwise competitors could be assigned a course that does not exist. With relation to assumptions, one assumption I made was that the tracks had no role in this program and that the tracks are generated/decided elsewhere once the courses have been created. This made generating the courses simpler as the only real checks needed were that valid nodes were selected and that the start and end nodes matched.

1.2 Checkpoint Manager Program

Implemented in Java this mainly GUI driven program took a bit of thinking with regards to the design. The main issue was figuring out how much information the program would require and the processing needed to validate a record. I made the assumption that records would arrive in chronological order to simplify the program and avoiding having to rewrite history. This meant that the program would only have to append to the time record file and only had to read in what it did not contain since it's last read.

With relation to making the program robust, I found it hard to cover every scenario that could be encountered so making the assumption that records would be entered in chronological order made things a lot easier. To make the program more robust I focused on the GUI design, keeping it simple and only supplying options to the user which would be easy to validate and reduce chances of error. By getting the type of checkpoint first, the program is able to filter the list of checkpoints in the next window thus getting rid of the user's need to check the type of checkpoint. Also there is a prompt telling the user they need to select both a checkpoint and competitor if only one was selected.

After getting the record details from the user I had to think about how to validate the new record and how to decide whether or not to append the new record to the time record file. I noticed that the program would need to keep track of every competitor's status and their current progress during the event. Using that information would allow the program to evaluate whether or not a new record is valid such as the program checking if the a competitor can be at a new checkpoint if the status tracked says that the competitor is currently at a medical checkpoint thus invalidating the new record.

When reading in the start up files at launch I implemented regular expressions for each file to make sure that every line in the files provided the right information. This also helped ensure

that the correct data was being inserted into the relevant data structures I implemented. If any of the lines in any of the files fail their respective regular expressions, a message is printed to the user stating there was an invalid line format and the program closes.

1.3 Event Manager Program

Implemented in C this program did not require much work as I used the same program I implemented but removed the function to manually supply a time for a competitor and added the file locking capability to the accessing of the time record files and the implementation of writing a log file of the user's actions. As the program worked extremely well already I decided not to change any of the functionality just add the 2 new features and remove the 1 no longer needed.

The event manager program I implemented tries to estimate a competitor's location on track as well as knowing their actual last checked-in location. The program was also implemented with the assumption made that files and records would arrive in chronological order. This means that if a record arrives that is not in chronological order, the program will message the user telling them the file is chronologically correct and won't be loaded. This is to prevent any segmentation faults from occurring as a lot of the functionality in the program relies on everything arriving in chronological order. The program also has regular expressions/checks in place to check the input from the files to help detect errors and prevent them from effecting the running of the program.

2 Code for the Event Creation Program

2.1 Header files

Listing 1: Header file for non-class specific functions.

```
1  /*
2   * Author: Chris Savill, chs17@aber.ac.uk
3   * File Name: creator.h
4   * Description: Header file for the starter function declarations.
5   * First Created: 11/03/2013
6   * Last Modified: 14/03/2013
7   */
8
9  #ifndef CREATOR_H
10 #define CREATOR_H
11
12 #include <memory>
13 #include "event.h"
14
15 bool get_acceptance(); //Function to get the user's input for accepting or
       rejecting their inputs.
16 bool checkCourseExists(char letter, Event *event); //Member function that
       checks if the letter given be the user matches any of the course letters.
17 void ecp_menu(Event *event); //Function that launches the event creation
       program menu.
18
19 #endif  /* CREATOR_H */
```

Listing 2: Header file Event class.

```
1  /*
2   * Author: Chris Savill, chs17@aber.ac.uk
3   * File Name: event.h
4   * Description: Header file for the Event class.
5   * First Created: 11/03/2013
6   * Last Modified: 14/03/2013
7   */
8
9  #ifndef EVENT_H
10 #define EVENT_H
11
12 #include <memory>
13 #include "competitor.h"
14 #include "course.h"
15 #include <vector>
16 #include <cstdlib>
17 #include <iostream>
18
19 #define MAX_EVENT_NAME_LENGTH 79
20 #define MAX_DATE_LENGTH 19
21
22 class Competitor;
23 class Course;
24
25 class Event {
26 private:
27     std::string name; //Name of the event.
28     std::string date; //Date of the event.
29     std::string start_time; //Start time of the event.
```

```
30      std::vector<Competitor*> *competitors; //Array of competitors to take
            part in the event.
31      std::vector<Course*> *courses; //Array of courses that are part of an
            event.
32
33      void set_name(); //Member function to get the user to input the events
            name.
34      void set_date(); //Member function to get the user to input the date of
            the event.
35      void set_start_time(); //Member function to get the user to input the
            start time of the event.
36
37  public:
38      Event();
39      ~Event();
40      std::vector<Course*>* getCourses(); //Member function that returns a
            pointer to the vector of courses.
41      void add_competitor(); //Member function that will handle adding a
            competitor to the event.
42      void add_course(); //Member function that will handle adding a course to
             the event.
43      void export_event(); //Member function that will handle exporting the
            name, date and start_time of the event to a '.txt' file.
44      void export_competitors(); //Member function that will handle the
            exporting of the array of competitors to a '.txt' file.
45      void export_courses(); //Member function that will handle the exporting
            of the array of courses to a '.txt' file.
46  };
47
48  #endif  /* EVENT_H */
```

Listing 3: Header file for Course class.

```
1   /*
2    * Author: Chris Savill, chs17@aber.ac.uk
3    * File Name: course.h
4    * Description: Header file for the Course class.
5    * First Created: 11/03/2013
6    * Last Modified: 14/03/2013
7    */
8
9   #ifndef COURSE_H
10  #define COURSE_H
11
12  #include <memory>
13  #include <vector>
14
15  class Event;
16
17  class Course {
18  private:
19      char letter; //The courses unique identification letter for an event.
20      int number_of_nodes; //The number of nodes the course contains.
21      std::vector<int> *nodes; //An array of nodes that are contained in the
            course.
22      std::vector<int> *nodes_available; //An array of nodes that are
            available to select from, read in from the 'nodes.txt' file.
23
24      void set_letter(Event *event); //Member function that will set the
            letter of the course.
25      void set_number_of_nodes(); //Member function that will set the number
            of nodes of the course.
```

```
26      bool read_nodes_available(); //Member function that reads in the nodes
            from the 'nodes.txt' file and adds them to the nodes available array.
27      void add_node(); //Member function that adds a new node to the course.
28      bool duplicated_last_node(int number); //Member function to check if the
             new node being selected matches the last node added.
29      bool check_node_exists(int number); //Member function that checks that
            the node being added exists in the array of nodes available.
30
31  public:
32      char get_letter(); //Member function to return a course's letter.
33      int get_number_of_nodes(); //Member function to return a course's number
             of nodes.
34      int get_node(int index); //Member function to return a node from the
            course's vector of nodes.
35      Course(Event *event);
36      ~Course();
37  };
38
39  #endif  /* COURSE_H */
```

Listing 4: Header file for Competitor class.

```
1   /*
2    * Author: Chris Savill, chs17@aber.ac.uk
3    * File Name: competitor.h
4    * Description: Header file for the Competitor class.
5    * First Created: 11/03/2013
6    * Last Modified: 14/03/2013
7    */
8
9   #ifndef COMPETITOR_H
10  #define COMPETITOR_H
11
12  #include <memory>
13  #include <string>
14
15  #define MAX_COMPETITOR_NAME_LENGTH 51 //Includes null terminator \0.
16
17  class Event;
18
19  class Competitor {
20  private:
21      int number; //The competitor's unique identification number for an event
            .
22      std::string name; //The competitor's name.
23      char course; //The course letter the competitor is entering in for.
24
25      void set_number(int number); //Member function that will set the number
            of the competitor.
26      void set_name(); //Member function that will set the name of the
            competitor.
27      void set_course(Event *event); //Member function that will set the
            course letter for the competitor.
28
29  public:
30      int get_number(); //Member function to return a competitor's number.
31      std::string get_name(); //Member function to return a competitor's name.
32      char get_course(); //Member function to return a competitor's course.
33      Competitor(int number, Event *event);
34  };
35
36  #endif  /* COMPETITOR_H */
```

2.2 Cpp files

Listing 5: Main method and menu file.

```
1  /*
2   * Author: Chris Savill, chs17@aber.ac.uk
3   * File Name: main.cpp
4   * Description: cpp file that contains function definitions for the start-up
        of the event creation program.
5   * First Created: 11/03/2013
6   * Last Modified: 14/03/2013
7   */
8
9  #include "creator.h"
10 #include <iostream>
11 #include <cstdlib>
12 #include <limits>
13
14 using namespace std;
15
16 /* Main function that just calls a function that takes over. */
17 int main(int argc, char** argv) {
18     Event *event = new Event();
19     ecp_menu(event);
20
21     return 0;
22 }
23
24 /* Function to get the user's input for accepting or rejecting their inputs.
       */
25 bool get_acceptance() {
26     char option;
27
28     do {
29         cout << "If yes press 'y' then 'Enter'" << endl << "If no press 'n'
               then 'Enter'" << endl;
30         cin.clear();
31         option = cin.get();
32         cin.ignore(numeric_limits<streamsize>::max(), '\n');
33
34         if (option == 'y') return true;
35         else if (option == 'n') return false;
36         else cout << "Invalid option selected" << endl;
37     } while (option != 'y' && option != 'n');
38 }
39
40 /* Function that displays the main menu for the event creation program. */
41 void ecp_menu(Event *event) {
42     int option; //Field to store the user's option input.
43
44     do {
45         cout << "*************************************************************"
               << endl;
46         cout << "*  Runners and Riders Event Creation Program Main Menu  *"
               << endl;
47         cout << "*************************************************************"
               << endl;
48         cout << "*                1. Add Competitor to Event                 *"
               << endl;
49         cout << "*                2. Add Course to Event                     *"
               << endl;
50         cout << "*                3. Export Event to File                    *"
```

```
            << endl;
51          cout << "*              4. Export Competitors to File            *"
            << endl;
52          cout << "*              5. Export Courses to File                *"
            << endl;
53          cout << "*              6. Exit Event Creation Program           *"
            << endl;
54          cout << "*********************************************************"
            << endl << endl;
55
56          cout << "Please enter in an option from the above an press 'Enter':
            ";
57          cin.clear();
58          cin >> option;
59          cin.ignore();
60
61          switch (option) {
62              case 1:
63                  event->add_competitor();
64                  break;
65              case 2:
66                  event->add_course();
67                  break;
68              case 3:
69                  event->export_event();
70                  break;
71              case 4:
72                  event->export_competitors();
73                  break;
74              case 5:
75                  event->export_courses();
76                  break;
77              case 6:
78                  delete(event);
79                  cout << "Exiting program..." << endl << endl;
80                  break;
81              default:
82                  cout << "Please enter in a valid option." << endl << endl;
83          }
84      } while (option != 6);
85  }
```

Listing 6: Cpp file for Event class.

```
1   /*
2    * Author: Chris Savill, chs17@aber.ac.uk
3    * File Name: event.cpp
4    * Description: cpp file that contains member function definitions for the
       event class.
5    * First Created: 11/03/2013
6    * Last Modified: 14/03/2013
7    */
8
9   #include "event.h"
10  #include "creator.h"
11  #include <iostream>
12  #include <stdlib.h>
13  #include <fstream>
14  #include <sstream>
15  #include <limits>
16
17  using namespace std;
```

```
18
19  /* Member function that returns a pointer to the vector of courses. */
20  vector<Course*>* Event::getCourses() {
21      return courses;
22  }
23
24  /* Member function to get the user to input the events name. */
25  void Event::set_name() {
26      bool name_chosen = false;
27      string name;
28
29      do {
30          do {
31              cout << "Please enter in the name for the event (no more than 79
                    characters): ";
32              cin.clear();
33              getline(cin, name);
34          } while (name.length() > MAX_EVENT_NAME_LENGTH);
35
36          cout << endl << endl << "Are you happy with the name: '" << name <<
                "'?" << endl;
37          name_chosen = get_acceptance();
38      } while (name_chosen == false);
39
40      this->name = name;
41  }
42
43  /* Member function to get the user to input the date of the event. */
44  void Event::set_date() {
45      bool date_chosen = false;
46      string date;
47
48      do {
49          do {
50              cout << endl << endl << "Please enter in the date for the event
                    (no more than 19 characters): ";
51              cin.clear();
52              getline(cin, date);
53          } while (date.length() > MAX_DATE_LENGTH);
54
55          cout << endl << endl << "Are you happy with the date: '" << date <<
                "'?" << endl;
56          date_chosen = get_acceptance();
57      } while (date_chosen == false);
58
59      this->date = date;
60  }
61
62  /* Member function to get the user to input the start time of the event. */
63  void Event::set_start_time() {
64      bool start_time_chosen = false;
65      bool valid_hours = false;
66      bool valid_minutes = false;
67      char input[3];
68      int hours;
69      int minutes;
70      string start_time;
71      string string_hours;
72      string string_minutes;
73
74      do {
75          do {
76              cout << endl << endl << "Please enter in the start time for the
```

```
                event with the 24 hour format 'HH:MM', hours first: ";
77              cin.clear();
78              cin >> input;
79              cin.ignore(numeric_limits<streamsize>::max(), '\n');
80              cout << endl;
81
82              if (isdigit(input[0]) && isdigit(input[1])) { //Ensures the
                    input has 2 digits.
83                  hours = atoi(input); //Converts the digits into an int and
                        stores it in hours.
84
85                  if (hours <= 23 && hours >= 00) { //Makes sure that the
                        hours are in 24-hour format.
86                      cout << "Valid hours entered." << endl << endl;
87                      valid_hours = true;
88                  }
89              } else cout << "Invalid hours entered, please enter in a value
                    between 00 and 23 inclusive." << endl << endl;
90          } while (valid_hours == false);
91
92          do {
93              cout << endl << endl << "Please now enter in the minutes: ";
94              cin.clear();
95              cin >> input;
96              cin.ignore(numeric_limits<streamsize>::max(), '\n');
97              cout << endl;
98
99              if (isdigit(input[0]) && isdigit(input[1])) {
100                 minutes = atoi(input);
101
102                 if (minutes <= 59 && minutes >= 00) { //Makes sure minutes
                        are valid.
103                     cout << "Valid minutes entered." << endl << endl;
104                     valid_minutes = true;
105                 }
106             } else cout << "Invalid minutes entered, please enter in a value
                    between 00 and 59 inclusive." << endl << endl;
107         } while (valid_minutes == false);
108
109         cout << endl << endl << "Are you happy with the start time: '" <<
                hours << ":" << minutes << "'?" << endl;
110         start_time_chosen = get_acceptance();
111     } while (start_time_chosen == false);
112
113     ostringstream string_retriever; //Converts ints into strings.
114     string_retriever << hours;
115     string_hours = string_retriever.str();
116     string_retriever.str(""); //Clears the string stream.
117     string_retriever << minutes;
118     string_minutes = string_retriever.str();
119
120     start_time = string_hours + ":" + string_minutes; //Concatenates the
            final time into HH:MM format.
121     this->start_time = start_time;
122 }
123
124 /* Member function that will handle adding a competitor to the event.
125  * @param number The current competitor number.
126  */
127 void Event::add_competitor() {
128     if (courses->empty()) cout << "No courses exist for competitor course
            selection. Please create a course first." << endl << endl;
129     else {
```

```
130         Competitor *competitor = new Competitor((competitors->size() + 1),
                this);
131         competitors->push_back(competitor);
132         cout << "New competitor added to event." << endl << endl;
133         cout << "Competitor number: " << competitors->back()->get_number();
134         cout << "Competitor name: " << competitors->back()->get_name() <<
                endl;
135         cout << "Course: " << competitors->back()->get_course() << endl;
136     }
137 }
138
139 /* Member function that will handle adding a course to the event. */
140 void Event::add_course() {
141     Course *course = new Course(this);
142     courses->push_back(course);
143     cout << "New course added to event." << endl << endl;
144     cout << "Course letter: " << courses->back()->get_letter() << endl;
145     cout << "Number of course nodes: " << courses->back()->
            get_number_of_nodes() << endl;
146     cout << "Nodes: " << courses->back()->get_node(0);
147
148     for (int counter = 1; counter < courses->back()->get_number_of_nodes();
            counter++) {
149         cout << ", " << courses->back()->get_node(counter);
150     }
151
152     cout << endl << endl;
153 }
154
155 /* Member function that will handle exporting the name, date and start_time
        of the event to a '.txt' file. */
156 void Event::export_event() {
157     ofstream competitors_file;
158     competitors_file.open("name.txt", ios::out);
159
160     if (competitors_file.is_open()) {
161         competitors_file << this->name << "\n" << this->date << "\n" << this
                ->start_time;
162         competitors_file.close();
163         cout << "Event successfully exported to 'name.txt'." << endl << endl
                ;
164     } else cout << "File 'name.txt' could not be written." << endl;
165 }
166
167 /* Member function that will handle the exporting of the array of
        competitors to a '.txt' file. */
168 void Event::export_competitors() {
169     if (competitors->empty()) cout << "No competitors to export. Exporting
            cancelled." << endl << endl;
170     else {
171         ofstream competitors_file;
172         competitors_file.open("entrants.txt", ios::out);
173
174         if (competitors_file.is_open()) {
175             for (int counter = 0; counter < this->competitors->size();
                    counter++) {
176                 competitors_file << this->competitors->at(counter)->
                        get_number() << " " << this->competitors->at(counter)->
                        get_course()
177                         << " " << this->competitors->at(counter)->get_name()
                                << "\n";
178             }
179
```

```
180             competitors_file.close();
181             cout << "Competitors successfully exported to 'entrants.txt'."
                    << endl << endl;
182         } else cout << "File 'entrants.txt' could not be written." << endl;
183     }
184 }
185
186 /* Member function that will handle the exporting of the array of courses to
        a '.txt' file. */
187 void Event::export_courses() {
188     if (courses->empty()) cout << "No courses to export. Exporting cancelled
            ." << endl << endl;
189     else {
190         ofstream courses_file;
191         courses_file.open("courses.txt", ios::out);
192
193         if (courses_file.is_open()) {
194             for (int counter = 0; counter < this->courses->size(); counter
                    ++) {
195                 courses_file << this->courses->at(counter)->get_letter() <<
                        " " << this->courses->at(counter)->get_number_of_nodes();
196
197                 for (int counter2 = 0; counter2 < this->courses->at(counter)
                        ->get_number_of_nodes(); counter2++) {
198                     courses_file << " " << this->courses->at(counter)->
                            get_node(counter2);
199                 }
200                 courses_file << "\n";
201             }
202
203             courses_file.close();
204             cout << "Courses successfully exported to 'courses.txt'." <<
                    endl << endl;
205         } else cout << "File 'courses.txt' could not be written." << endl;
206     }
207 }
208
209 /* Constructor for Event class. */
210 Event::Event() {
211     competitors = new vector<Competitor* > ();
212     courses = new vector<Course* > ();
213     set_name();
214     set_date();
215     set_start_time();
216
217     cout << "Event name: " << this->name << endl;
218     cout << "Event date: " << this->date << endl;
219     cout << "Event start time: " << this->start_time << endl << endl;
220 }
221
222 /* Destructor for Event class. */
223 Event::~Event() {
224     delete(competitors);
225     delete(courses);
226 }
```

Listing 7: Cpp file for Course class.

```
1 /*
2  * Author: Chris Savill, chs17@aber.ac.uk
3  * File Name: course.cpp
4  * Description: cpp file that contains member function definitions for the
```

```
    course class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#include "course.h"
#include "creator.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <limits>

using namespace std;

/* Member function to return a course's letter. */
char Course::get_letter() {
    return this->letter;
}

/* Member function to return a course's number of nodes. */
int Course::get_number_of_nodes() {
    return this->number_of_nodes;
}

/* Member function to return a node from the course's vector of nodes. */
int Course::get_node(int index) {
    return this->nodes->at(index);
}

/* Member function that checks if the letter given be the user matches any
   of the course letters. */
bool checkCourseExists(char letter, Event *event) {
    for (int counter = 0; counter < event->getCourses()->size(); counter++)
        {
        if (letter == event->getCourses()->at(counter)->get_letter()) return
            true; //Checks if letter matches any of the course letters.
    }

    return false; //Return false if no match found.
}

/* Member function that will set the letter of the course. */
void Course::set_letter(Event *event) {
    bool valid_letter = false;
    bool letter_chosen = false;
    char letter;

    do {
        do {
            cout << endl << endl << "Please enter in the course letter for
                the course: ";
            cin.clear();
            letter = cin.get();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            if (isalpha(letter) && !checkCourseExists(letter, event))
                valid_letter = true; //Checks that character entered is a
                letter and that it does not match any course letters.
            else {
                cout << "Please enter in a valid course letter that does not
                    already exist in this event, a-z or A-Z." << endl <<
                    endl;
                valid_letter = false;
```

```
59              }
60          } while (valid_letter == false);
61
62          cout << endl << "Are you happy with the course letter: '" << letter
                << "'?" << endl;
63          letter_chosen = get_acceptance();
64      } while (letter_chosen == false);
65
66      this->letter = letter;
67  }
68
69  /* Member function that will set the number of nodes of the course. */
70  void Course::set_number_of_nodes() {
71      bool number_chosen = false;
72      int number;
73
74      do {
75          cout << endl << endl << "Please enter in the number of nodes for
                this course: ";
76          cin.clear();
77          cin >> number;
78          cin.ignore(numeric_limits<streamsize>::max(), '\n');
79
80          cout << endl << endl << "Are you happy with the number of nodes: '"
                << number << "'?" << endl;
81          number_chosen = get_acceptance();
82      } while (number_chosen == false && number > 0);
83
84      this->number_of_nodes = number;
85  }
86
87  /* Member function that reads in the nodes from the 'nodes.txt' file and
        adds them to the nodes available array. */
88  bool Course::read_nodes_available() {
89      ifstream nodes_file;
90      string input;
91      int node_number;
92
93      nodes_file.open("nodes.txt", ios::in);
94
95      if (nodes_file.is_open()) {
96          while (getline(nodes_file, input)) { //Keep reading until EOF
                reached.
97              stringstream int_retriever(input); //Retrieves int from the
                    string stream.
98              int_retriever >> node_number; //Stores the int in node_number.
99              this->nodes_available->push_back(node_number);
100         }
101
102         nodes_file.close();
103         cout << "Nodes from 'nodes.txt' read in successfully." << endl;
104         cout << "Nodes read in: " << nodes_available->at(0);
105         for (int counter = 1; counter < nodes_available->size(); counter++)
                cout << ", " << nodes_available->at(counter);
106         cout << endl << endl;
107     } else cout << "File 'nodes.txt' could not be opened. Please check file
            is in correct directory and permissions." << endl;
108 }
109
110 /* Member function that adds a new node to the course. */
111 void Course::add_node() {
112     bool number_chosen = false;
113     string input;
```

```
114     int number = 0;
115
116     do {
117         do {
118             cout << "Please enter in the node number you wish to add to the
                    course: ";
119             getline(cin, input);
120             stringstream int_retriever(input);
121             int_retriever >> number;
122         } while (duplicated_last_node(number) || !check_node_exists(number))
                ; //Makes sure that the number entered doesn't match the last
                number entered and that it does exist.
123
124         cout << endl << endl << "Are you happy with the node number: '" <<
                number << "'?" << endl;
125         number_chosen = get_acceptance();
126     } while (number_chosen == false);
127
128     this->nodes->push_back(number);
129 }
130
131 /* Member function to check if the new node being selected matches the last
       node added. */
132 bool Course::duplicated_last_node(int number) {
133     if (!nodes->empty()) { //Only checks if there are nodes present.
134         if (number == nodes->back()) {
135             cout << "Node matches last node. Please choose a different node
                    number to add." << endl;
136             return true;
137         }
138     }
139
140     return false; //Returns false if the number entered and the last number
            entered don't match.
141 }
142
143 /* Member function that checks that the node being added exists in the array
        of node available. */
144 bool Course::check_node_exists(int number) {
145     for (int counter = 0; counter < this->nodes_available->size(); counter
            ++) {
146         if (number == this->nodes_available->at(counter)) return true;
147     }
148
149     cout << "Node does not exist, please choose a different node number to
            add." << endl;
150     return false; //Returns false if the number entered does not exist in
            the vector of nodes available.
151 }
152
153 /* Constructor for Course class. */
154 Course::Course(Event *event) {
155     this->nodes = new vector<int>();
156     this->nodes_available = new vector<int>();
157
158     if (read_nodes_available()) {
159         set_letter(event);
160         set_number_of_nodes();
161
162         for (int counter = 0; counter < number_of_nodes - 1; counter++) {
163             add_node();
164         }
165
```

```
166         nodes->push_back(nodes->front()); //Adds the last node, matching the
                first node to the course.
167     } else cout << "Nodes could not be read in from 'nodes.txt' file. Course
            creation cancelled." << endl << endl;
168 }
169
170 /* Destructor for Course class. */
171 Course::~Course() {
172     delete(nodes);
173     delete(nodes_available);
174 }
```

Listing 8: Cpp file for Competitor class.

```
1  /*
2   * Author: Chris Savill, chs17@aber.ac.uk
3   * File Name: competitor.cpp
4   * Description: cpp file that contains member function definitions for the
        competitor class.
5   * First Created: 11/03/2013
6   * Last Modified: 14/03/2013
7   */
8
9  #include "competitor.h"
10 #include "creator.h"
11 #include <ctype.h>
12 #include <iostream>
13 #include <limits>
14
15 using namespace std;
16
17 /* Member function to return a competitor's number. */
18 int Competitor::get_number() {
19     return this->number;
20 }
21
22 /* Member function to return a competitor's name. */
23 string Competitor::get_name() {
24     return this->name;
25 }
26
27 /* Member function to return a competitor's course. */
28 char Competitor::get_course() {
29     return this->course;
30 }
31
32 /* Member function that will set the number of the competitor.
33  * @param number The number for the competitor.
34  */
35 void Competitor::set_number(int number) {
36     this->number = number;
37 }
38
39 /* Member function that will set the name of the competitor. */
40 void Competitor::set_name() {
41     bool name_chosen = false;
42     string name;
43
44     do {
45         do {
46             cout << endl << endl << "Please enter in the name for the
                    competitor (no more than 50 characters): ";
```

```
47              getline(cin, name);
48          } while (name.length() > MAX_COMPETITOR_NAME_LENGTH);
49
50          cout << endl << endl << "Are you happy with the name: '" << name <<
                "'?" << endl;
51
52          name_chosen = get_acceptance();
53
54      } while (name_chosen == false);
55
56      this->name = name;
57  }
58
59  /* Member function that will set the course letter for the competitor. */
60  void Competitor::set_course(Event *event) {
61      bool valid_letter = false;
62      bool letter_chosen = false;
63      char letter;
64
65      do {
66          do {
67              cout << endl << endl << "List of courses available for the
                    competitor to enter on: " << event->getCourses()->front()->
                    get_letter();
68
69              if (event->getCourses()->size() > 1) { //Only prints out other
                    courses if the size of the vector > 1.
70                  for (int counter = 1; counter < event->getCourses()->size();
                        counter++)
71                      cout << ", " << event->getCourses()->at(counter)->
                            get_letter();
72              }
73
74              cout << endl << endl << "Please enter in the letter of the
                    course that the competitor is entering: ";
75              cin.clear(); //Resets the input stream flags.
76              letter = cin.get(); //Gets a single character.
77              cin.ignore(numeric_limits<streamsize>::max(), '\n'); //Clears
                    the input stream.
78
79              if (isalpha(letter) && checkCourseExists(letter, event))
                    valid_letter = true; //Makes sure character is a letter and
                    that it corresponds to a course that exists.
80              else {
81                  cout << "Please enter in a valid course letter." << endl <<
                        endl;
82                  valid_letter = false;
83              }
84          } while (valid_letter == false);
85
86          cout << endl << "Are you happy with the course letter: '" << letter
                << "'?" << endl;
87          letter_chosen = get_acceptance();
88      } while (letter_chosen == false);
89
90      this->course = letter;
91  }
92
93  /* Constructor for Competitor class.
94   * @param number The number for the new competitor.
95   */
96  Competitor::Competitor(int number, Event *event) {
97      set_number(number);
```

```
98      cout << "Competitor number: " << this->number << endl;
99      set_name();
100     cout << "Competitor name: " << this->name << endl;
101     set_course(event);
102     cout << "Competitor course:" << this-> course << endl;
103 }
```

3 Clean build and compilation of Event Creation Program

```
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .clean-
    conf
make[1]: Entering directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Event_Creation_Program'
rm -f -r build/Debug
rm -f dist/Debug/GNU-Linux-x86/event_creation_program
make[1]: Leaving directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Event_Creation_Program'


CLEAN SUCCESSFUL (total time: 217ms)

"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-
    conf
make[1]: Entering directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Event_Creation_Program'
"/usr/bin/make"  -f nbproject/Makefile-Debug.mk dist/Debug/GNU-Linux-x86/
    event_creation_program
make[2]: Entering directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Event_Creation_Program'
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/main.o.d
g++    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/main.o.d -o build/Debug/
    GNU-Linux-x86/main.o main.cpp
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/course.o.d
g++    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/course.o.d -o build/
    Debug/GNU-Linux-x86/course.o course.cpp
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/event.o.d
g++    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/event.o.d -o build/Debug
    /GNU-Linux-x86/event.o event.cpp
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/competitor.o.d
g++    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/competitor.o.d -o build/
    Debug/GNU-Linux-x86/competitor.o competitor.cpp
mkdir -p dist/Debug/GNU-Linux-x86
g++     -o dist/Debug/GNU-Linux-x86/event_creation_program build/Debug/GNU-
    Linux-x86/main.o build/Debug/GNU-Linux-x86/course.o build/Debug/GNU-Linux
    -x86/event.o build/Debug/GNU-Linux-x86/competitor.o
make[2]: Leaving directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Event_Creation_Program'
make[1]: Leaving directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Event_Creation_Program'


BUILD SUCCESSFUL (total time: 5s)
```

4 Run through of Event Creation Program

```
Please enter in the name for the event (no more than 79 characters): Horse
   Trekkers 21st Anniversary


Are you happy with the name: 'Horse Trekkers 21st Anniversary'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y


Please enter in the date for the event (no more than 19 characters): 3rd May
    2013


Are you happy with the date: '3rd May 2013'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y


Please enter in the start time for the event with the 24 hour format 'HH:MM
   ', hours first: 13

Valid hours entered.



Please now enter in the minutes: 37

Valid minutes entered.



Are you happy with the start time: '13:37'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Event name: Horse Trekkers 21st Anniversary
Event date: 3rd May 2013
Event start time: 13:37

*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
*              2. Add Course to Event                   *
*              3. Export Event to File                  *
*              4. Export Competitors to File            *
*              5. Export Courses to File                *
*              6. Exit Event Creation Program           *
*********************************************************

Please enter in an option from the above an press 'Enter': 1
No courses exist for competitor course selection. Please create a course
   first.

*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
```

```
*              2. Add Course to Event                    *
*              3. Export Event to File                   *
*              4. Export Competitors to File             *
*              5. Export Courses to File                 *
*              6. Exit Event Creation Program            *
*********************************************************

Please enter in an option from the above an press 'Enter': 2
Nodes from 'nodes.txt' read in successfully.
Nodes read in: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18



Please enter in the course letter for the course: A

Are you happy with the course letter: 'A'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y


Please enter in the number of nodes for this course: 5


Are you happy with the number of nodes: '5'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Please enter in the node number you wish to add to the course: 1


Are you happy with the node number: '1'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Please enter in the node number you wish to add to the course: 3


Are you happy with the node number: '3'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Please enter in the node number you wish to add to the course: 12


Are you happy with the node number: '12'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Please enter in the node number you wish to add to the course: 20
Node does not exist, please choose a different node number to add.
Please enter in the node number you wish to add to the course: 8


Are you happy with the node number: '8'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
New course added to event.

Course letter: A
Number of course nodes: 5
Nodes: 1, 3, 12, 8, 1
```

```
*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
*              2. Add Course to Event                   *
*              3. Export Event to File                  *
*              4. Export Competitors to File            *
*              5. Export Courses to File                *
*              6. Exit Event Creation Program           *
*********************************************************

Please enter in an option from the above an press 'Enter': 1
Competitor number: 1


Please enter in the name for the competitor (no more than 50 characters):
   Julius Munching


Are you happy with the name: 'Julius Munching'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Competitor name: Julius Munching


List of courses available for the competitor to enter on: A

Please enter in the letter of the course that the competitor is entering: a
Please enter in a valid course letter.



List of courses available for the competitor to enter on: A

Please enter in the letter of the course that the competitor is entering: A

Are you happy with the course letter: 'A'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Competitor course:A
New competitor added to event.

Competitor number: 1Competitor name: Julius Munching
Course: A
*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
*              2. Add Course to Event                   *
*              3. Export Event to File                  *
*              4. Export Competitors to File            *
*              5. Export Courses to File                *
*              6. Exit Event Creation Program           *
*********************************************************

Please enter in an option from the above an press 'Enter': 1
Competitor number: 2


Please enter in the name for the competitor (no more than 50 characters):
```

```
   Helen Boon


Are you happy with the name: 'Helen Boon'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Competitor name: Helen Boon


List of courses available for the competitor to enter on: A

Please enter in the letter of the course that the competitor is entering: A

Are you happy with the course letter: 'A'?
If yes press 'y' then 'Enter'
If no press 'n' then 'Enter'
y
Competitor course:A
New competitor added to event.

Competitor number: 2Competitor name: Helen Boon
Course: A
*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
*              2. Add Course to Event                   *
*              3. Export Event to File                  *
*              4. Export Competitors to File            *
*              5. Export Courses to File                *
*              6. Exit Event Creation Program           *
*********************************************************

Please enter in an option from the above an press 'Enter': 3
Event successfully exported to 'name.txt'.

*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
*              2. Add Course to Event                   *
*              3. Export Event to File                  *
*              4. Export Competitors to File            *
*              5. Export Courses to File                *
*              6. Exit Event Creation Program           *
*********************************************************

Please enter in an option from the above an press 'Enter': 4
Competitors successfully exported to 'entrants.txt'.

*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*              1. Add Competitor to Event               *
*              2. Add Course to Event                   *
*              3. Export Event to File                  *
*              4. Export Competitors to File            *
*              5. Export Courses to File                *
*              6. Exit Event Creation Program           *
*********************************************************

Please enter in an option from the above an press 'Enter': 5
```

```
Courses successfully exported to 'courses.txt'.

*********************************************************
*  Runners and Riders Event Creation Program Main Menu  *
*********************************************************
*               1. Add Competitor to Event              *
*               2. Add Course to Event                  *
*               3. Export Event to File                 *
*               4. Export Competitors to File           *
*               5. Export Courses to File               *
*               6. Exit Event Creation Program          *
*********************************************************

Please enter in an option from the above an press 'Enter': 6
Exiting program...


RUN SUCCESSFUL (total time: 2m 46s)
```

5 Files created by execution of Event Creation Program

Listing 9: Event 'name.txt' file

```
Horse Trekkers 21st Anniversary
3rd May 2013
13:37
```

Listing 10: Event 'courses.txt' file

```
A 5 1 3 12 8 1
```

Listing 11: Event 'entrants.txt' file

```
1 A Julius Munching
2 A Helen Boon
```

6 Code for Checkpoint Manager Program

6.1 Data_Structures package

Listing 12: Launcher class.

```
1  /* File Name: Launcher.java
2   * Description: Launcher class which handles the initial launching of the
        Checkpoint Manager Program.
3   * First Created: 15/03/2013
4   * Last Modified: 19/03/2013
5   */
6  package Data_Structures;
7
8  import GUI.TypeWindow;
9  import java.io.IOException;
10 import javax.swing.JOptionPane;
11
12 /**
13  * @author Chris Savill, chs17@aber.ac.uk
14  */
15 public class Launcher {
16
17     /**
18      * Main method that checks that the right number of arguments were
            received
19      * and calls methods to load the file required and launch the GUI.
20      *
21      * @param args String array of arguments, should be a list of file names
            .
22      */
23     public static void main(String[] args) throws IOException {
24         if (args.length < 4) {
25             JOptionPane.showMessageDialog(null, "Invalid number of file
                   names supplied required for program to run.\n\n"
26                     + "File names required for:\nFile containing nodes\nFile
                        containing courses\nFile containing entrants\n"
27                     + "File to retrieve time records and write time records
                        to.\n\n"
28                     + "Now exiting program.");
29         } else {
30             Event event = new Event(args);
31
32             if (event.loadCycle(args)) {
33                 JOptionPane.showMessageDialog(null, "Data files loaded
                       successfully.");
34                 TypeWindow typeWindow = new TypeWindow(event);
35             } else {
36                 System.out.print("Exiting Program...\n");
37             }
38         }
39     }
40 }
```

Listing 13: Event class.

```
1  /* File Name: Manager.java
2   * Description: Event class which stores all members and functions
       pertaining to an event.
3   * First Created: 15/03/2013
4   * Last Modified: 18/03/2013
```

```
 */
package Data_Structures;

import File_Handling.FileHandler;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;

/**
 * @author Chris Savill, chs17@aber.ac.uk
 */
public class Event {

    private ArrayList<Competitor> competitors; //Array list of competitors
        in an event.
    private ArrayList<Node> nodes; //Array list of nodes in an event.
    private ArrayList<Node> checkpoints; //Array list of nodes that are of
        type "CP" or "MC".
    private ArrayList<Course> courses; //Array list of courses in an event.
    private ArrayList<Record> records; //Array list of records logged.
    private int lastLineRead;
    private Date lastRecordedTime;
    private boolean timeFileExists;
    private String[] fileNames;

    /**
     * Method to return array list of competitors.
     *
     * @return The array list of competitors.
     */
    public ArrayList<Competitor> getCompetitors() {
        return competitors;
    }

    /**
     * Method to return array list of nodes.
     *
     * @return The array list of nodes.
     */
    public ArrayList<Node> getNodes() {
        return nodes;
    }

    /**
     * Method to return array list of checkpoints.
     *
     * @return The array list of checkpoints (non-junction nodes).
     */
    public ArrayList<Node> getCheckpoints() {
        return checkpoints;
    }

    /**
     * Method to return array list of courses.
     *
     * @return The array list of courses.
     */
    public ArrayList<Course> getCourses() {
        return courses;
    }

    /**
     * Method to return array list of records.
```

```
66          *
67          * @return The array list of records.
68          */
69         public ArrayList<Record> getRecords() {
70             return records;
71         }
72
73         /**
74          * Method to get the last line read number.
75          *
76          * @return The line read from the times file.
77          */
78         public int getLastLineRead() {
79             return lastLineRead;
80         }
81
82         /**
83          * Method to return the array of file names.
84          *
85          * @return The string array of file names.
86          */
87         public String[] getFileNames() {
88             return fileNames;
89         }
90
91         /**
92          * Method to set the last line read number.
93          *
94          * @param lineNumber The line read from the times file.
95          */
96         public void setLastLineRead(int lineNumber) {
97             this.lastLineRead = lineNumber;
98         }
99
100        /**
101         * Method to set the last time recorded.
102         *
103         * @param time The last time recorded.
104         */
105        public void setLastRecordedTime(Date time) {
106            this.lastRecordedTime = time;
107        }
108
109        /**
110         * Method to call a series of methods to load in the data required by
                the
111         * program.
112         *
113         * @param args The list of filenames to load the required data into the
114         * system.
115         * @return Successful/Unsuccessful.
116         */
117        public boolean loadCycle(String[] fileNames) throws IOException {
118            this.fileNames = fileNames;
119
120            FileHandler fileReader = new FileHandler();
121
122            if (fileReader.readNodes(fileNames[0], this)) {
123                if (fileReader.readCourses(fileNames[1], this)) {
124                    if (fileReader.readCompetitors(fileNames[2], this)) {
125                        return true;
126                    } else {
127                        System.out.print("Failed to load competitors. Program
```

```
                        Exiting.\n");
128                 }
129             } else {
130                 System.out.print("Failed to load courses. Program Exiting.\n
                        ");
131             }
132         } else {
133             System.out.print("Failed to load nodes. Program Exiting.\n");
134         }
135 
136         return false;
137     }
138 
139     /**
140      * Method that checks if the node number passed in exists in the array
             list
141      * of nodes loaded in.
142      *
143      * @param number The number to be compared with.
144      * @return True if node exists else false.
145      */
146     public boolean checkNodeExists(int number) {
147         for (int counter = 0; counter < nodes.size(); counter++) {
148             if (number == nodes.get(counter).getNumber()) {
149                 return true;
150             } //Nodes exists.
151         }
152 
153         return false; //Returns false if the node number passed in does not
                exist in the array list of nodes.
154     }
155 
156     /**
157      * Method that checks if the course letter passed in exists in the array
158      * list of courses loaded in.
159      *
160      * @param letter The letter to be compared with.
161      * @return True if course exists else false.
162      */
163     public boolean checkCourseExists(char letter) {
164         for (int counter = 0; counter < courses.size(); counter++) {
165             if (letter == courses.get(counter).getLetter()) {
166                 return true;
167             } //Course exists.
168         }
169 
170         return false; //Returns false if the course letter passed in does
                not exist in the array list of courses.
171     }
172 
173     /**
174      * Method to let the know event instance know that a time file does now
175      * exist.
176      *
177      */
178     public void setTimesFilesExistsTrue() {
179         timeFileExists = true;
180     }
181 
182     /**
183      * Method to find a competitor and return it.
184      *
185      * @param competitorNumber The number of the competitor being looked for
```

```
186         .
187          * @return The competitor matched.
188          */
189         public Competitor retrieveCompetitor(int competitorNumber) {
190             for (Competitor competitor : competitors) {
191                 if (competitor.getNumber() == competitorNumber) {
192                     return competitor;
193                 }
194             }
195             return null;
196         }
197 
198         /**
199          * Method to find a course and return it.
200          *
201          * @param courseLetter The course being looked for.
202          * @return The course matched.
203          */
204         public Course retrieveCourse(char courseLetter) {
205             for (Course course : courses) {
206                 if (course.getLetter() == courseLetter) {
207                     return course;
208                 }
209             }
210             return null;
211         }
212 
213         /**
214          * Method to retrieve the checkpoint number.
215          *
216          * @param type The type of the checkpoint.
217          * @param listIndex The index of the list element.
218          * @param numberOfElements The size of the list.
219          * @return The checkpoint number being looked for.
220          */
221         public int retrieveCheckpointNumber(String type, int listIndex, int
                numberOfElements) {
222             int[] checkpointArray = new int[numberOfElements];
223             int arrayIndex = 0;
224 
225             for (int counter = 0; counter < checkpoints.size(); counter++) {
226                 if (checkpoints.get(counter).getType().equals(type)) {
227                     checkpointArray[arrayIndex++] = checkpoints.get(counter).
                            getNumber();
228                 }
229             }
230 
231             return checkpointArray[listIndex];
232         }
233 
234         /**
235          * Method to check if the new record is valid.
236          *
237          * @param checkpoint The checkpoint number.
238          * @param status The status.
239          * @param competitorNumber The competitor's number.
240          * @param time The time of the record.
241          * @return True is record is valid, else false.
242          */
243         public boolean checkNewRecord(int checkpoint, int status, int
                competitorNumber, Date time) {
244             Competitor competitor = retrieveCompetitor(competitorNumber);
```

```
245             if (timeFileExists != false) {
246                 if (time.before(lastRecordedTime)) {
247                     System.out.println("\nInvalid time.");
248                     return false;
249                 } else if (competitor.getCheckpointIndex() == competitor.
                        getCheckpoints().length) {
250                     System.out.println("\nCompetitor already finished.");
251                     return false;
252                 }
253             }
254 
255 
256             if (competitor.getStatus() == 'I' || competitor.getStatus() == 'E')
                    {
257                 System.out.println("\nCompetitor already excluded.");
258                 return false; //Should not be updated as competitor already
                        excluded.
259             } else if (status == 2 || status == 3) {
260                 if (competitor.getStatus() != 'A') {
261                     System.out.println("\nCompetitor hasn't arrived at a medical
                            checkpoint yet.");
262                     return false; //Competitor cannot be departing or be exclude
                            from a medical checkpoint they haven't arrived at.
263                 } else {
264                     return true;
265                 }
266             } else if (status == 0) {
267                 if (competitor.getStatus() != 'A') {
268                     return true;
269                 } else {
270                     System.out.println("\nCompetitor is still being examined at
                            a medical checkpoint.");
271                     return false; //Competitor cannot be at a time checkpoint
                            when should be at a medical checkpoint being examined.
272                 }
273             } else if (status == 1) {
274                 return true;
275             }
276 
277             return false;
278         }
279 
280         /**
281          * Method to determine the final status to be written to the time record
                file.
282          * @param checkpoint The checkpoint number.
283          * @param status The status.
284          * @param competitorNumber The competitor's number.
285          * @return The final status for the record.
286          */
287         public char determineFinalStatus(int checkpoint, int status, int
                competitorNumber) {
288             Competitor competitor = retrieveCompetitor(competitorNumber);
289 
290             if (competitor.getStatus() == 'N') {
291                 if (checkpoint != competitor.getCheckpoints()[competitor.
                        getCheckpointIndex()]) {
292                     return 'I';
293                 } else if (status == 0) {
294                     return 'T';
295                 } else if (status == 1) {
296                     return 'A';
297                 }
```

```
298         } else if (competitor.getStatus() == 'A') {
299             if (status == 2) {
300                 return 'D';
301             } else if (status == 3) {
302                 return 'E';
303             }
304         } else if (checkpoint != competitor.getCheckpoints()[competitor.
                getCheckpointIndex() + 1]) {
305             return 'I';
306         } else {
307             if (status == 0) {
308                 return 'T';
309             } else if (status == 1) {
310                 return 'A';
311             } else if (status == 2) {
312                 return 'D';
313             } else if (status == 3) {
314                 return 'E';
315             }
316         }
317 
318         System.out.print("\n\nInvalid final status, returning 'I'.\n");
319         return 'I';
320     }
321 
322     /**
323      * Constructor to initialise the event.
324      */
325     public Event(String[] fileNames) {
326         competitors = new ArrayList<Competitor>();
327         nodes = new ArrayList<Node>();
328         checkpoints = new ArrayList<Node>();
329         courses = new ArrayList<Course>();
330         records = new ArrayList<Record>();
331         lastLineRead = 0;
332         timeFileExists = false;
333     }
334 }
```

Listing 14: Node class.

```
1  /* File Name: Node.java
2   * Description: Node class which stores all members and functions pertaining
        to a node.
3   * First Created: 15/03/2013
4   * Last Modified: 15/03/2013
5   */
6  package Data_Structures;
7 
8  /**
9   * @author Chris Savill, chs17@aber.ac.uk
10  */
11 public class Node {
12 
13     private int number;
14     private String type;
15 
16     /**
17      * Constructor to initialise Node.
18      *
19      * @param number The number of the node.
20      * @param type The type of the node.
```

```
21          */
22         public Node(int number, String type) {
23             this.number = number;
24             this.type = type;
25         }
26 
27         /**
28          * Method to return the node's number.
29          *
30          * @return The node number.
31          */
32         public int getNumber() {
33             return number;
34         }
35 
36         /**
37          * Method to return the node's type.
38          * @return The type of the node.
39          */
40         public String getType() {
41             return type;
42         }
43 }
```

Listing 15: Course class.

```
1  /* File Name: Couse.java
2   * Description: Course class which stores all members and functions
        pertaining to a course.
3   * First Created: 15/03/2013
4   * Last Modified: 17/03/2013
5   */
6  package Data_Structures;
7 
8  /**
9   * @author Chris Savill, chs17@aber.ac.uk
10  */
11 public class Course {
12 
13     private char letter;
14     private int numberOfNodes;
15     private int[] nodes;
16 
17     /**
18      * Constructor to initialise course.
19      *
20      * @param letter The course letter identifier.
21      * @param numberOfNodes The number of nodes the course contains.
22      * @param nodes The array of nodes the course contains.
23      */
24     public Course(char letter, int numberOfNodes, int[] nodes) {
25         this.letter = letter;
26         this.numberOfNodes = numberOfNodes;
27         this.nodes = nodes;
28     }
29 
30     /**
31      * Method to return the course letter.
32      */
33     public char getLetter() {
34         return letter;
35     }
```

```
36
37      /**
38       * Method to return the number of nodes the course contains.
39       */
40      public int getNumberOfNodes() {
41          return numberOfNodes;
42      }
43
44      /**
45       * Method to return the array of nodes the course contains.
46       */
47      public int[] getNodes() {
48          return nodes;
49      }
50  }
```

Listing 16: Competitor class.

```
1   /* File Name: Competitor.java
2    * Description: Competitor class which stores all members and functions
         pertaining to a competitor.
3    * First Created: 15/03/2013
4    * Last Modified: 18/03/2013
5    */
6   package Data_Structures;
7
8   import java.util.ArrayList;
9
10  /**
11   * @author Chris Savill, chs17@aber.ac.uk
12   */
13  public class Competitor {
14
15      private String name;
16      private int number;
17      private char course;
18      private char status;
19      private int[] checkpoints;
20      private int checkpointIndex;
21
22      /**
23       * Constructor to initialise competitor.
24       *
25       * @param number The competitor's number.
26       * @param course The competitor's course.
27       * @param name The competitor's name.
28       */
29      public Competitor(int number, char course, String name, Event event) {
30          this.number = number;
31          this.course = course;
32          this.name = name;
33          this.checkpoints = setCheckpoints(event);
34          this.checkpointIndex = 0;
35          this.status = 'N'; //Not started yet.
36      }
37
38      /**
39       * Method to return the competitor's number.
40       *
41       * @return The number of the competitor.
42       */
43      public int getNumber() {
```

```
44          return number;
45      }
46
47      /**
48       * Method to return the course the competitor is entered on.
49       *
50       * @return The course the competitor entered in on.
51       */
52      public char getCourse() {
53          return course;
54      }
55
56      /**
57       * Method to return the competitor's name.
58       *
59       * @return The name of the competitor.
60       */
61      public String getName() {
62          return name;
63      }
64
65      /**
66       * Method to return the competitor's status.
67       *
68       * @return The status of the competitor.
69       */
70      public char getStatus() {
71          return status;
72      }
73
74      /**
75       * Method to return the index of the last checkpoint the competitor
             arrived
76       * at.
77       *
78       * @return The index of the last checkpoint the competitor arrived at.
79       */
80      public int getCheckpointIndex() {
81          return checkpointIndex;
82      }
83
84      /**
85       * Method to return the int array of checkpoints.
86       *
87       * @return The int array of checkpoints.
88       */
89      public int[] getCheckpoints() {
90          return checkpoints;
91      }
92
93      /**
94       * Method to get the nodes which are recordable checkpoints (non-
             junction
95       * nodes).
96       *
97       * @param event The event instance.
98       * @return The int array of checkpoints.
99       */
100     private int[] setCheckpoints(Event event) {
101         ArrayList<Integer> checkpointsList = new ArrayList<Integer>();
102         Course courseReference = event.retrieveCourse(course);
103
104         for (int counter = 0; counter < courseReference.getNumberOfNodes();
```

```
            counter++) {
105             for (int counter2 = 0; counter2 < event.getNodes().size();
                    counter2++) {
106                 if ((!event.getNodes().get(counter2).getType().equals("JN"))
107                         && (event.getNodes().get(counter2).getNumber() ==
                            courseReference.getNodes()[counter])) {
108                     checkpointsList.add(event.getNodes().get(counter2).
                        getNumber());
109                     break;
110                 }
111             }
112         }
113
114         int[] intList = new int[checkpointsList.size()];
115
116         for (int counter = 0; counter < checkpointsList.size(); counter++) {
117             intList[counter] = checkpointsList.get(counter).intValue();
118         }
119
120         return intList;
121     }
122
123     /**
124      * Method to set the status of the competitor.
125      *
126      * @param status The current status of the competitor.
127      */
128     public void setStatus(char status) {
129         this.status = status;
130     }
131
132     /**
133      * Method to increment the checkpoint index by 1.
134      */
135     public void incrementCheckpointIndex() {
136         checkpointIndex++;
137     }
138 }
```

Listing 17: Record class.

```
1  /* File Name: Record.java
2   * Description: Record class which stores all members and functions
        pertaining to checking a competitor in at a checkpoint.
3   * First Created: 15/03/2013
4   * Last Modified: 17/03/2013
5   */
6  package Data_Structures;
7
8  import java.util.Date;
9
10 /**
11  * @author Chris Savill, chs17@aber.ac.uk
12  */
13 public class Record {
14
15     private Event event;
16     private char competitorStatus;
17     private int checkpoint;
18     private int competitorNumber;
19     private Date time;
20
```

```
21      /**
22       * Constructor to initialise record data when read in from file.
23       *
24       * @param checkpoint The number of the checkpoint.
25       * @param competitorNumber The number of the competitor.
26       * @param time The time of the record.
27       */
28      public Record(char status, int checkpoint, int competitorNumber, Date
            time) {
29          this.competitorStatus = status;
30          this.checkpoint = checkpoint;
31          this.competitorNumber = competitorNumber;
32          this.time = time;
33      }
34
35      /**
36       * Constructor to initialise record data when recorded through GUI.
37       *
38       * @param checkpoint The number of the checkpoint.
39       * @param competitorNumber The number of the competitor.
40       * @param time The time of the record.
41       */
42      public Record(int checkpoint, char status, int competitorNumber, Date
            time) {
43          this.competitorStatus = status;
44          this.checkpoint = checkpoint;
45          this.competitorNumber = competitorNumber;
46          this.time = time;
47      }
48
49      /**
50       * Method to return the status of the competitor as marked by the
51       * checkpoint.
52       *
53       * @return The status of the competitor.
54       */
55      public char getCompetitorStatus() {
56          return competitorStatus;
57      }
58
59      /**
60       * Method to return the checkpoint number being recorded.
61       *
62       * @return The checkpoint number.
63       */
64      public int getCheckpointNumber() {
65          return checkpoint;
66      }
67
68      /**
69       * Method to return the competitor number being recorded.
70       *
71       * @return The competitor number.
72       */
73      public int getCompetitorNumber() {
74          return competitorNumber;
75      }
76
77      /**
78       * Method to return the time being recorded.
79       *
80       * @return The time of the record.
81       */
```

```
82      public Date getTime() {
83          return time;
84      }
85  }
```

6.2 File_Handling package

Listing 18: FileHandler class.

```
1   /* File Name: FileHandler.java
2    * Description: FileHandler class which stores methods to handle the reading
         of files.
3    * First Created: 15/03/2013
4    * Last Modified: 18/03/2013
5    */
6   package File_Handling;
7
8   import Data_Structures.Competitor;
9   import Data_Structures.Course;
10  import Data_Structures.Event;
11  import Data_Structures.Node;
12  import Data_Structures.Record;
13  import java.io.BufferedReader;
14  import java.io.FileNotFoundException;
15  import java.io.FileReader;
16  import java.io.FileWriter;
17  import java.io.IOException;
18  import java.io.RandomAccessFile;
19  import java.nio.channels.FileChannel;
20  import java.nio.channels.FileLock;
21  import java.text.ParseException;
22  import java.text.SimpleDateFormat;
23  import java.util.Date;
24  import java.util.logging.Level;
25  import java.util.logging.Logger;
26
27  /**
28   * @author Chris Savill, chs17@aber.ac.uk
29   */
30  public class FileHandler {
31
32      /**
33       * Method to read in all the details for the nodes pertaining to an
             event.
34       *
35       * @param fileName The file name required to access the file needed.
36       * @param event The event instance.
37       * @return True if file loaded successfully, else false if it fails at
             any
38       * point.
39       */
40      public boolean readNodes(String fileName, Event event) throws
            IOException {
41          String input;
42          int nodeNumber;
43          String nodeType;
44          String[] subStrings;
45          String pattern = "(\\d+\\s+([A-Z]{2})$)"; //Regular expression for
                nodes file.
46
47          try {
48              BufferedReader reader = new BufferedReader(new FileReader(
```

```
                fileName));
49
50              while ((input = reader.readLine()) != null) {
51                  if (input.matches(pattern)) { //Checks to make sure the line
                        is in the right format.
52                      subStrings = input.split("\\s+"); //Gets rid of
                            whitespace and separates the two sides into two
                            substrings.
53                      nodeNumber = Integer.parseInt(subStrings[0]); //
                            Retrieves the node number by parsing the string into
                            an int.
54                      nodeType = subStrings[1]; //Retrieves the node type.
55
56                      Node node = new Node(nodeNumber, nodeType); //Creates
                            new node with parameters read in.
57                      event.getNodes().add(node); //Adds new node to array
                            list of nodes.
58
59                      if (node.getType().equals("CP") || node.getType().equals
                            ("MC")) {
60                          event.getCheckpoints().add(node); //Adds new node to
                                array list of checkpoints if the node is of type
                                "CP or "MC".
61                      }
62                  } else {
63                      System.out.print("Invalid line format. Cancelling
                            loading of nodes.\n\n");
64                      reader.close();
65                      return false;
66                  }
67              }
68
69              if (!event.getNodes().isEmpty()) {
70                  System.out.print("Loading in of nodes successful.\n\n");
71                  reader.close();
72                  return true;
73              } else {
74                  System.out.print("Loading in of nodes unsuccessful. No nodes
                        in file.\n\n");
75                  reader.close();
76                  return false;
77              }
78          } catch (FileNotFoundException ex) {
79              Logger.getLogger(FileHandler.class.getName()).log(Level.SEVERE,
                    null, ex);
80          }
81
82          System.out.print("Could not open file that contains nodes.\n\n");
83          return false;
84      }
85
86      /**
87       * Method to read in all the details for the courses pertaining to an
               event.
88       *
89       * @param fileName The file name required to access the file needed.
90       * @param event The event instance.
91       * @return True if file loaded successfully, else false if it fails at
               any
92       * point.
93       */
94      public boolean readCourses(String fileName, Event event) throws
            IOException {
```

```
95          String input;
96          char courseLetter;
97          int numberOfNodes;
98          int[] nodes;
99          String[] subStrings;
100         String pattern = "(([A-Za-z]+)((\\s+\\d+)+)$)"; //Regular expression
                for courses file.
101
102         try {
103             BufferedReader reader = new BufferedReader(new FileReader(
                    fileName));
104
105             while ((input = reader.readLine()) != null) {
106                 if (input.matches(pattern)) { //Checks to make sure the line
                        is in the right format.
107                     subStrings = input.split("\\s+"); //Gets rid of
                            whitespace and separates the strings into substrings.
108                     courseLetter = subStrings[0].charAt(0); //Retrieves the
                            course letter.
109                     numberOfNodes = Integer.parseInt(subStrings[1]);
110                     nodes = new int[numberOfNodes];
111
112                     for (int counter = 0; counter < numberOfNodes; counter
                            ++) {
113                         if (event.checkNodeExists(Integer.parseInt(
                                subStrings[counter + 2]))) {
114                             nodes[counter] = Integer.parseInt(subStrings[
                                    counter + 2]);
115                         } else {
116                             System.out.print("Invalid node in course file
                                    found. Cancelling loading of courses\n\n");
117                             reader.close();
118                             return false;
119                         }
120                     }
121
122                     Course course = new Course(courseLetter, numberOfNodes,
                            nodes); //Creates new course with parameters read in.
123                     event.getCourses().add(course); //Adds new course to
                            array list of courses.
124                 } else {
125                     System.out.print("Invalid line format. Cancelling
                            loading of courses\n\n");
126                     reader.close();
127                     return false;
128                 }
129             }
130
131             if (!event.getCourses().isEmpty()) {
132                 System.out.print("Loading in of courses successful.\n\n");
133                 reader.close();
134                 return true;
135             } else {
136                 System.out.print("Loading in of courses unsuccessful. No
                        courses in file.\n\n");
137                 reader.close();
138                 return false;
139             }
140         } catch (FileNotFoundException ex) {
141             Logger.getLogger(FileHandler.class.getName()).log(Level.SEVERE,
                    null, ex);
142         }
143
```

```
144             System.out.print("Could not open file that contains courses.\n\n");
145             return false;
146         }
147 
148         /**
149          * Method to read in all the details for the competitors pertaining to
                an
150          * event.
151          *
152          * @param fileName The file name required to access the file needed.
153          * @param event The event instance.
154          * @return True if file loaded successfully, else false if it fails at
                any
155          * point.
156          */
157         public boolean readCompetitors(String fileName, Event event) throws
               IOException {
158             String input;
159             int competitorNumber;
160             char courseLetter;
161             String[] subStrings;
162             String competitorName;
163             String pattern = "(\\d+\\s+[A-Za-z]((\\s+[A-Za-z]{1}[a-z]+)+)$)"; //
                   Regular expression for competitors file.
164 
165             try {
166                 BufferedReader reader = new BufferedReader(new FileReader(
                       fileName));
167 
168                 while ((input = reader.readLine()) != null) {
169                     if (input.matches(pattern)) { //Checks to make sure the line
                           is in the right format.
170                         subStrings = input.split("\\s+"); //Gets rid of
                               whitespace and separates the strings into substrings.
171                         competitorNumber = Integer.parseInt(subStrings[0]); //
                               Retrieves the competitor number by parsing the string
                                into an int.
172 
173                         if (event.checkCourseExists(subStrings[1].charAt(0))) {
174                             courseLetter = subStrings[1].charAt(0); //Retrieves
                                   the course the competitor is entering in on.
175                         } else {
176                             System.out.print("Invalid course in competitor file
                                   found. Cancelling loading of competitors.\n\n");
177                             reader.close();
178                             return false;
179                         }
180 
181                         competitorName = subStrings[2];
182 
183                         if (subStrings.length > 3) {
184                             for (int counter = 3; counter < subStrings.length;
                                   counter++) {
185                                 competitorName += " " + subStrings[counter]; //
                                       Concatanates name substrings together.
186                             }
187                         }
188 
189                         Competitor competitor = new Competitor(competitorNumber,
                               courseLetter, competitorName, event); //Creates new
                               competitor with parameters read in.
190                         event.getCompetitors().add(competitor); //Adds new
                               competitor to array list of competitors.
```

```
                } else {
                    System.out.print("Invalid line format. Cancelling
                        loading of competitors.\n\n");
                    reader.close();
                    return false;
                }
            }

            if (!event.getCompetitors().isEmpty()) {
                System.out.print("Loading in of competitors successful.\n\n"
                    );
                reader.close();
                return true;
            } else {
                System.out.print("Loading in of competitors unsuccessful. No
                    competitors in file.\n\n");
                reader.close();
                return false;
            }
        } catch (FileNotFoundException ex) {
            Logger.getLogger(FileHandler.class
                    .getName()).log(Level.SEVERE, null, ex);
        }

        System.out.print("Could not open file that contains competitors.\n\n
            ");
        return false;
    }

    /**
     * Method to read in all the details for the checkpoint times pertaining
         to
     * an event.
     *
     * @param fileName The file name required to access the file needed.
     * @param event The event instance.
     * @return True if file loaded successfully, else false if it fails at
         any
     * point.
     */
    public boolean readTimes(String fileName, Event event) throws
        IOException, ParseException {
        String input;
        int currentLineNumber = 0;
        int lastLineNumber = event.getLastLineRead();
        char competitorStatus;
        int competitorNumber;
        int nodeNumber;
        String[] subStrings;
        String pattern = "([A-Z{1}]((\\s+\\d+){2})\\s
            +[0-2{1}][0-9{1}]:[0-5{1}][0-9{1}]$)"; //Regular expression for
            times file.
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");
        Date time;

        event.getRecords().clear(); //Empties array list.

        try {
            FileChannel channel = new RandomAccessFile(fileName, "rw").
                getChannel(); //Creates a channel for the file.
            FileLock lock = channel.lock(); //Blocks/Halts thread until lock
                aquired.

```

```
243             BufferedReader reader = new BufferedReader(new FileReader("
                    cp_times.txt"));
244
245             while ((input = reader.readLine()) != null) {
246                 currentLineNumber++;
247                 if (currentLineNumber > lastLineNumber) {
248                     if (input.matches(pattern)) { //Checks to make sure the
                            line is in the right format.
249                         subStrings = input.split("[\\s+]"); //Gets rid of
                                whitespace and separates the strings into
                                substrings.
250                         competitorStatus = subStrings[0].charAt(0); //
                                Retrieves competitor status.
251                         nodeNumber = Integer.parseInt(subStrings[1]); //
                                Retrieves the node number by parsing the string
                                into an int.
252                         competitorNumber = Integer.parseInt(subStrings[2]);
                                //Retrieves the competitor number by parsing the
                                string into an int.
253                         time = formatter.parse(subStrings[3]); //Retrieves
                                the time being recorded and formats it into 24
                                hour HH:MM.
254
255                         Competitor competitor = event.retrieveCompetitor(
                                competitorNumber);
256                         if (competitor.getStatus() == 'T') {
257                             competitor.incrementCheckpointIndex(); //
                                    Increments the competitor's checkpoint intdex
                                     by 1.
258                         }
259
260                         Record record = new Record(competitorStatus,
                                nodeNumber, competitorNumber, time); //Creates
                                new record with parameters read in.
261                         event.getRecords().add(record); //Adds new record to
                                 array list of records.
262                         competitor.setStatus(competitorStatus); //Updates
                                competitor's status.
263
264                         event.setLastLineRead(currentLineNumber);
265                         event.setLastRecordedTime(time);
266                     } else {
267                         System.out.print("Invalid line format. Cancelling
                                loading of times.\n\n");
268                         reader.close();
269                         lock.release();
270                         channel.close();
271                         return false;
272                     }
273                 }
274             }
275
276             event.setTimesFilesExistsTrue(); //Lets the event instance know
                    that an event does exist.
277             reader.close(); //Closes reader.
278             lock.release(); //Releases file lock.
279             channel.close(); //Closes channel ensuring lock release and
                    release of resources.
280             return true;
281         } catch (FileNotFoundException ex) {
282             System.out.print("Could not open file that contains times.\n\n")
                    ;
283         }
```

```
284             return false;
285         }
286 
287         /**
288          * Method to write a record on a line in the time records file.
289          *
290          * @param fileName The file name required to access the file needed.
291          * @param record The record to be written.
292          * @return True if file written to successfully, else false if it fails
                 at
293          * any point.
294          */
295         public boolean appendTimeRecord(String fileName, Record record) {
296             SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");
297 
298             try {
299                 FileChannel channel = new RandomAccessFile(fileName, "rw").
                        getChannel(); //Creates a channel for the file.
300                 FileLock lock = channel.lock();
301 
302                 FileWriter writer = new FileWriter(fileName, true); //True sets
                        append mode.
303                 writer.write(record.getCompetitorStatus() + " " + record.
                        getCheckpointNumber()
304                         + " " + record.getCompetitorNumber() + " " + formatter.
                            format(record.getTime()) + "\n");
305                 writer.close();
306                 lock.release();
307                 channel.close();
308                 return true;
309             } catch (IOException ex) {
310                 System.out.print("\nCould not open file for writing.\n\n");
311             }
312             return false;
313         }
314 }
```

6.3 GUI package

Listing 19: TypeWindow class.

```
1  /* File Name: TypeWindow.java
2   * Description: TypeWindow GUI class using swing.
3   * First Created: 17/03/2013
4   * Last Modified: 18/03/2013
5   */
6  package GUI;
7 
8  import Data_Structures.Event;
9  import java.awt.BorderLayout;
10 import java.awt.Dimension;
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import javax.swing.ButtonGroup;
14 import javax.swing.ImageIcon;
15 import javax.swing.JButton;
16 import javax.swing.JFrame;
17 import javax.swing.JLabel;
18 import javax.swing.JPanel;
19 import javax.swing.JRadioButton;
20 import javax.swing.border.EmptyBorder;
21 
```

```
22 /**
23  * @author Chris Savill, chs17@aber.ac.uk
24  */
25 public class TypeWindow extends JFrame implements ActionListener {
26 
27     private Event event;
28     private boolean medicalSelected;
29     private JFrame typeFrame;
30     private JPanel typePanel, bottomPanel;
31     private JLabel typeLabel;
32     private JRadioButton time, medical;
33     private ButtonGroup typeGroup;
34     private JButton next;
35 
36     /**
37      * Constructor for TypeWindow GUI class that sets up and launches GUI.
38      *
39      * @param event The event instance.
40      */
41     public TypeWindow(Event event) {
42         this.event = event;
43         medicalSelected = false;
44 
45         //Setup frame:
46         typeFrame = new JFrame("Checkpoint Type Selection");
47         typeFrame.setPreferredSize(new Dimension(300, 200));
48         typeFrame.setLocation(400, 200);
49         typeFrame.setLayout(new BorderLayout());
50         typeFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Sets the
               default close operation
51         typeFrame.setIconImage(new ImageIcon("horse.jpg").getImage()); //
               Loads an image and sets it as the frame icon
52         ////////////////////////////////////////////////////////////
53 
54         //Setup panels:
55         typePanel = new JPanel(new BorderLayout()); //Creates new JPanel.
56         typePanel.setBorder(new EmptyBorder(25, 25, 25, 25));  //Sets an
               invisible border to simulate a padding effect
57         typeFrame.add(typePanel, BorderLayout.NORTH); //Adds panel to frame
               and places it in NORTH container.
58         bottomPanel = new JPanel();
59         typeFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
               frame and places it in SOUTH container.
60         ////////////////////////////////////////////////////////////
61 
62         //Setup checkpoint panel components:
63         typeLabel = new JLabel("Select Checkpoint Type Below: ");
64         typePanel.add(typeLabel, BorderLayout.NORTH);
65 
66         time = new JRadioButton("Time Checkpoint");
67         time.setActionCommand("time");
68         time.addActionListener(this);
69         time.setSelected(true); //Defaults this button to be selected.
70         typePanel.add(time, BorderLayout.CENTER);
71         medical = new JRadioButton("Medical Checkpoint");
72         medical.setActionCommand("medical");
73         medical.addActionListener(this);
74         medical.setSelected(false);
75         typePanel.add(medical, BorderLayout.SOUTH);
76 
77         typeGroup = new ButtonGroup(); //Creates a group for the radio
               buttons to prevent both from being selected.
78         typeGroup.add(time);
```

```
79          typeGroup.add(medical);
80          ////////////////////////////////////////////////////////////
81
82          //Setup bottom panel components:
83          next = new JButton("Next");
84          next.setPreferredSize(new Dimension(100, 50));
85          bottomPanel.add(next);
86          next.addActionListener(this);
87          ////////////////////////////////////////////////////////////
88
89          //Finialise frame setup:
90          typeFrame.pack();
91          typeFrame.setVisible(true); //Makes the frame visible
92          ////////////////////////////////////////////////////////////
93      }
94
95      /**
96       * Method to handle actions performed.
97       *
98       * @param evt The event triggered.
99       */
100     @Override
101     public void actionPerformed(ActionEvent evt) {
102         String actionCommand = evt.getActionCommand();
103
104         switch (actionCommand) {
105             case "Next":
106                 if (medicalSelected == true) {
107                     typeFrame.setVisible(false);
108                     SelectionWindow selectionWindow = new SelectionWindow(
                            event, "MC", typeFrame);
109                 } else {
110                     typeFrame.setVisible(false);
111                     SelectionWindow selectionWindow = new SelectionWindow(
                            event, "CP", typeFrame);
112                 }
113
114                 typeFrame.dispose();
115                 this.dispose();
116                 break;
117             case "time":
118                 medicalSelected = false;
119                 break;
120             case "medical":
121                 medicalSelected = true;
122                 break;
123         }
124     }
125 }
```

Listing 20: SelectionWindow class.

```
1  /* File Name: SelectionWindow.java
2   * Description: SelectionWindow GUI class using swing.
3   * First Created: 16/03/2013
4   * Last Modified: 17/03/2013
5   */
6  package GUI;
7
8  import Data_Structures.Competitor;
9  import Data_Structures.Event;
10 import Data_Structures.Node;
```

```
11 import java.awt.BorderLayout;
12 import java.awt.Color;
13 import java.awt.Dimension;
14 import java.awt.event.ActionEvent;
15 import java.awt.event.ActionListener;
16 import javax.swing.DefaultListModel;
17 import javax.swing.ImageIcon;
18 import javax.swing.JButton;
19 import javax.swing.JFrame;
20 import javax.swing.JLabel;
21 import javax.swing.JList;
22 import javax.swing.JOptionPane;
23 import javax.swing.JPanel;
24 import javax.swing.JScrollPane;
25 import javax.swing.ScrollPaneConstants;
26 import javax.swing.border.EmptyBorder;
27 import javax.swing.border.LineBorder;
28 import javax.swing.event.ListSelectionEvent;
29 import javax.swing.event.ListSelectionListener;
30
31 /**
32  * @author Chris Savill, chs17@aber.ac.uk
33  */
34 public class SelectionWindow extends JFrame implements ActionListener,
       ListSelectionListener {
35
36     private Event event;
37     private int checkpoint;
38     private String type;
39     private int competitor;
40     private boolean checkpointSelected = false;
41     private boolean competitorSelected = false;
42     private JFrame selectionFrame, typeFrame;
43     private JPanel checkpointPanel, competitorPanel, bottomPanel;
44     private JLabel checkpointLabel, competitorLabel;
45     private DefaultListModel checkpointListModel, competitorListModel;
46     private JList checkpointList, competitorList;
47     private JScrollPane checkpointListScrollBar, competitorListScrollBar;
48     private JButton next;
49
50     /**
51      * Constructor for SelectionWindow GUI class, sets up and runs GUI.
52      * @param event The event instance.
53      * @param type The type of the checkpoint.
54      * @param typeFrame The JFrame this transitioned from.
55      */
56     public SelectionWindow(Event event, String type, JFrame typeFrame) {
57         typeFrame.dispose();
58         this.typeFrame = typeFrame;
59         this.event = event;
60         this.type = type;
61
62         //Setup frame:
63         selectionFrame = new JFrame("Checkpoint and Competitor Selection");
64         selectionFrame.setLocation(400, 200);
65         selectionFrame.setLayout(new BorderLayout());
66         selectionFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //
               Sets the default close operation
67         selectionFrame.setIconImage(new ImageIcon("horse.jpg").getImage());
               //Loads an image and sets it as the frame icon
68         ////////////////////////////////////////////////////////////
69
70         //Setup panels:
```

```
71          checkpointPanel = new JPanel(new BorderLayout()); //Creates new
                JPanel.
72          checkpointPanel.setBorder(new EmptyBorder(10, 25, 10, 25));  //Sets
                an invisible border to simulate a padding effect
73          selectionFrame.add(checkpointPanel, BorderLayout.WEST); //Adds panel
                 to frame and places it in WEST container.
74          competitorPanel = new JPanel(new BorderLayout());
75          competitorPanel.setBorder(new EmptyBorder(10, 25, 10, 25));
76          selectionFrame.add(competitorPanel, BorderLayout.EAST); //Adds panel
                 to frame and places it in EASTcontainer.
77          bottomPanel = new JPanel();
78          selectionFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
                 frame and places it in SOUTH container.
79          ////////////////////////////////////////////////////////////////
80
81          //Setup checkpoint panel components:
82          checkpointLabel = new JLabel("Select Checkpoint Below: ");
83          checkpointPanel.add(checkpointLabel, BorderLayout.NORTH);
84
85          checkpointListModel = new DefaultListModel();
86          checkpointList = new JList(checkpointListModel);
87          checkpointList.setBorder(new LineBorder(Color.BLACK));
88          checkpointPanel.add(checkpointList, BorderLayout.CENTER);
89          checkpointList.addListSelectionListener(this);
90
91          checkpointListScrollBar = new JScrollPane(checkpointList);
92          checkpointListScrollBar.setPreferredSize(new Dimension(50, 100));
93          checkpointListScrollBar.setVerticalScrollBarPolicy(
                ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED); //Adds
                vertical scrollbar to JList
94          checkpointListScrollBar.setHorizontalScrollBarPolicy(
                ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED); //Adds
                horizontal scrollbar to JList
95          checkpointPanel.add(checkpointListScrollBar);
96          ////////////////////////////////////////////////////////////////
97
98          //Setup competitor panel components:
99          competitorLabel = new JLabel("Select Competitor Below: ");
100         competitorPanel.add(competitorLabel, BorderLayout.NORTH);
101
102         competitorListModel = new DefaultListModel();
103         competitorList = new JList(competitorListModel);
104         competitorList.setBorder(new LineBorder(Color.BLACK));
105         competitorPanel.add(competitorList, BorderLayout.CENTER);
106         competitorList.addListSelectionListener(this);
107
108         competitorListScrollBar = new JScrollPane(competitorList);
109         competitorListScrollBar.setPreferredSize(new Dimension(400, 300));
110         competitorListScrollBar.setVerticalScrollBarPolicy(
                ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED); //Adds
                vertical scrollbar to JList
111         competitorListScrollBar.setHorizontalScrollBarPolicy(
                ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED); //Adds
                horizontal scrollbar to JList
112         competitorPanel.add(competitorListScrollBar);
113         ////////////////////////////////////////////////////////////////
114
115         //Setup bottom panel components:
116         next = new JButton("Next");
117         next.setPreferredSize(new Dimension(100, 50));
118         bottomPanel.add(next);
119         next.addActionListener(this);
120         ////////////////////////////////////////////////////////////////
```

```
121
122         //Finialise frame setup:
123         addCheckpoints();
124         addCompetitors();
125         selectionFrame.pack();
126         selectionFrame.setVisible(true); //Makes the frame visible
127         ////////////////////////////////////////////////////////////////
128     }
129
130     /**
131      * Method that adds the checkpoint checkpoints to the checkpoint JList
132      */
133     public void addCheckpoints() {
134         checkpointListModel.removeAllElements();
135
136         for (Node currentCheckpoint : event.getCheckpoints()) {
137             if (currentCheckpoint.getType().equals(type)) {
138                 checkpointListModel.addElement(currentCheckpoint.getNumber()
                        + ": " + currentCheckpoint.getType());
139             }
140         }
141     }
142
143     /**
144      * Method that adds the competitors to the competitor JList
145      */
146     public void addCompetitors() {
147         competitorListModel.removeAllElements();
148
149         for (Competitor currentCompetitor : event.getCompetitors()) {
150             competitorListModel.addElement("Competitor: " +
                    currentCompetitor.getNumber()
151                     + "   Course: " + currentCompetitor.getCourse() + "
                        Name: " + currentCompetitor.getName());
152         }
153     }
154
155     /**
156      * Method to handle actions performed.
157      * @param evt The event triggered.
158      */
159     @Override
160     public void actionPerformed(ActionEvent evt) {
161         String actionCommand = evt.getActionCommand();
162
163         if (actionCommand.equals("Next")) {
164             if (checkpointSelected == true && competitorSelected == true) {
165                 selectionFrame.setVisible(false);
166                 TimeWindow timeWindow = new TimeWindow(event, checkpoint,
                        type, competitor, selectionFrame, typeFrame);
167                 selectionFrame.dispose();
168                 this.dispose();
169             } else {
170                 JOptionPane.showMessageDialog(selectionFrame, "Please select
                        both a checkpoint and competitor.");
171             }
172         }
173     }
174
175     /**
176      * Method to handle values changing in a JList.
177      * @param evt The event triggered.
178      */
```

```
179     @Override
180     public void valueChanged(ListSelectionEvent evt) {
181 
182         if (!evt.getValueIsAdjusting()) {
183             JList list = (JList) evt.getSource();
184 
185             if (list.equals(checkpointList)) {
186                 checkpoint = event.retrieveCheckpointNumber(type, list.
                        getSelectedIndex(), list.getModel().getSize());
187                 checkpointSelected = true;
188             } else if (list.equals(competitorList)) {
189                 competitor = event.getCompetitors().get(list.
                        getSelectedIndex()).getNumber();
190                 competitorSelected = true;
191             }
192         }
193     }
194 }
```

Listing 21: TimeWindow class.

```
1  /* File Name: TimeWindow.java
2   * Description: TimeWindow GUI class using swing.
3   * First Created: 16/03/2013
4   * Last Modified: 17/03/2013
5   */
6  package GUI;
7 
8  import Data_Structures.Event;
9  import Data_Structures.Record;
10 import File_Handling.FileHandler;
11 import java.awt.BorderLayout;
12 import java.awt.Dimension;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.io.IOException;
16 import java.text.ParseException;
17 import java.util.Calendar;
18 import java.util.Date;
19 import java.util.logging.Level;
20 import java.util.logging.Logger;
21 import javax.swing.ImageIcon;
22 import javax.swing.JButton;
23 import javax.swing.JFrame;
24 import javax.swing.JLabel;
25 import javax.swing.JOptionPane;
26 import javax.swing.JPanel;
27 import javax.swing.JSpinner;
28 import javax.swing.SpinnerDateModel;
29 import javax.swing.border.EmptyBorder;
30 
31 /**
32  * @author Chris Savill, chs17@aber.ac.uk
33  */
34 public class TimeWindow extends JFrame implements ActionListener {
35 
36     private Event event;
37     private FileHandler fileHandler;
38     private int checkpoint;
39     private String type;
40     private int competitor;
41     private int status;
```

```
42      private JFrame timeFrame, typeFrame;
43      private JPanel timePanel, bottomPanel;
44      private JLabel timeLabel;
45      private JButton submit;
46      private Date date;
47      private SpinnerDateModel spinnerModel;
48      private JSpinner spinner;
49      private JSpinner.DateEditor dateEditor;
50
51      /**
52       * Constructor for TimeWindow GUI class that sets up and launches the
             GUI.
53       *
54       * @param event The event instance.
55       * @param checkpoint The checkpoint number.
56       * @param type The checkpoint type.
57       * @param competitor The competitor number.
58       * @param selectionFrame The JFrame this transitioned from.
59       * @param typeFrame The JFrame that is reopened after this JFrame closes
              .
60       */
61      public TimeWindow(Event event, int checkpoint, String type, int
            competitor, JFrame selectionFrame, JFrame typeFrame) {
62          selectionFrame.dispose();
63
64          this.typeFrame = typeFrame;
65          this.event = event;
66          this.checkpoint = checkpoint;
67          this.type = type;
68          this.competitor = competitor;
69          fileHandler = new FileHandler();
70
71          //Setup frame:
72          timeFrame = new JFrame("Time Of Record");
73
74          if (type.equals("MC")) {
75              status = getMedicalOptions();
76          } else {
77              status = 0; //Comeptitor status not a medical related status.
78          }
79
80          timeFrame.setLocation(400, 200);
81          timeFrame.setLayout(new BorderLayout());
82          timeFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Sets the
                 default close operation
83          timeFrame.setIconImage(new ImageIcon("horse.jpg").getImage()); //
                Loads an image and sets it as the frame icon
84          ////////////////////////////////////////////////////////////////
85
86          //Setup panels:
87          timePanel = new JPanel(new BorderLayout()); //Creates new JPanel.
88          timePanel.setBorder(new EmptyBorder(10, 25, 10, 25));  //Sets an
                invisible border to simulate a padding effect
89          timeFrame.add(timePanel, BorderLayout.WEST); //Adds panel to frame
                and places it in WEST container.
90          bottomPanel = new JPanel();
91          timeFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
                frame and places it in SOUTH container.
92          ////////////////////////////////////////////////////////////////
93
94          //Setup checkpoint panel components:
95          timeLabel = new JLabel("Select Time Below: ");
96          timePanel.add(timeLabel, BorderLayout.NORTH);
```

```
97
98          date = new Date();
99          spinnerModel = new SpinnerDateModel(date, null, null, Calendar.
                HOUR_OF_DAY);
100         spinner = new JSpinner(spinnerModel);
101         dateEditor = new JSpinner.DateEditor(spinner, "HH:mm"); //24-hour
                format.
102         spinner.setEditor(dateEditor);
103         timePanel.add(spinner, BorderLayout.CENTER);
104         ////////////////////////////////////////////////////////////////
105
106         //Setup bottom panel components:
107         submit = new JButton("Submit Checkpoint Record");
108         submit.setPreferredSize(new Dimension(225, 30));
109         bottomPanel.add(submit);
110         submit.addActionListener(this);
111         ////////////////////////////////////////////////////////////////
112
113         //Finialise frame setup:
114         timeFrame.pack();
115         timeFrame.setVisible(true); //Makes the frame visible
116         ////////////////////////////////////////////////////////////////
117     }
118
119     /**
120      * Method to handle actions performed.
121      *
122      * @param evt The event triggered.
123      */
124     @Override
125     public void actionPerformed(ActionEvent evt) {
126         String actionCommand = evt.getActionCommand();
127
128         if (actionCommand.equals("Submit Checkpoint Record")) {
129             try {
130                 if (!fileHandler.readTimes(event.getFileNames()[3], event))
                        {
131                     JOptionPane.showMessageDialog(timeFrame, "Failed to load
                            time records from file.");
132                 }
133             } catch (IOException | ParseException ex) {
134                 Logger.getLogger(TimeWindow.class.getName()).log(Level.
                        SEVERE, null, ex);
135             }
136
137             if (event.checkNewRecord(checkpoint, status, competitor, (Date)
                    spinner.getValue())) {
138                 char finalStatus = event.determineFinalStatus(checkpoint,
                        status, competitor);
139
140                 Record record = new Record(checkpoint, finalStatus,
                        competitor, (Date) spinner.getValue());
141                 event.getRecords().add(record);
142                 event.setLastLineRead(event.getLastLineRead() + 1);
143                 event.setLastRecordedTime((Date) spinner.getValue());
144
145                 fileHandler.appendTimeRecord(event.getFileNames()[3], record
                        );
146                 JOptionPane.showMessageDialog(timeFrame, "Time record
                        succesfully added.");
147             } else {
148                 JOptionPane.showMessageDialog(timeFrame, "Non-valid record.
                        Record will not added.");
```

```
149             }
150
151             timeFrame.dispose(); //Closes frame and releases resourses.
152             this.dispose(); //Releases resources.
153             TypeWindow typeFrame = new TypeWindow(event);
154
155         }
156     }
157
158     /**
159      * Method to get the user to select the status of the competitor at the
160      * medical checkpoint.
161      *
162      * @return The status of the competitor at the medical checkpoint.
163      */
164     public int getMedicalOptions() {
165         String[] options = new String[]{"Arriving", "Departing", "Excluded"
                };
166
167         int selection = JOptionPane.showOptionDialog(timeFrame, "Is the
                competitor being marked as 'Arriving',"
168                 + " 'Departing' or as 'Excluded' on medical grounds?", "
                        Medical Marking", JOptionPane.DEFAULT_OPTION,
169                 JOptionPane.PLAIN_MESSAGE, null, options, options[0]);
170
171         if (selection == 0) {
172             return 1; //Competitor status to be set to arriving at medical
                    checkpoint.
173         } else if (selection == 1) {
174             return 2; //Competitor status to be set to departing medical
                    checkpoint.
175         } else if (selection == 2) {
176             return 3; //Competitor status to be set to excluded based on
                    medical grounds.
177         }
178
179         return 0;
180     }
181 }
```

7 Clean build and compilation of Checkpoint Program

```
ant -f /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program clean jar
init:
deps-clean:
Updating property file: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build/built-clean.properties
Deleting directory /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build
clean:
init:
deps-jar:
Created dir: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build
Updating property file: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build/built-jar.properties
Created dir: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build/classes
Created dir: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build/empty
Created dir: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
```

```
    Checkpoint_Manager_Program/build/generated-sources/ap-source-output
Compiling 10 source files to /home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Checkpoint_Manager_Program/build/classes
Note: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/src/GUI/SelectionWindow.java uses unchecked or
     unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
compile:
Created dir: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/dist
Copying 1 file to /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/build
Nothing to copy.
Building jar: /home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/dist/Checkpoint_Manager_Program.jar
To run this application from the command line without Ant, try:
java -jar "/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Checkpoint_Manager_Program/dist/Checkpoint_Manager_Program.jar"
jar:
BUILD SUCCESSFUL (total time: 2 seconds)
```

8 Run through of Checkpoint Manager Program

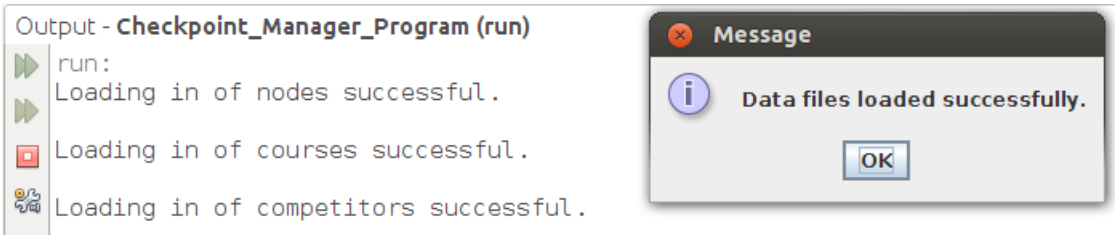

Figure 1: Start up of GUI letting user know data file loaded successfully (if they did) else the program would close.

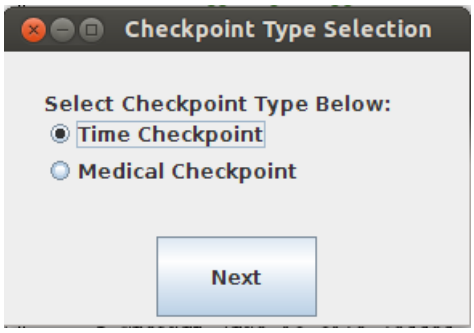

Figure 2: Checkpoint type selection window showing the time checkpoint type selected.

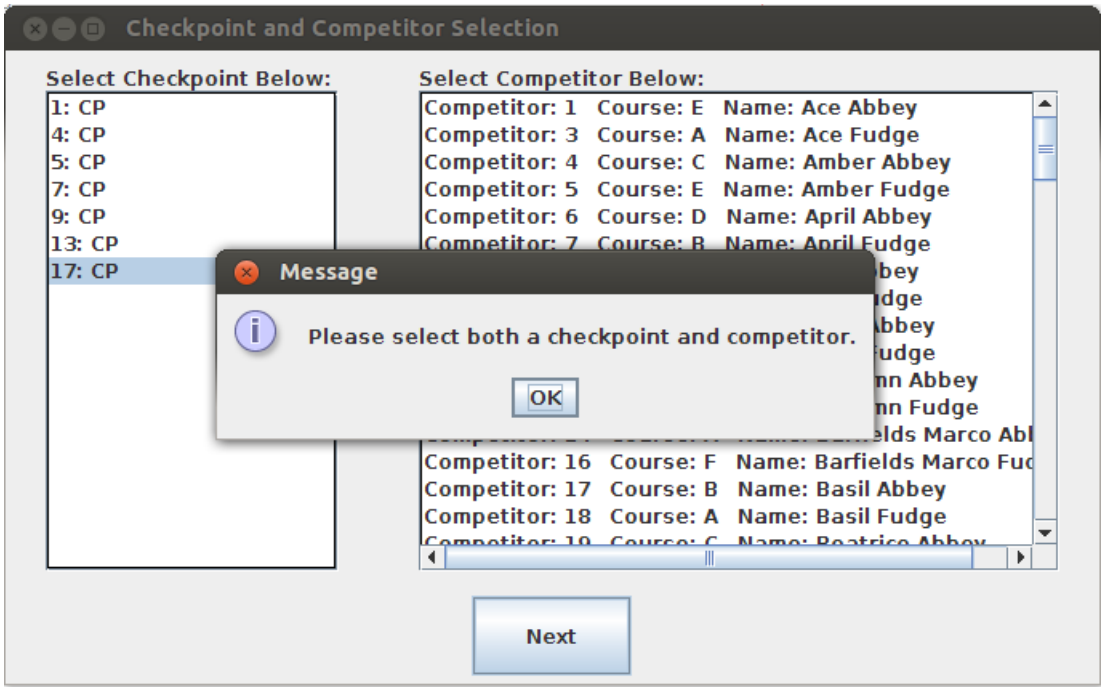

Figure 3: Message that pops up when both a checkpoint and competitor are not selected.

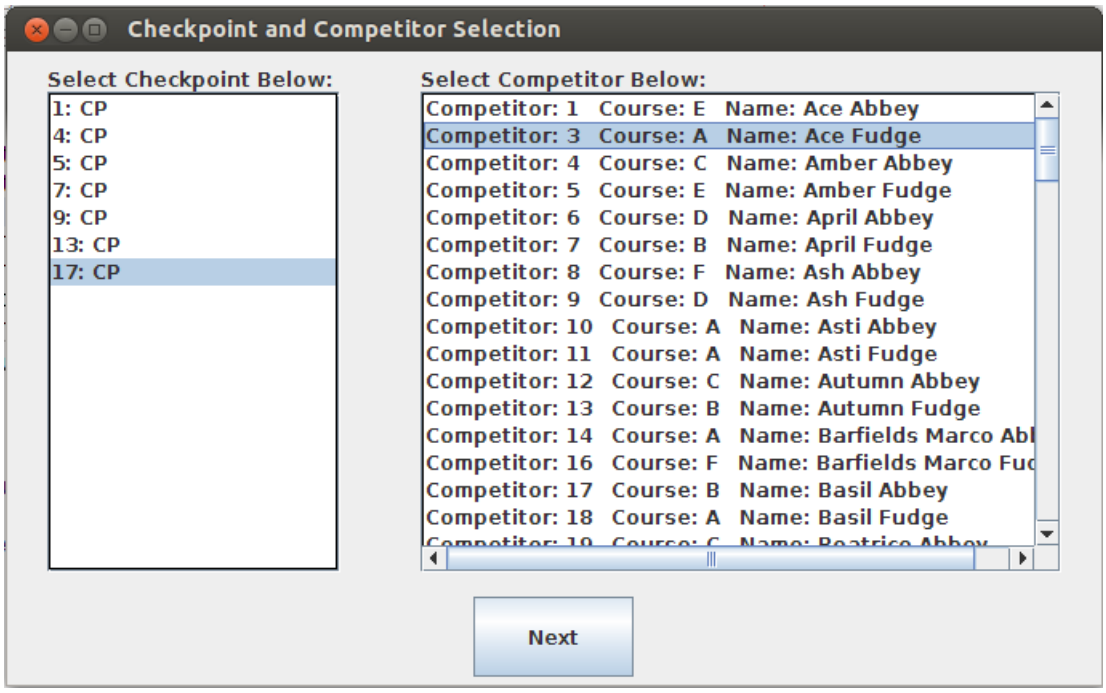

Figure 4: Time checkpoint 17 selection and competitor 3 selected within selection window.

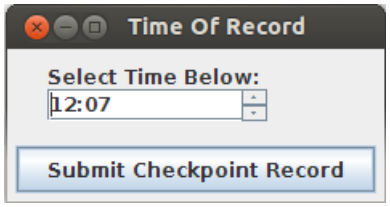

Figure 5: Time window where the user has to input the time for the new record.

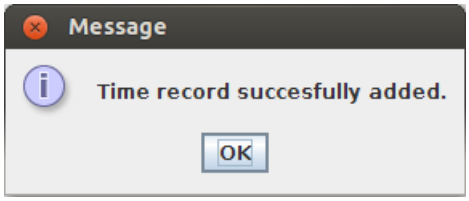

Figure 6: Message stating that the new record was successfully added and written to the time records file.

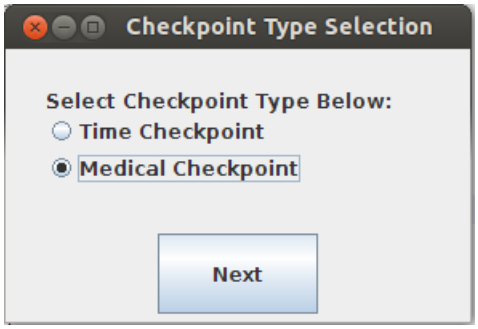

Figure 7: Checkpoint type selection window showing the medical checkpoint type selected.

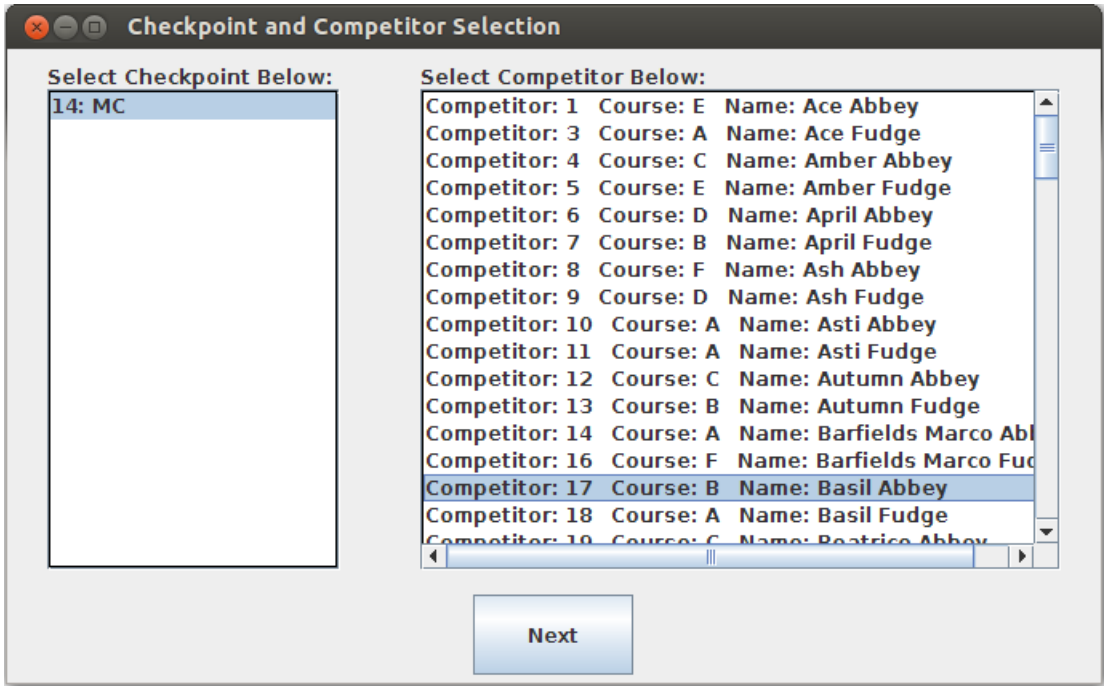

Figure 8: Medical checkpoint 14 selection and competitor 17 selected within the selection window.

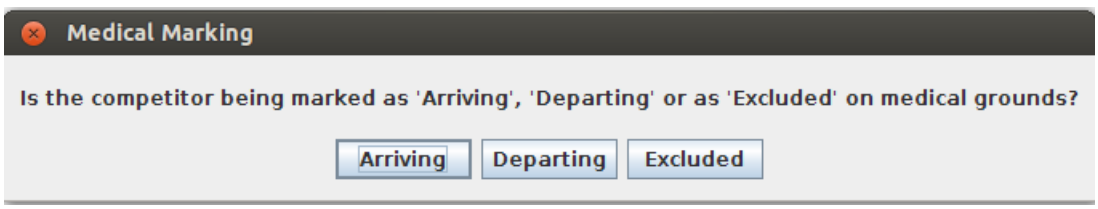

Figure 9: Prompt asking the user the status of the competitor at the chosen medical checkpoint.

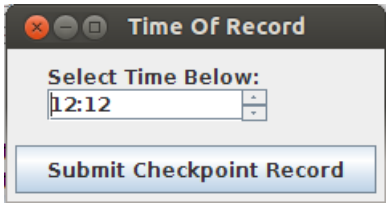

Figure 10: Time window with a valid time entered (a time after the last record time.

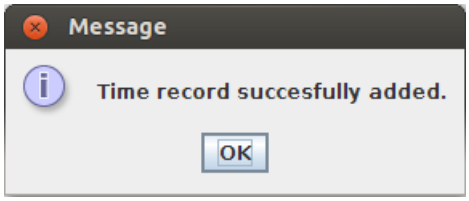

Figure 11: Message stating that the new record was successfully added and written to the time records file.

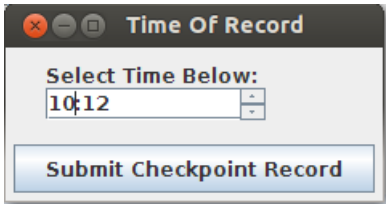

Figure 12: The same record as above time window with an invalid time entered (a time before the last record time.

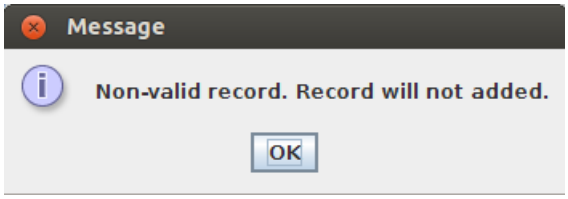

Figure 13: Message stating that the new record attempted to be added was invalid (either the time was before the last record, or the system determined that the status given cannot be correct due to the current status of the competitor.

9 Files created by execution of Checkpoint Manager Program

Listing 22: Copy of cp_times_1.txt file from event_3 that was read into the program plus 2 new records appended.

```
T 1 1 07:30
T 1 3 07:33
T 1 4 07:37
T 1 5 07:40
T 1 6 07:43
T 1 7 07:46
T 1 8 07:49
T 1 9 07:52
T 1 10 07:55
T 1 11 07:58
T 1 12 08:01
T 1 13 08:04
T 1 14 08:07
T 1 16 08:10
T 4 3 08:11
T 9 1 08:12
T 1 17 08:13
T 4 4 08:15
T 1 18 08:16
T 1 19 08:20
T 4 7 08:22
T 9 5 08:22
T 1 20 08:23
T 5 3 08:24
T 4 6 08:25
T 1 22 08:26
T 4 9 08:28
T 1 23 08:29
T 5 4 08:30
T 1 24 08:32
T 4 10 08:32
T 5 7 08:34
T 9 8 08:34
T 1 26 08:35
T 4 12 08:36
T 1 27 08:38
T 5 6 08:38
T 4 11 08:39
T 4 13 08:40
T 1 28 08:41
T 5 9 08:42
T 7 3 08:42
T 4 14 08:43
T 1 30 08:45
T 5 10 08:47
T 1 31 08:48
T 4 17 08:49
T 5 12 08:49
T 13 1 08:50
T 7 4 08:50
T 1 32 08:51
T 5 11 08:52
T 7 7 08:52
T 9 16 08:53
T 1 34 08:54
T 4 18 08:55
```

```
T 5 13 08:55
T 1 35 08:57
T 4 19 08:57
T 5 14 08:57
T 13 5 08:59
T 13 8 08:59
T 1 36 09:00
T 1 38 09:03
T 4 20 09:03
T 5 17 09:04
T 4 23 09:05
T 9 6 09:05
T 1 39 09:06
T 7 10 09:06
T 4 22 09:07
T 9 9 09:07
T 7 12 09:08
T 1 40 09:09
T 5 18 09:10
T 7 11 09:10
T 5 19 09:11
T 1 41 09:12
T 4 24 09:14
T 1 42 09:15
T 7 13 09:15
T 5 20 09:16
T 7 14 09:16
T 13 16 09:17
T 5 23 09:17
T 1 44 09:18
T 4 28 09:18
T 5 22 09:19
T 9 26 09:19
T 9 27 09:19
T 4 30 09:20
T 1 45 09:21
A 14 3 09:22
T 7 17 09:22
D 14 3 09:25
T 1 46 09:25
T 13 4 09:26
T 4 32 09:26
A 14 7 09:28
T 17 3 12:07
A 14 17 12:13
```

10 Clean build and compilation of Event Manager Program

```
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .clean-
    conf
make[1]: Entering directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Event_Manager_Program'
rm -f -r build/Debug
rm -f dist/Debug/GNU-Linux-x86/event_manager_program
make[1]: Leaving directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Event_Manager_Program'


CLEAN SUCCESSFUL (total time: 57ms)

"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-
    conf
make[1]: Entering directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Event_Manager_Program'
"/usr/bin/make"  -f nbproject/Makefile-Debug.mk dist/Debug/GNU-Linux-x86/
    event_manager_program
make[2]: Entering directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part
    /Event_Manager_Program'
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/loader.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/loader.o.d -o build/
    Debug/GNU-Linux-x86/loader.o loader.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/logger.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/logger.o.d -o build/
    Debug/GNU-Linux-x86/logger.o logger.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/updater.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/updater.o.d -o build/
    Debug/GNU-Linux-x86/updater.o updater.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/courses.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/courses.o.d -o build/
    Debug/GNU-Linux-x86/courses.o courses.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/competitors.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/competitors.o.d -o build
    /Debug/GNU-Linux-x86/competitors.o competitors.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/nodes.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/nodes.o.d -o build/Debug
    /GNU-Linux-x86/nodes.o nodes.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/main.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/main.o.d -o build/Debug/
    GNU-Linux-x86/main.o main.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/event.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/event.o.d -o build/Debug
    /GNU-Linux-x86/event.o event.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/tracks.o.d
gcc     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/tracks.o.d -o build/
    Debug/GNU-Linux-x86/tracks.o tracks.c
mkdir -p dist/Debug/GNU-Linux-x86
gcc      -o dist/Debug/GNU-Linux-x86/event_manager_program build/Debug/GNU-
```

```
    Linux-x86/loader.o build/Debug/GNU-Linux-x86/logger.o build/Debug/GNU-
    Linux-x86/updater.o build/Debug/GNU-Linux-x86/courses.o build/Debug/GNU-
    Linux-x86/competitors.o build/Debug/GNU-Linux-x86/nodes.o build/Debug/GNU
    -Linux-x86/main.o build/Debug/GNU-Linux-x86/event.o build/Debug/GNU-Linux
    -x86/tracks.o
make[2]: Leaving directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Event_Manager_Program'
make[1]: Leaving directory `/home/clsavill/GitHub/Runners_and_Riders_3_Part/
    Event_Manager_Program'


BUILD SUCCESSFUL (total time: 857ms)
```

11 Run through of Event Manager Program

```
Event Monitoring Program Launching...

Please enter in the file path and name of the event file:  Mission_Files/event_3/name.txt

Endurance Horse Race - The Main Event
27th June 2012
07:30

Event file loaded in successfully.
Event loading finished.


Please enter in the file path and name of the nodes file:  Mission_Files/event_3/nodes.txt
Head Node: Number: 1, Type: 0 = CP
Node: Number: 2, Type: 1 = JN
Node: Number: 3, Type: 1 = JN
Node: Number: 4, Type: 0 = CP
Node: Number: 5, Type: 0 = CP
Node: Number: 6, Type: 1 = JN
Node: Number: 7, Type: 0 = CP
Node: Number: 8, Type: 1 = JN
Node: Number: 9, Type: 0 = CP
Node: Number: 10, Type: 1 = JN
Node: Number: 11, Type: 1 = JN
Node: Number: 12, Type: 1 = JN
Node: Number: 13, Type: 0 = CP
Node: Number: 14, Type: -13 = MC
Node: Number: 15, Type: 1 = JN
Node: Number: 16, Type: 1 = JN
Node: Number: 17, Type: 0 = CP
Node: Number: 18, Type: 1 = JN

Nodes file loaded in successfully.
Node loading finished.


Please enter in the file path and name of the tracks file:  Mission_Files/event_3/tracks.txt
```

```
Head Track: Number: 1, Start: 1, End: 2, Max Time: 20
Track: Number: 2, Start: 2, End: 3, Max Time: 10
Track: Number: 3, Start: 3, End: 4, Max Time: 11
Track: Number: 4, Start: 4, End: 5, Max Time: 15
Track: Number: 5, Start: 5, End: 6, Max Time: 12
Track: Number: 6, Start: 6, End: 8, Max Time: 10
Track: Number: 7, Start: 6, End: 7, Max Time: 8
Track: Number: 8, Start: 7, End: 10, Max Time: 12
Track: Number: 9, Start: 8, End: 10, Max Time: 10
Track: Number: 10, Start: 8, End: 9, Max Time: 5
Track: Number: 11, Start: 3, End: 9, Max Time: 18
Track: Number: 12, Start: 9, End: 12, Max Time: 20
Track: Number: 13, Start: 2, End: 13, Max Time: 30
Track: Number: 14, Start: 12, End: 13, Max Time: 5
Track: Number: 15, Start: 10, End: 11, Max Time: 15
Track: Number: 16, Start: 11, End: 12, Max Time: 5
Track: Number: 17, Start: 11, End: 14, Max Time: 12
Track: Number: 18, Start: 14, End: 15, Max Time: 15
Track: Number: 19, Start: 15, End: 16, Max Time: 8
Track: Number: 20, Start: 16, End: 17, Max Time: 8
Track: Number: 21, Start: 17, End: 18, Max Time: 7
Track: Number: 22, Start: 15, End: 18, Max Time: 5

Tracks file loaded in successfully.
Track loading finished.


Please enter in the file path and name of the courses file:  Mission_Files/event_3/courses.txt

Head Course: ID: A, Number of Nodes: 21,  Nodes: [1,2,3,4,5,6,7,10,11,14,15,16,17,18,15,14,11,12,13,2,1]

Course: ID: B, Number of Nodes: 15, Nodes: [1,2,3,4,5,6,7,10,11,14,11,12,13,2,1]

Course: ID: C, Number of Nodes: 13, Nodes: [1,2,3,4,5,6,7,10,11,12,13,2,1]

Course: ID: D, Number of Nodes: 11, Nodes: [1,2,3,4,5,6,8,9,3,2,1]

Course: ID: E, Number of Nodes: 11, Nodes: [1,2,3,9,8,10,11,12,13,2,1]

Course: ID: F, Number of Nodes: 8, Nodes: [1,2,3,9,12,13,2,1]
```

```
Courses file loaded in successfully.
Course loading finished.


Please enter in the file path and name of the competitors file:  Mission_Files/event_3/entrants.txt
Head Competitor: Number: 1, Course: E, Name: Ace Abbey
Competitor: Number: 3, Course: A, Name: Ace Fudge
Competitor: Number: 4, Course: C, Name: Amber Abbey
Competitor: Number: 5, Course: E, Name: Amber Fudge
Competitor: Number: 6, Course: D, Name: April Abbey
Competitor: Number: 7, Course: B, Name: April Fudge
Competitor: Number: 8, Course: F, Name: Ash Abbey
Competitor: Number: 9, Course: D, Name: Ash Fudge
Competitor: Number: 10, Course: A, Name: Asti Abbey
Competitor: Number: 11, Course: A, Name: Asti Fudge
Competitor: Number: 12, Course: C, Name: Autumn Abbey
Competitor: Number: 13, Course: B, Name: Autumn Fudge
Competitor: Number: 14, Course: A, Name: Barfields Marco Abbey
Competitor: Number: 16, Course: F, Name: Barfields Marco Fudge
Competitor: Number: 17, Course: B, Name: Basil Abbey
Competitor: Number: 18, Course: A, Name: Basil Fudge
Competitor: Number: 19, Course: C, Name: Beatrice Abbey
Competitor: Number: 20, Course: A, Name: Beatrice Fudge
Competitor: Number: 22, Course: D, Name: Beau Abbey
Competitor: Number: 23, Course: C, Name: Beau Fudge
Competitor: Number: 24, Course: B, Name: Bella Abbey
Competitor: Number: 26, Course: F, Name: Bella Fudge
Competitor: Number: 27, Course: F, Name: Black Jack Abbey
Competitor: Number: 28, Course: A, Name: Black Jack Fudge
Competitor: Number: 30, Course: B, Name: Blue Abbey
Competitor: Number: 31, Course: B, Name: Blue Fudge
Competitor: Number: 32, Course: A, Name: Bobby Abbey
Competitor: Number: 34, Course: E, Name: Bobby Fudge
Competitor: Number: 35, Course: C, Name: Bubbles Abbey
Competitor: Number: 36, Course: D, Name: Bubbles Fudge
Competitor: Number: 38, Course: A, Name: Captain Abbey
Competitor: Number: 39, Course: B, Name: Captain Fudge
Competitor: Number: 40, Course: D, Name: Chalkie Abbey
Competitor: Number: 41, Course: F, Name: Chalkie Fudge
```

```
Competitor: Number: 42, Course: E, Name: Copper Abbey
Competitor: Number: 44, Course: B, Name: Copper Fudge
Competitor: Number: 45, Course: C, Name: Diamond Abbey
Competitor: Number: 46, Course: B, Name: Diamond Fudge
Competitor: Number: 47, Course: E, Name: Dinky Abbey
Competitor: Number: 48, Course: F, Name: Dinky Fudge
Competitor: Number: 49, Course: B, Name: Ebony Abbey
Competitor: Number: 50, Course: C, Name: Ebony Fudge
Competitor: Number: 51, Course: C, Name: Ginger Abbey
Competitor: Number: 52, Course: F, Name: Ginger Fudge
Competitor: Number: 53, Course: A, Name: Goldie Abbey
Competitor: Number: 55, Course: E, Name: Goldie Fudge
Competitor: Number: 56, Course: F, Name: Honey Abbey
Competitor: Number: 57, Course: C, Name: Honey Fudge
Competitor: Number: 58, Course: A, Name: Izzy Abbey
Competitor: Number: 59, Course: A, Name: Izzy Fudge
Competitor: Number: 60, Course: A, Name: Jasmine Abbey
Competitor: Number: 61, Course: F, Name: Jasmine Fudge
Competitor: Number: 62, Course: D, Name: Lady Abbey
Competitor: Number: 64, Course: B, Name: Lady Fudge
Competitor: Number: 65, Course: C, Name: Lady Tara Abbey
Competitor: Number: 66, Course: B, Name: Lady Tara Fudge
Competitor: Number: 67, Course: B, Name: Lemon Abbey
Competitor: Number: 68, Course: E, Name: Lemon Fudge
Competitor: Number: 69, Course: F, Name: Lord Abbey
Competitor: Number: 70, Course: E, Name: Lord Fudge
Competitor: Number: 71, Course: A, Name: Lucky Abbey
Competitor: Number: 74, Course: E, Name: Lucky Fudge
Competitor: Number: 76, Course: D, Name: Lord Abbey
Competitor: Number: 77, Course: B, Name: Lord Fudge
Competitor: Number: 78, Course: F, Name: Maddy Abbey
Competitor: Number: 79, Course: A, Name: Maddy Fudge
Competitor: Number: 80, Course: D, Name: Magic Abbey
Competitor: Number: 81, Course: D, Name: Magic Fudge
Competitor: Number: 83, Course: A, Name: Major Abbey
Competitor: Number: 85, Course: A, Name: Major Fudge
Competitor: Number: 86, Course: B, Name: Mattie Abbey
Competitor: Number: 87, Course: A, Name: Mattie Fudge
Competitor: Number: 89, Course: B, Name: Prince Abbey
Competitor: Number: 90, Course: A, Name: Prince Fudge
```

```
Competitor: Number: 91, Course: B, Name: Princess Abbey
Competitor: Number: 92, Course: B, Name: Princess Fudge
Competitor: Number: 93, Course: D, Name: Rosie Abbey
Competitor: Number: 94, Course: B, Name: Rosie Fudge
Competitor: Number: 95, Course: F, Name: Ruby Abbey
Competitor: Number: 97, Course: C, Name: Ruby Fudge
Competitor: Number: 98, Course: C, Name: Sapphire Abbey
Competitor: Number: 100, Course: F, Name: Sapphire Fudge
Competitor: Number: 101, Course: C, Name: Scarlet Abbey
Competitor: Number: 102, Course: F, Name: Scarlet Fudge
Competitor: Number: 103, Course: D, Name: sienna Abbey
Competitor: Number: 106, Course: B, Name: sienna Fudge
Competitor: Number: 107, Course: F, Name: Silver Abbey
Competitor: Number: 108, Course: A, Name: Silver Fudge
Competitor: Number: 109, Course: A, Name: Smokey Abbey
Competitor: Number: 110, Course: D, Name: Smokey Fudge
Competitor: Number: 111, Course: E, Name: Snowy Abbey
Competitor: Number: 113, Course: C, Name: Snowy Fudge
Competitor: Number: 114, Course: A, Name: sonic Abbey
Competitor: Number: 115, Course: D, Name: sonic Fudge
Competitor: Number: 117, Course: A, Name: Summer Abbey
Competitor: Number: 118, Course: E, Name: Summer Fudge
Competitor: Number: 121, Course: B, Name: Tango Abbey
Competitor: Number: 122, Course: A, Name: Tango Fudge
Competitor: Number: 123, Course: B, Name: Topaz Abbey
Competitor: Number: 124, Course: F, Name: Topaz Fudge
Competitor: Number: 126, Course: D, Name: Zizou Abbey
Competitor: Number: 127, Course: F, Name: Zizou Fudge

Competitors file loaded in successfully.
Competitor loading finished.
Loading Cycle Finished.
Press enter to continue.


=================================== MAIN MENU ====================================
|                                                                                 |
| 1: Query competitor for current location/status.                                |
| 2: Display how many competitors have not started yet.                           |
| 3: Display how many competitors are out on the courses.                         |
```

```
| 4: Display how many competitors have completed their course successfully.         |
| 5: Read in a file of times at which competitors have reached time checkpoints.    |
| 6: Display the result times for the successfully completed.                       |
| 7: Display the competitors who have been excluded.                                |
| 8: Exit program.                                                                  |
|                                                                                   |
=====================================================================================

Please select from one of the options above (number): 5

Please enter in the file path and name of the time record file:  Mission_Files/event_3/cp_times_1.txt

End of file reached.
Loading of times files complete.
Time record loading finished.

Press enter to continue.


================================== MAIN MENU ==================================
|                                                                                   |
| 1: Query competitor for current location/status.                                  |
| 2: Display how many competitors have not started yet.                             |
| 3: Display how many competitors are out on the courses.                           |
| 4: Display how many competitors have completed their course successfully.         |
| 5: Read in a file of times at which competitors have reached time checkpoints.    |
| 6: Display the result times for the successfully completed.                       |
| 7: Display the competitors who have been excluded.                                |
| 8: Exit program.                                                                  |
|                                                                                   |
=====================================================================================

Please select from one of the options above (number): 3

 Printing competitors that are out on a course...

=============================================================================================================
| Number  |                        Name                        | Course | Last Recorded Checkpoint | Presumed Location |
=============================================================================================================
|   001   | Ace Abbey                                          |   E    |          13              |     TN - 01       |
```

003	Ace Fudge	A	14	TN - 18
004	Amber Abbey	C	13	TC - 13
005	Amber Fudge	E	13	TN - 13
006	April Abbey	D	09	TN - 02
007	April Fudge	B	14	A - 14
008	Ash Abbey	F	13	TN - 13
009	Ash Fudge	D	09	TN - 02
010	Asti Abbey	A	07	TN - 15
011	Asti Fudge	A	07	TN - 15
012	Autumn Abbey	C	07	TN - 15
013	Autumn Fudge	B	07	TN - 08
014	Barfields Marco Abbey	A	07	TN - 08
016	Barfields Marco Fudge	F	13	TN - 13
017	Basil Abbey	B	07	TN - 08
018	Basil Fudge	A	05	TN - 07
019	Beatrice Abbey	C	05	TN - 07
020	Beatrice Fudge	A	05	TN - 05
022	Beau Abbey	D	05	TN - 05
023	Beau Fudge	C	05	TN - 05
024	Bella Abbey	B	04	TN - 04
026	Bella Fudge	F	09	TN - 12
027	Black Jack Abbey	F	09	TN - 12
028	Black Jack Fudge	A	04	TN - 04
030	Blue Abbey	B	04	TN - 04
031	Blue Fudge	B	01	TN - 03
032	Bobby Abbey	A	04	TC - 04
034	Bobby Fudge	E	01	TN - 11
035	Bubbles Abbey	C	01	TN - 02
036	Bubbles Fudge	D	01	TN - 02
038	Captain Abbey	A	01	TN - 02
039	Captain Fudge	B	01	TN - 01
040	Chalkie Abbey	D	01	TN - 01
041	Chalkie Fudge	F	01	TN - 01
042	Copper Abbey	E	01	TN - 01
044	Copper Fudge	B	01	TN - 01
045	Diamond Abbey	C	01	TN - 01
046	Diamond Fudge	B	01	TN - 01

Key: NS = Not Started, TC = Time Checkpoint, TN = Track Number,

```
A = Medical Checkpoint, D = Departed Medical Checkpoint.

Number of Competitors out on course: 38 out of 102

Current Event Time: 9:26.

Press enter to continue.


==================================== MAIN MENU ======================================
|                                                                                    |
| 1: Query competitor for current location/status.                                   |
| 2: Display how many competitors have not started yet.                              |
| 3: Display how many competitors are out on the courses.                            |
| 4: Display how many competitors have completed their course successfully.          |
| 5: Read in a file of times at which competitors have reached time checkpoints.     |
| 6: Display the result times for the successfully completed.                        |
| 7: Display the competitors who have been excluded.                                 |
| 8: Exit program.                                                                   |
|                                                                                    |
======================================================================================

Please select from one of the options above (number): 5

Please enter in the file path and name of the time record file:  Mission_Files/event_3/cp_times_2.txt

End of file reached.
Loading of times files complete.
Time record loading finished.

Press enter to continue.


==================================== MAIN MENU ======================================
|                                                                                    |
| 1: Query competitor for current location/status.                                   |
| 2: Display how many competitors have not started yet.                              |
| 3: Display how many competitors are out on the courses.                            |
| 4: Display how many competitors have completed their course successfully.          |
| 5: Read in a file of times at which competitors have reached time checkpoints.     |
```

```
| 6: Display the result times for the successfully completed.                    |
| 7: Display the competitors who have been excluded.                             |
| 8: Exit program.                                                               |
|                                                                                |
==================================================================================

Please select from one of the options above (number): 5

Please enter in the file path and name of the time record file:  Mission_Files/event_3/cp_times_3.txt

End of file reached.
Loading of times files complete.
Time record loading finished.

Press enter to continue.


================================== MAIN MENU ======================================
|                                                                                |
| 1: Query competitor for current location/status.                               |
| 2: Display how many competitors have not started yet.                          |
| 3: Display how many competitors are out on the courses.                        |
| 4: Display how many competitors have completed their course successfully.      |
| 5: Read in a file of times at which competitors have reached time checkpoints. |
| 6: Display the result times for the successfully completed.                    |
| 7: Display the competitors who have been excluded.                             |
| 8: Exit program.                                                               |
|                                                                                |
==================================================================================

Please select from one of the options above (number): 2

 Printing competitors that have not yet started...

================================================================================
| Number  |                         Name                         | Course | Location |
================================================================================
|   095   | Ruby Abbey                                           |   F    |    NS    |
|   097   | Ruby Fudge                                           |   C    |    NS    |
|   098   | Sapphire Abbey                                       |   C    |    NS    |
```

```
|   100   | Sapphire Fudge                                                  |   F   |    NS    |
|   101   | Scarlet Abbey                                                   |   C   |    NS    |
|   102   | Scarlet Fudge                                                   |   F   |    NS    |
|   103   | sienna Abbey                                                    |   D   |    NS    |
|   106   | sienna Fudge                                                    |   B   |    NS    |
|   107   | Silver Abbey                                                    |   F   |    NS    |
|   108   | Silver Fudge                                                    |   A   |    NS    |
|   109   | Smokey Abbey                                                    |   A   |    NS    |
|   110   | Smokey Fudge                                                    |   D   |    NS    |
|   111   | Snowy Abbey                                                     |   E   |    NS    |
|   113   | Snowy Fudge                                                     |   C   |    NS    |
|   114   | sonic Abbey                                                     |   A   |    NS    |
|   115   | sonic Fudge                                                     |   D   |    NS    |
|   117   | Summer Abbey                                                    |   A   |    NS    |
|   118   | Summer Fudge                                                    |   E   |    NS    |
|   121   | Tango Abbey                                                     |   B   |    NS    |
|   122   | Tango Fudge                                                     |   A   |    NS    |
|   123   | Topaz Abbey                                                     |   B   |    NS    |
|   124   | Topaz Fudge                                                     |   F   |    NS    |
|   126   | Zizou Abbey                                                     |   D   |    NS    |
|   127   | Zizou Fudge                                                     |   F   |    NS    |
================================================================================

Key: NS = Not Started.

Number of Competitors not started yet: 24 out of 102

Current Event Time: 11:39.

Press enter to continue.


================================== MAIN MENU ===================================
|                                                                              |
| 1: Query competitor for current location/status.                             |
| 2: Display how many competitors have not started yet.                        |
| 3: Display how many competitors are out on the courses.                      |
| 4: Display how many competitors have completed their course successfully.    |
| 5: Read in a file of times at which competitors have reached time checkpoints.|
| 6: Display the result times for the successfully completed.                  |
```

```
| 7: Display the competitors who have been excluded.                                  |
| 8: Exit program.                                                                    |
|                                                                                     |
=======================================================================================

Please select from one of the options above (number): 3

 Printing competitors that are out on a course...

=========================================================================================================================
| Number |                          Name                          | Course | Last Recorded Checkpoint | Presumed Location |
=========================================================================================================================
|   010  | Asti Abbey                                             |   A    |          13              |     TN - 01       |
|   011  | Asti Fudge                                             |   A    |          13              |     TN - 01       |
|   014  | Barfields Marco Abbey                                  |   A    |          13              |     TN - 13       |
|   018  | Basil Fudge                                            |   A    |          13              |     TN - 13       |
|   020  | Beatrice Fudge                                         |   A    |          13              |     TN - 13       |
|   028  | Black Jack Fudge                                       |   A    |          14              |     TN - 20       |
|   032  | Bobby Abbey                                            |   A    |          14              |     TN - 17       |
|   038  | Captain Abbey                                          |   A    |          17              |     TN - 22       |
|   039  | Captain Fudge                                          |   B    |          13              |     TN - 13       |
|   044  | Copper Fudge                                           |   B    |          14              |     TN - 14       |
|   045  | Diamond Abbey                                          |   C    |          13              |     TN - 01       |
|   049  | Ebony Abbey                                            |   B    |          14              |     TN - 17       |
|   050  | Ebony Fudge                                            |   C    |          13              |     TN - 13       |
|   051  | Ginger Abbey                                           |   C    |          13              |     TN - 13       |
|   052  | Ginger Fudge                                           |   F    |          13              |     TN - 01       |
|   055  | Goldie Fudge                                           |   E    |          13              |     TN - 13       |
|   056  | Honey Abbey                                            |   F    |          13              |     TN - 01       |
|   057  | Honey Fudge                                            |   C    |          07              |     TN - 16       |
|   058  | Izzy Abbey                                             |   A    |          07              |     TN - 17       |
|   060  | Jasmine Abbey                                          |   A    |          07              |     TN - 15       |
|   061  | Jasmine Fudge                                          |   F    |          13              |     TN - 13       |
|   062  | Lady Abbey                                             |   D    |          09              |     TN - 11       |
|   064  | Lady Fudge                                             |   B    |          07              |     TN - 08       |
|   065  | Lady Tara Abbey                                        |   C    |          07              |     TN - 08       |
|   066  | Lady Tara Fudge                                        |   B    |          07              |     TN - 08       |
|   067  | Lemon Abbey                                            |   B    |          07              |     TN - 08       |
|   068  | Lemon Fudge                                            |   E    |          09              |     TN - 15       |
|   069  | Lord Abbey                                             |   F    |          13              |     TN - 13       |
```

```
|   070   | Lord Fudge                                                   |   E    |            09            |      TN - 15       |
|   071   | Lucky Abbey                                                  |   A    |            05            |      TN - 05       |
|   074   | Lucky Fudge                                                  |   E    |            09            |      TN - 09       |
|   076   | Lord Abbey                                                   |   D    |            05            |      TC - 05       |
|   077   | Lord Fudge                                                   |   B    |            04            |      TN - 04       |
|   078   | Maddy Abbey                                                  |   F    |            09            |      TN - 12       |
|   079   | Maddy Fudge                                                  |   A    |            04            |      TN - 04       |
|   080   | Magic Abbey                                                  |   D    |            01            |      TN - 03       |
|   081   | Magic Fudge                                                  |   D    |            01            |      TN - 03       |
|   083   | Major Abbey                                                  |   A    |            01            |      TN - 03       |
|   085   | Major Fudge                                                  |   A    |            01            |      TN - 02       |
|   086   | Mattie Abbey                                                 |   B    |            01            |      TN - 02       |
|   087   | Mattie Fudge                                                 |   A    |            01            |      TN - 02       |
|   089   | Prince Abbey                                                 |   B    |            01            |      TN - 01       |
|   090   | Prince Fudge                                                 |   A    |            01            |      TN - 01       |
|   091   | Princess Abbey                                               |   B    |            01            |      TN - 01       |
|   092   | Princess Fudge                                               |   B    |            01            |      TN - 01       |
|   093   | Rosie Abbey                                                  |   D    |            01            |      TN - 01       |
|   094   | Rosie Fudge                                                  |   B    |            01            |      TN - 01       |
========================================================================================================================

Key: NS = Not Started, TC = Time Checkpoint, TN = Track Number,
A = Medical Checkpoint, D = Departed Medical Checkpoint.

Number of Competitors out on course: 47 out of 102

Current Event Time: 11:39.

Press enter to continue.


==================================== MAIN MENU ====================================
|                                                                                 |
| 1: Query competitor for current location/status.                               |
| 2: Display how many competitors have not started yet.                          |
| 3: Display how many competitors are out on the courses.                        |
| 4: Display how many competitors have completed their course successfully.      |
| 5: Read in a file of times at which competitors have reached time checkpoints.  |
| 6: Display the result times for the successfully completed.                    |
| 7: Display the competitors who have been excluded.                             |
```

```
| 8: Exit program.                                                                    |
|                                                                                     |
=======================================================================================

Please select from one of the options above (number): 4

 Printing competitors that have finished...

=======================================================================================
| Number  |                          Name                          | Course | Location |
=======================================================================================
|   001   | Ace Abbey                                              |   E    |    CC    |
|   003   | Ace Fudge                                              |   A    |    CC    |
|   004   | Amber Abbey                                            |   C    |    CC    |
|   005   | Amber Fudge                                            |   E    |    CC    |
|   006   | April Abbey                                            |   D    |    CC    |
|   007   | April Fudge                                            |   B    |    CC    |
|   008   | Ash Abbey                                              |   F    |    CC    |
|   009   | Ash Fudge                                              |   D    |    CC    |
|   012   | Autumn Abbey                                           |   C    |    CC    |
|   013   | Autumn Fudge                                           |   B    |    CC    |
|   016   | Barfields Marco Fudge                                  |   F    |    CC    |
|   017   | Basil Abbey                                            |   B    |    CC    |
|   019   | Beatrice Abbey                                         |   C    |    CC    |
|   022   | Beau Abbey                                             |   D    |    CC    |
|   024   | Bella Abbey                                            |   B    |    CC    |
|   026   | Bella Fudge                                            |   F    |    CC    |
|   027   | Black Jack Abbey                                       |   F    |    CC    |
|   030   | Blue Abbey                                             |   B    |    CC    |
|   031   | Blue Fudge                                             |   B    |    CC    |
|   034   | Bobby Fudge                                            |   E    |    CC    |
|   035   | Bubbles Abbey                                          |   C    |    CC    |
|   040   | Chalkie Abbey                                          |   D    |    CC    |
|   042   | Copper Abbey                                           |   E    |    CC    |
|   047   | Dinky Abbey                                            |   E    |    CC    |
|   048   | Dinky Fudge                                            |   F    |    CC    |
=======================================================================================

Number of Competitors completed course successfully: 25 out of 102
```

```
Current Event Time: 11:39.

Press enter to continue.


==================================== MAIN MENU ======================================
|                                                                                    |
| 1: Query competitor for current location/status.                                   |
| 2: Display how many competitors have not started yet.                              |
| 3: Display how many competitors are out on the courses.                            |
| 4: Display how many competitors have completed their course successfully.          |
| 5: Read in a file of times at which competitors have reached time checkpoints.     |
| 6: Display the result times for the successfully completed.                        |
| 7: Display the competitors who have been excluded.                                 |
| 8: Exit program.                                                                   |
|                                                                                    |
====================================================================================

Please select from one of the options above (number): 6

 Printing results...

=============================================================================
| Number  |                        Name                        | Status |  Time  |
=============================================================================
|   001   | Ace Abbey                                          |   CC   |  02:04 |
|   003   | Ace Fudge                                          |   CC   |  03:52 |
|   004   | Amber Abbey                                        |   CC   |  02:37 |
|   005   | Amber Fudge                                        |   CC   |  02:11 |
|   006   | April Abbey                                        |   CC   |  02:03 |
|   007   | April Fudge                                        |   CC   |  02:46 |
|   008   | Ash Abbey                                          |   CC   |  01:56 |
|   009   | Ash Fudge                                          |   CC   |  01:58 |
|   012   | Autumn Abbey                                       |   CC   |  02:30 |
|   013   | Autumn Fudge                                       |   CC   |  02:53 |
|   016   | Barfields Marco Fudge                              |   CC   |  01:55 |
|   017   | Basil Abbey                                        |   CC   |  02:49 |
|   019   | Beatrice Abbey                                     |   CC   |  02:27 |
|   022   | Beau Abbey                                         |   CC   |  02:02 |
|   024   | Bella Abbey                                        |   CC   |  02:54 |
```

```
|   026   | Bella Fudge                                                  |   CC   |   01:49  |
|   027   | Black Jack Abbey                                             |   CC   |   01:49  |
|   030   | Blue Abbey                                                   |   CC   |   02:43  |
|   031   | Blue Fudge                                                   |   CC   |   02:44  |
|   034   | Bobby Fudge                                                  |   CC   |   02:03  |
|   035   | Bubbles Abbey                                                |   CC   |   02:32  |
|   040   | Chalkie Abbey                                                |   CC   |   02:03  |
|   042   | Copper Abbey                                                 |   CC   |   02:05  |
|   047   | Dinky Abbey                                                  |   CC   |   02:10  |
|   048   | Dinky Fudge                                                  |   CC   |   01:54  |
================================================================================

Number of Competitors completed course successfully: 25 out of 102

Current Event Time: 11:39.

Press enter to continue.


================================== MAIN MENU ===================================
|                                                                              |
| 1: Query competitor for current location/status.                             |
| 2: Display how many competitors have not started yet.                        |
| 3: Display how many competitors are out on the courses.                      |
| 4: Display how many competitors have completed their course successfully.    |
| 5: Read in a file of times at which competitors have reached time checkpoints.|
| 6: Display the result times for the successfully completed.                  |
| 7: Display the competitors who have been excluded.                           |
| 8: Exit program.                                                             |
|                                                                              |
================================================================================

Please select from one of the options above (number): 7

 Printing excluded...

================================================================================
| Number  |                        Name                        | Status |  At Time |
================================================================================
|   023   | Beau Fudge                                         |   EI   |   09:49  |
```

```
|   036   | Bubbles Fudge                                                 |   EI   |   09:57  |
|   041   | Chalkie Fudge                                                 |   EI   |   11:05  |
|   046   | Diamond Fudge                                                 |   EI   |   11:13  |
|   059   | Izzy Fudge                                                    |   EI   |   11:10  |
================================================================================

Number of Competitors excluded: 5 out of 102

Key: EI = Excluded for taking an Incorrect Route, EM = Excluded for Medical Safety Reasons.

Current Event Time: 11:39.

Press enter to continue.


=================================== MAIN MENU ==================================
|                                                                              |
| 1: Query competitor for current location/status.                             |
| 2: Display how many competitors have not started yet.                        |
| 3: Display how many competitors are out on the courses.                      |
| 4: Display how many competitors have completed their course successfully.    |
| 5: Read in a file of times at which competitors have reached time checkpoints.|
| 6: Display the result times for the successfully completed.                  |
| 7: Display the competitors who have been excluded.                           |
| 8: Exit program.                                                             |
|                                                                              |
================================================================================

Please select from one of the options above (number): 8


Exiting Program...

RUN SUCCESSFUL (total time: 2m 5s)
```

12 Results list produced at the end of an event

12.1 Results of successful competitors

Number	Name	Status	Time
001	Ace Abbey	CC	02:04
003	Ace Fudge	CC	03:52
004	Amber Abbey	CC	02:37
005	Amber Fudge	CC	02:11
006	April Abbey	CC	02:03
007	April Fudge	CC	02:46
008	Ash Abbey	CC	01:56
009	Ash Fudge	CC	01:58
010	Asti Abbey	CC	03:49
011	Asti Fudge	CC	03:51
012	Autumn Abbey	CC	02:30
013	Autumn Fudge	CC	02:53
014	Barfields Marco Abbey	CC	03:49
016	Barfields Marco Fudge	CC	01:55
017	Basil Abbey	CC	02:49
018	Basil Fudge	CC	03:50
019	Beatrice Abbey	CC	02:27
020	Beatrice Fudge	CC	03:50
022	Beau Abbey	CC	02:02
024	Bella Abbey	CC	02:54
026	Bella Fudge	CC	01:49
027	Black Jack Abbey	CC	01:49
030	Blue Abbey	CC	02:43
031	Blue Fudge	CC	02:44
032	Bobby Abbey	CC	03:52
034	Bobby Fudge	CC	02:03
035	Bubbles Abbey	CC	02:32
038	Captain Abbey	CC	03:52
039	Captain Fudge	CC	02:51
040	Chalkie Abbey	CC	02:03
042	Copper Abbey	CC	02:05
045	Diamond Abbey	CC	02:29

047	Dinky Abbey	CC	02:10
048	Dinky Fudge	CC	01:54
049	Ebony Abbey	CC	03:04
050	Ebony Fudge	CC	02:35
051	Ginger Abbey	CC	02:32
052	Ginger Fudge	CC	01:58
055	Goldie Fudge	CC	02:12
056	Honey Abbey	CC	01:54
057	Honey Fudge	CC	02:36
058	Izzy Abbey	CC	03:53
060	Jasmine Abbey	CC	03:50
061	Jasmine Fudge	CC	01:55
064	Lady Fudge	CC	02:59
065	Lady Tara Abbey	CC	02:29
066	Lady Tara Fudge	CC	02:50
067	Lemon Abbey	CC	03:02
069	Lord Abbey	CC	01:54
070	Lord Fudge	CC	02:14
074	Lucky Fudge	CC	02:12
076	Lord Abbey	CC	02:03
077	Lord Fudge	CC	02:56
079	Maddy Fudge	CC	03:54
080	Magic Abbey	CC	02:04
081	Magic Fudge	CC	02:02
083	Major Abbey	CC	03:43
086	Mattie Abbey	CC	02:52
087	Mattie Fudge	CC	03:52
089	Prince Abbey	CC	02:59
090	Prince Fudge	CC	04:00
091	Princess Abbey	CC	03:01
092	Princess Fudge	CC	03:02
093	Rosie Abbey	CC	01:59
094	Rosie Fudge	CC	02:45
095	Ruby Abbey	CC	01:59
097	Ruby Fudge	CC	02:36
098	Sapphire Abbey	CC	02:30
100	Sapphire Fudge	CC	01:57
101	Scarlet Abbey	CC	02:35
102	Scarlet Fudge	CC	01:56
103	sienna Abbey	CC	02:02

107	Silver Abbey	CC	01:56
108	Silver Fudge	CC	03:43
109	Smokey Abbey	CC	03:55
110	Smokey Fudge	CC	02:03
113	Snowy Fudge	CC	02:28
114	sonic Abbey	CC	03:47
115	sonic Fudge	CC	02:03
117	Summer Abbey	CC	03:58
118	Summer Fudge	CC	02:10
121	Tango Abbey	CC	02:57
122	Tango Fudge	CC	04:02
123	Topaz Abbey	CC	02:54
124	Topaz Fudge	CC	01:53
126	Zizou Abbey	CC	02:03
127	Zizou Fudge	CC	01:55

Number of Competitors completed course successfully: 87 out of 102

Current Event Time: 16:48.

12.2 Table of excluded competitors

Number	Name	Status	At Time
023	Beau Fudge	EI	09:49
028	Black Jack Fudge	EI	11:46
036	Bubbles Fudge	EI	09:57
041	Chalkie Fudge	EI	11:05
044	Copper Fudge	EI	11:47
046	Diamond Fudge	EI	11:13
053	Goldie Abbey	EI	12:57
059	Izzy Fudge	EI	11:10
062	Lady Abbey	EI	13:01
068	Lemon Fudge	EI	12:54
071	Lucky Abbey	EI	13:56
078	Maddy Abbey	EI	12:55
085	Major Fudge	EI	14:23
106	sienna Fudge	EI	14:53

```
|   111   | Snowy Abbey                                                |   EI   |   13:36  |
================================================================================

Number of Competitors excluded: 15 out of 102

Key: EI = Excluded for taking an Incorrect Route, EM = Excluded for Medical Safety Reasons.

Current Event Time: 16:48.
```

13 Log file contents

```
Action: Read in a time records file, Date: Tue Mar 19 17:07:56 2013
Action: Viewed results of competitors that completed their course successfully, Date: Tue Mar 19 17:07:56 2013
Action: Queried competitor, Date: Tue Mar 19 17:08:25 2013
Action: Viewed results of competitors that completed their course successfully, Date: Tue Mar 19 17:08:30 2013
Action: Quit Program, Date: Tue Mar 19 17:08:32 2013
Action: Read in a time records file, Date: Tue Mar 19 17:15:06 2013
Action: Viewed list of competitors out on course, Date: Tue Mar 19 17:15:13 2013
Action: Read in a time records file, Date: Tue Mar 19 17:15:38 2013
Action: Read in a time records file, Date: Tue Mar 19 17:15:56 2013
Action: Viewed list of competitors not started, Date: Tue Mar 19 17:16:03 2013
Action: Viewed list of competitors out on course, Date: Tue Mar 19 17:16:08 2013
Action: Viewed list of competitors that have finished, Date: Tue Mar 19 17:16:13 2013
Action: Viewed results of competitors that completed their course successfully, Date: Tue Mar 19 17:16:17 2013
Action: Viewed results of competitors that were excluded, Date: Tue Mar 19 17:16:19 2013
Action: Quit Program, Date: Tue Mar 19 17:16:27 2013
Action: Read in a time records file, Date: Wed Mar 20 11:19:11 2013
Action: Read in a time records file, Date: Wed Mar 20 11:19:25 2013
Action: Read in a time records file, Date: Wed Mar 20 11:19:31 2013
Action: Read in a time records file, Date: Wed Mar 20 11:19:38 2013
Action: Read in a time records file, Date: Wed Mar 20 11:19:43 2013
Action: Viewed results of competitors that completed their course successfully, Date: Wed Mar 20 11:19:56 2013
Action: Viewed results of competitors that were excluded, Date: Wed Mar 20 11:20:00 2013
```