## CS22510 Assignment 1 Runners and Riders "Out and About"

Chris Savill chs17@aber.ac.uk
March 20, 2013

### Contents

| 1         | Description of three programs                         | 3  |
|-----------|---|----|
|           | 1.1 Event Creation Program                            | 3  |
|           | 1.2 Checkpoint Manager Program                        | 3  |
|           | 1.3 Event Manager Program                             | 3  |
| 2         | Code for the Event Creation Program                   | 3  |
| 3         | Clean build and compilation of Event Creation Program | 17 |
| 4         | Run through of Event Creation Program                 | 18 |
| 5         | Files created by execution of Event Creation Program  | 18 |
| 6         | Code for Checkpoint Manager Program                   | 18 |
| 7         | Clean build and compilation of Checkpoint Program     | 45 |
| 8         | Run through of Checkpoint Manager Program             | 46 |
| 9         | Files created by execution of Event Creation Program  | 46 |
| 10        | Clean build and compilation of Event Manager Program  | 46 |
| 11        | Run through of Event Manager Program                  | 48 |
| <b>12</b> | Results list produced at the end of an event          | 64 |
| 13        | Lor file contents                                     | 64 |

#### 1 Description of three programs

- 1.1 Event Creation Program
- 1.2 Checkpoint Manager Program
- 1.3 Event Manager Program

#### 2 Code for the Event Creation Program

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

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

Listing 2: Main method and menu file.

```
* Author: Chris Savill, chs17@aber.ac.uk
2
    * File Name: competitor.cpp
 3
 4
    * Description: cpp file that contains function definitions for the start-up
         of the event creation program.
    * First Created: 11/03/2013
5
 6
    * Last Modified: 14/03/2013
 7
8
   #include "creator.h"
9
   #include <iostream>
10
   #include <cstdlib>
11
12
   #include <limits>
13
   using namespace std;
14
15
   /* Main function that just calls a function that takes over. */
16
17
   int main(int argc, char** argv) {
       Event *event = new Event();
18
19
       ecp_menu(event);
20
21
       return 0;
22 || }
```

```
23
24
   \primest Function to get the user's input for accepting or rejecting their inputs.
25
   bool get_acceptance() {
      char option;
26
27
28
      do {
29
          cout << "If yes press 'y' then 'Enter'" << endl << "If no press 'n'</pre>
             then 'Enter'" << endl;
30
          cin.clear();
31
          option = cin.get();
32
          cin.ignore(numeric_limits < streamsize >:: max(), '\n');
33
          if (option == 'y') return true;
34
35
          else if (option == 'n') return false;
36
          else cout << "Invalid option selected" << endl;</pre>
37
      } while (option != 'y' && option != 'n');
38
   }
39
40
   \slash* Function that displays the main menu for the event creation program. */
   void ecp_menu(Event *event) {
41
42
      int option; //Field to store the user's option input.
43
44
      do {
          45
             << endl;
46
          cout << "* Runners and Riders Event Creation Program Main Menu *"</pre>
             << endl;
          47
             << 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
             << endl << endl;
55
56
          cout << "Please enter in an option from the above an press 'Enter':</pre>
57
          cin.clear();
58
          cin >> option;
59
          cin.ignore();
60
          switch (option) {
61
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;
                  case 6:
77
78
                      delete(event);
79
                      cout << "Exiting program..." << endl << endl;</pre>
80
                      break;
81
                 default:
82
                      cout << "Please enter in a valid option." << endl << endl;</pre>
83
        } while (option != 6);
84
85 || }
```

Listing 3: Header file Event class.

```
1 ||
 2
    * Author: Chris Savill, chs17@aber.ac.uk
 3
    * File Name: event.h
4
    st Description: Header file for the Event class.
    * First Created: 11/03/2013
5
 6
    * Last Modified: 14/03/2013
7
    */
8
   #ifndef EVENT_H
9
   #define EVENT_H
10
11
12
   #include <memory>
13
   #include "competitor.h"
14
   #include "course.h"
   #include <vector>
15
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.
       std::vector<Course*> *courses; //Array of courses that are part of an
31
           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
   public:
37
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.
       void export_competitors(); //Member function that will handle the
44
           exporting of the array of competitors to a '.txt' file.
       void export_courses(); //Member function that will handle the exporting
45
          of the array of courses to a '.txt' file.
46
   };
47
48 #endif /* EVENT_H */
```

Listing 4: Cpp file for Event class.

```
1
 2
    * Author: Chris Savill, chs17@aber.ac.uk
 3
    * File Name: event.cpp
    * Description: cpp file that contains member function definitions for the
 4
        event class.
 5
    * First Created: 11/03/2013
 6
    * Last Modified: 14/03/2013
 7
 8
9
   #include "event.h"
   #include "creator.h"
10
11
   #include <iostream>
12
   #include <stdlib.h>
13
   #include <fstream>
   #include <sstream>
14
   #include <limits>
15
16
17
   using namespace std;
18
19
   /st Member function that returns a pointer to the vector of courses. st/
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</pre>
                    characters): ";
32
                cin.clear();
33
                getline(cin, name);
34
           } while (name.length() > MAX_EVENT_NAME_LENGTH);
35
36
            cout << 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
   /* Member function to get the user to input the date of the event. */
43
   void Event::set_date() {
```

```
45
        bool date_chosen = false;
46
        string date;
47
48
        do {
49
            do {
                 cout << endl << endl << "Please enter in the date for the event</pre>
50
                    (no more than 19 characters): ";
51
                 cin.clear();
52
                 getline(cin, date);
            } while (date.length() > MAX_DATE_LENGTH);
53
54
55
             cout << endl << endl << "Are you happy with the date: '" << date <<
                "'?" << endl;
             date_chosen = get_acceptance();
56
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];
        int hours;
68
69
        int minutes;
70
        string start_time;
71
        string string_hours;
72
        string string_minutes;
73
74
        do {
             do {
75
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();
                 cin >> input;
78
                 cin.ignore(numeric_limits < streamsize >:: max(), '\n');
79
80
                 cout << endl;</pre>
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
                     if (hours \leq 23 && hours \geq 00) { //Makes sure that the
85
                        hours are in 24-hour format.
                         cout << "Valid hours entered." << endl << endl;</pre>
86
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 {
                 cout << endl << "Please now enter in the minutes: ";</pre>
93
94
                 cin.clear();
95
                 cin >> input;
96
                 cin.ignore(numeric_limits < streamsize >:: max(), '\n');
97
                 cout << endl;</pre>
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;</pre>
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
            cout << endl << "Are you happy with the start time: '" <<</pre>
109
                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;</pre>
115
        string_hours = string_retriever.str();
116
        string_retriever.str(""); //Clears the string stream.
117
        string_retriever << minutes;</pre>
        string_minutes = string_retriever.str();
118
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
     * Oparam 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;</pre>
129
        else {
            Competitor *competitor = new Competitor((competitors->size() + 1),
130
                this);
131
             competitors ->push_back(competitor);
132
             cout << "New competitor added to event." << endl << endl;</pre>
133
             cout << "Competitor number: " << competitors -> back() -> get_number();
             cout << "Competitor name: " << competitors->back()->get_name() <<</pre>
134
135
            cout << "Course: " << competitors->back()->get_course() << endl;</pre>
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;</pre>
        cout << "Course letter: " << courses->back()->get_letter() << endl;</pre>
144
        cout << "Number of course nodes: " << courses->back()->
145
            get_number_of_nodes() << endl;</pre>
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;</pre>
153
    }
154
```

```
155 \parallel /* 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()) {
            competitors_file << this->name << "\n" << this->date << "\n" << this
161
                ->start_time;
162
            competitors_file.close();
163
            cout << "Event successfully exported to 'name.txt'." << endl << endl</pre>
164
        } else cout << "File 'name.txt' could not be written." << endl;</pre>
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;</pre>
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++) {
                     competitors_file << this->competitors->at(counter)->
176
                        get_number() << " " << this->competitors->at(counter)->
                        get_course()
                             << " " << this->competitors->at(counter)->get_name()
177
                                  << "\n";
                }
178
179
180
                 competitors_file.close();
181
                 cout << "Competitors successfully exported to 'entrants.txt'."</pre>
                    << endl << endl;
182
            } else cout << "File 'entrants.txt' could not be written." << endl;</pre>
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() {
        if (courses->empty()) cout << "No courses to export. Exporting cancelled
188
            ." << endl << endl;
189
        else {
190
            ofstream courses_file;
191
            courses_file.open("courses.txt", ios::out);
192
193
            if (courses_file.is_open()) {
                 for (int counter = 0; counter < this->courses->size(); counter
194
                    ++) {
195
                     courses_file << this->courses->at(counter)->get_letter() <</pre>
                        " " << this->courses->at(counter)->get_number_of_nodes();
196
197
                     for (int counter2 = 0; counter2 < this->courses->at(counter)
                        ->get_number_of_nodes(); counter2++) {
                         courses_file << " " << this->courses->at(counter)->
198
                             get_node(counter2);
199
                     }
200
                     courses_file << "\n";</pre>
                }
201
```

```
202
203
                  courses_file.close();
204
                  cout << "Courses successfully exported to 'courses.txt'." <<</pre>
                     endl << endl;</pre>
             } else cout << "File 'courses.txt' could not be written." << endl;</pre>
205
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;</pre>
         cout << "Event date: " << this->date << endl;</pre>
218
219
         cout << "Event start time: " << this->start_time << endl << endl;</pre>
220
221
222
    /* Destructor for Event class. */
223
    Event::~Event() {
224
        delete(competitors);
225
        delete(courses);
226 || }
```

Listing 5: Header file for Course class.

```
1 ||
    * Author: Chris Savill, chs17@aber.ac.uk
2
3
    * File Name: course.h
4
    * Description: Header file for the Course class.
    * First Created: 11/03/2013
5
6
    * Last Modified: 14/03/2013
7
    */
8
9
   #ifndef COURSE_H
10
   #define COURSE_H
11
12
   #include <memory>
   #include <vector>
13
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.
       std::vector<int> *nodes_available; //An array of nodes that are
22
           available to select from, read in from the 'nodes.txt' file.
23
       void set_letter(Event *event); //Member function that will set the
24
           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.
       bool duplicated_last_node(int number); //Member function to check if the
28
            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
   public:
31
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
   #endif /* COURSE_H */
39 ||
```

Listing 6: Cpp file for Course class.

```
1
 2
    * Author: Chris Savill, chs17@aber.ac.uk
 3
    * File Name: course.cpp
    * Description: cpp file that contains member function definitions for the
 4
       course class.
    * First Created: 11/03/2013
 5
    * Last Modified: 14/03/2013
 6
 7
 8
   #include "course.h"
9
10
   #include "creator.h"
   #include <iostream>
11
12
   #include <fstream>
   #include <sstream>
13
   #include <limits>
14
15
16
   using namespace std;
17
18
   /* Member function to return a course's letter. */
19
   char Course::get_letter() {
20
       return this->letter;
21
22
23
   /* Member function to return a course's number of nodes. */
24
   int Course::get_number_of_nodes() {
25
       return this->number_of_nodes;
26
27
28
   /* Member function to return a node from the course's vector of nodes. */
29
   int Course::get_node(int index) {
30
       return this->nodes->at(index);
31
   }
32
   /* Member function that checks if the letter given be the user matches any
33
      of the course letters. */
34
   bool checkCourseExists(char letter, Event *event) {
       for (int counter = 0; counter < event->getCourses()->size(); counter++)
35
            if (letter == event->getCourses()->at(counter)->get_letter()) return
36
                true; //Checks if letter matches any of the course letters.
37
       }
38
39
       return false; //Return false if no match found.
40
   }
41
42
   /* Member function that will set the letter of the course. */
```

```
43 ||
   void Course::set_letter(Event *event) {
44
       bool valid_letter = false;
45
       bool letter_chosen = false;
46
       char letter;
47
48
       do {
49
            do {
50
                cout << endl << "Please enter in the course letter for</pre>
                   the course: ";
51
                cin.clear();
52
                letter = cin.get();
53
                cin.ignore(numeric_limits < streamsize >:: max(), '\n');
54
55
                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.
56
                else {
57
                    cout << "Please enter in a valid course letter that does not</pre>
                         already exist in this event, a-z or A-Z." << endl <<
                        endl;
                    valid_letter = false;
58
                }
59
60
           } while (valid_letter == false);
61
            cout << endl << "Are you happy with the course letter: '" << letter
62
               << "'?" << endl;
63
            letter_chosen = get_acceptance();
       } while (letter_chosen == false);
64
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 {
            cout << endl << "Please enter in the number of nodes for</pre>
75
               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: '"</pre>
               << number << "'?" << endl;
            number_chosen = get_acceptance();
81
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
       nodes_file.open("nodes.txt", ios::in);
93
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;</pre>
            cout << "Nodes read in: " << nodes_available->at(0);
104
105
            for (int counter = 1; counter < nodes_available -> size(); counter++)
                cout << ", " << nodes_available->at(counter);
106
             cout << endl << endl;</pre>
        } else cout << "File 'nodes.txt' could not be opened. Please check file
107
            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</pre>
                    course: ";
119
                 getline(cin, input);
                 stringstream int_retriever(input);
120
121
                 int_retriever >> number;
            } while (duplicated_last_node(number) || !check_node_exists(number))
122
                ; //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: '" <<</pre>
                number << "'?" << endl;</pre>
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</pre>
                    number to add." << endl;</pre>
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. */
    bool Course::check_node_exists(int number) {
144
        for (int counter = 0; counter < this->nodes_available->size(); counter
145
            ++) {
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</pre>
            add." << endl;</pre>
        return false; //Returns false if the number entered does not exist in
150
            the vector of nodes available.
151
152
153
    /* Constructor for Course class. */
    Course::Course(Event *event) {
154
155
        this->nodes = new vector <int>();
156
        this->nodes_available = new vector<int>();
157
        if (read_nodes_available()) {
158
159
             set_letter(event);
160
             set_number_of_nodes();
161
162
             for (int counter = 0; counter < number_of_nodes - 1; counter++) {</pre>
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;</pre>
168 || }
169
170
    /* Destructor for Course class. */
    Course:: Course() {
171
172
        delete(nodes);
173
        delete(nodes_available);
174 || }
```

Listing 7: Header file for Competitor class.

```
1
 2
    * Author: Chris Savill, chs17@aber.ac.uk
 3
    * File Name: competitor.h
    st Description: Header file for the Competitor class.
 4
    * First Created: 11/03/2013
 5
    * Last Modified: 14/03/2013
 6
 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:
       int number; // The competitor's unique identification number for an event
21
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
   public:
29
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 */
```

Listing 8: Cpp file for Competitor class.

```
1 ||
    * Author: Chris Savill, chs17@aber.ac.uk
 2
 3
    * File Name: competitor.cpp
 4
    st 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
   #include "competitor.h"
9
10
   #include "creator.h"
   #include <ctype.h>
11
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() {
       return this->name;
24
25
26
27
   /* Member function to return a competitor's course. */
28
   char Competitor::get_course() {
29
       return this->course;
30
   }
31
   /* Member function that will set the number of the competitor.
32
33
    * Oparam number The number for the competitor.
34
    */
35
   void Competitor::set_number(int number) {
36
       this->number = number;
37
   }
38
   /* Member function that will set the name of the competitor. */
39
40
   void Competitor::set_name() {
41
       bool name_chosen = false;
42
       string name;
43
44
       do {
45
            do {
```

```
46
                cout << endl << "Please enter in the name for the</pre>
                   competitor (no more than 50 characters): ";
47
                getline(cin, name);
           } while (name.length() > MAX_COMPETITOR_NAME_LENGTH);
48
49
50
            cout << 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 {
                cout << endl << "List of courses available for the</pre>
67
                   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++)
                        cout << ", " << event->getCourses()->at(counter)->
71
                            get_letter();
72
                }
73
                cout << endl << endl << "Please enter in the letter of the</pre>
74
                   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 <<</pre>
                       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
    st @param number The number for the new competitor.
95 ||
```

# 3 Clean build and compilation of Event Creation Program

```
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .clean-
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-
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
      -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
      -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
      -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
     -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/competitor.o.d -o build/
   {\tt Debug/GNU-Linux-x86/competitor.o\ competitor.cpp}
mkdir -p dist/Debug/GNU-Linux-x86
        -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
- 5 Files created by execution of Event Creation Program
- 6 Code for Checkpoint Manager Program

Listing 9: Launcher class.

```
/* File Name: Launcher.java
 1
 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
   /**
    * @author Chris Savill, chs17@aber.ac.uk
13
14
   public class Launcher {
15
16
17
       /**
        * Main method that checks that the right number of arguments were
18
            received
19
         * and calls methods to load the file required and launch the GUI.
20
         * @param args String array of arguments, should be a list of file names
21
22
23
       public static void main(String[] args) throws IOException {
24
            if (args.length < 4) {</pre>
                JOptionPane.showMessageDialog(null, "Invalid number of file
25
                   names supplied required for program to run.\n\"
26
                        + "File names required for:\nFile containing nodes\nFile
                             containing courses\nFile containing entrants\n"
                        + "File to retrieve time records and write time records
27
                           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
36
                    System.out.print("Exiting Program...\n");
                }
37
           }
38
39
       }
40 || }
```

Listing 10: Event class.

<sup>1 /\*</sup> File Name: Manager.java

```
st Description: Event class which stores all members and functions
 2
       pertaining to an event.
 3
    * First Created: 15/03/2013
4
    * Last Modified: 18/03/2013
    */
5
 6
   package Data_Structures;
7
8
   import File_Handling.FileHandler;
9
   import java.io.IOException;
10
   import java.util.ArrayList;
11
   import java.util.Date;
12
13
    * @author Chris Savill, chs17@aber.ac.uk
14
15
   public class Event {
16
17
18
       private ArrayList < Competitor > competitors; //Array list of competitors
           in an event.
19
       private ArrayList < Node > nodes; //Array list of nodes in an event.
20
       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.
21
22
       private ArrayList < Record > records; //Array list of records logged.
23
       private int lastLineRead;
24
       private Date lastRecordedTime;
25
       private boolean timeFileExists;
26
       private String[] fileNames;
27
28
29
         * Method to return array list of competitors.
30
31
         * Oreturn The array list of competitors.
32
33
       public ArrayList < Competitor > getCompetitors() {
34
            return competitors;
35
36
37
       /**
38
         * Method to return array list of nodes.
39
         * Oreturn The array list of nodes.
40
41
         */
42
       public ArrayList < Node > getNodes() {
43
            return nodes;
44
45
        /**
46
47
         * Method to return array list of checkpoints.
48
49
         * @return The array list of checkpoints (non-junction nodes).
         */
50
51
       public ArrayList < Node > getCheckpoints() {
52
           return checkpoints;
53
       }
54
55
56
         * Method to return array list of courses.
57
58
         * @return The array list of courses.
59
         */
60
       public ArrayList <Course > getCourses() {
61
            return courses;
```

```
62
        }
63
64
        /**
65
         * Method to return array list of records.
66
67
         * Oreturn 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
         * Oreturn 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
         * Oreturn The string array of file names.
85
86
         */
87
        public String[] getFileNames() {
88
            return fileNames;
89
90
91
        /**
         * Method to set the last line read number.
92
93
94
         * @param lineNumber The line read from the times file.
95
        public void setLastLineRead(int lineNumber) {
96
97
            this.lastLineRead = lineNumber;
98
99
100
        /**
101
         * Method to set the last time recorded.
102
103
         * Oparam 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
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 {
                         System.out.print("Failed to load competitors. Program
127
                             Exiting.\n");
                     }
128
129
                 } else {
130
                     System.out.print("Failed to load courses. Program Exiting.\n
                 }
131
132
            } else {
                 System.out.print("Failed to load nodes. Program Exiting.\n");
133
134
135
136
            return false;
137
        }
138
139
        /**
140
         * Method that checks if the node number passed in exists in the array
141
         * of nodes loaded in.
142
143
         * Oparam number The number to be compared with.
144
         * Oreturn True if node exists else false.
145
         */
146
        public boolean checkNodeExists(int number) {
            for (int counter = 0; counter < nodes.size(); counter++) {</pre>
147
                 if (number == nodes.get(counter).getNumber()) {
148
                     return true;
149
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
         * Oparam letter The letter to be compared with.
161
         * Oreturn True if course exists else false.
162
         */
163
        public boolean checkCourseExists(char letter) {
164
            for (int counter = 0; counter < courses.size(); counter++) {</pre>
                 if (letter == courses.get(counter).getLetter()) {
165
166
                     return true;
                 } //Course exists.
167
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
        public void setTimesFilesExistsTrue() {
178
179
            timeFileExists = true;
180
181
```

```
182
        /**
183
         * Method to find a competitor and return it.
184
185
          st Oparam competitorNumber The number of the competitor being looked for
186
         * Oreturn The competitor matched.
187
188
        public Competitor retrieveCompetitor(int competitorNumber) {
189
             for (Competitor competitor: competitors) {
                 if (competitor.getNumber() == competitorNumber) {
190
191
                     return competitor;
192
                 }
193
             }
194
             return null;
195
        }
196
197
        /**
198
         * Method to find a course and return it.
199
200
          * Oparam courseLetter The course being looked for.
201
          * Oreturn The course matched.
202
        public Course retrieveCourse(char courseLetter) {
203
204
             for (Course course : courses) {
205
                 if (course.getLetter() == courseLetter) {
206
                     return course;
207
208
             }
209
             return null;
210
        }
211
212
        /**
213
         * Method to retrieve the checkpoint number.
214
215
          * Oparam type The type of the checkpoint.
216
          * @param listIndex The index of the list element.
217
          st Oparam numberOfElements The size of the list.
218
          * @return The checkpoint number being looked for.
219
220
        public int retrieveCheckpointNumber(String type, int listIndex, int
            numberOfElements) {
221
             int[] checkpointArray = new int[numberOfElements];
222
             int arrayIndex = 0;
223
224
             for (int counter = 0; counter < checkpoints.size(); counter++) {</pre>
                 if (checkpoints.get(counter).getType().equals(type)) {
225
226
                     checkpointArray[arrayIndex++] = checkpoints.get(counter).
                         getNumber();
227
                 }
228
            }
229
230
             return checkpointArray[listIndex];
231
        }
232
233
        /**
234
         * Method to check if the new record is valid.
235
236
          * Oparam checkpoint The checkpoint number.
237
          * @param status The status.
238
          st Oparam competitorNumber The competitor's number.
239
          st Oparam time The time of the record.
240
          * @return True is record is valid, else false.
241
```

```
242
        public boolean checkNewRecord(int checkpoint, int status, int
            competitorNumber, Date time) {
243
            Competitor competitor = retrieveCompetitor(competitorNumber);
244
245
            if (timeFileExists != false) {
246
                 if (time.before(lastRecordedTime)) {
247
                     System.out.println("\nInvalid time.");
248
                     return false;
249
                }
            }
250
251
            if (competitor.getStatus() == 'I' || competitor.getStatus() == 'E')
252
253
                System.out.println("\nCompetitor already excluded.");
254
                return false; //Should not be updated as competitor already
                    excluded.
255
            } else if (status == 2 || status == 3) {
256
                 if (competitor.getStatus() != 'A') {
257
                     System.out.println("\nCompetitor hasn't arrived at a medical
                         checkpoint yet.");
258
                     return false; //Competitor cannot be departing or be exclude
                         from a medical checkpoint they haven't arrived at.
259
                } else {
260
                    return true;
                }
261
262
            } else if (status == 0) {
263
                if (competitor.getStatus() != 'A') {
264
                     return true;
265
                } else {
266
                     System.out.println("\nCompetitor is still being examined at
                        a medical checkpoint.");
267
                     return false; //Competitor cannot be at a time checkpoint
                        when should be at a medical checkpoint being examined.
268
269
            } else if (status == 1) {
270
                return true;
271
272
273
            return false;
        }
274
275
276
277
         * Method to determine the final status to be written to the time record
              file.
278
         * Oparam checkpoint The checkpoint number.
279
         * Oparam status The status.
280
         st Oparam competitorNumber The competitor's number.
281
         * @return The final status for the record.
282
         */
283
        public char determineFinalStatus(int checkpoint, int status, int
            competitorNumber) {
284
            Competitor competitor = retrieveCompetitor(competitorNumber);
285
286
            if (competitor.getStatus() == 'N') {
287
                if (checkpoint != competitor.getCheckpoints()[competitor.
                    getCheckpointIndex()]) {
288
                     return 'I';
289
                } else if (status == 0) {
290
                     return 'T';
291
                } else if (status == 1) {
292
                     return 'A';
293
294
            } else if (competitor.getStatus() == 'A') {
```

```
295
                 if (status == 2) {
296
                      return 'D';
297
                 } else if (status == 3) {
298
                      return 'E';
299
                 }
300
             } else if (checkpoint != competitor.getCheckpoints()[competitor.
                 getCheckpointIndex() + 1]) {
301
                  return 'I';
302
             } else {
303
                 if (status == 0) {
304
                      return 'T';
305
                 } else if (status == 1) {
306
                      return 'A';
                 } else if (status == 2) {
307
308
                      return 'D';
309
                 } else if (status == 3) {
310
                      return 'E';
                 }
311
312
             }
313
314
             System.out.print("\n\nInvalid final status, returning 'I'.\n");
315
             return 'I';
316
        }
317
318
         /**
319
          * Constructor to initialise the event.
320
          */
321
         public Event(String[] fileNames) {
             competitors = new ArrayList < Competitor > ();
322
323
             nodes = new ArrayList < Node > ();
324
             checkpoints = new ArrayList < Node > ();
325
             courses = new ArrayList < Course > ();
326
             records = new ArrayList < Record > ();
327
             lastLineRead = 0;
328
             timeFileExists = false;
329
        }
330 || }
```

Listing 11: 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
    * Last Modified: 15/03/2013
 4
 5
6
   package Data_Structures;
7
8
9
    * @author Chris Savill, chs17@aber.ac.uk
10
   public class Node {
11
12
13
       private int number;
14
       private String type;
15
16
        /**
         st Constructor to initialise Node.
17
18
19
         * Oparam number The number of the node.
20
         * Oparam 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
         * Oreturn The node number.
31
32
       public int getNumber() {
33
            return number;
34
35
        /**
36
37
         * Method to return the node's type.
38
         * Oreturn The type of the node.
39
40
       public String getType() {
41
            return type;
42
43
   }
```

Listing 12: Course class.

```
/* 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
   package Data_Structures;
 6
7
8
9
    * @author Chris Savill, chs17@aber.ac.uk
10
11
   public class Course {
12
13
       private char letter;
       private int numberOfNodes;
14
       private int[] nodes;
15
16
17
       /**
18
         * Constructor to initialise course.
19
20
         * Oparam letter The course letter identifier.
21
         st Oparam numberOfNodes The number of nodes the course contains.
22
         * Oparam 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
        /**
         * Method to return the array of nodes the course contains.
45
46
47
        public int[] getNodes() {
48
            return nodes;
49
        }
50 || }
```

Listing 13: 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
    * Last Modified: 18/03/2013
 4
 5
    */
 6
   package Data_Structures;
 7
   import java.util.ArrayList;
 8
9
10
    * @author Chris Savill, chs17@aber.ac.uk
11
12
13
   public class Competitor {
14
       private String name;
15
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
         * Oparam number The competitor's number.
26
         * Oparam course The competitor's course.
27
         * Oparam 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
         * Method to return the competitor's number.
39
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
         * Oreturn 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
         * Oreturn The name of the competitor.
60
         */
61
        public String getName() {
62
            return name;
63
        }
64
65
        /**
         * Method to return the competitor's status.
66
67
         st Oreturn The status of the competitor.
68
69
         */
70
        public char getStatus() {
            return status;
71
        }
72
73
74
75
         * Method to return the index of the last checkpoint the competitor
             arrived
76
         * at.
77
         * Greturn The index of the last checkpoint the competitor arrived at.
78
79
         */
80
        public int getCheckpointIndex() {
81
            return checkpointIndex;
82
        }
83
84
85
         * Method to return the int array of checkpoints.
86
87
         * Oreturn 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
         * nodes).
95
96
         * Oparam event The event instance.
97
98
         * Oreturn 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();</pre>
                counter++) {
105
                 for (int counter2 = 0; counter2 < event.getNodes().size();</pre>
                    counter2++) {
                     if ((!event.getNodes().get(counter2).getType().equals("JN"))
106
```

```
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
             for (int counter = 0; counter < checkpointsList.size(); counter++) {</pre>
116
117
                 intList[counter] = checkpointsList.get(counter).intValue();
118
119
120
             return intList;
121
        }
122
123
124
          st 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 14: Record class.

```
/* File Name: Record.java
 1
 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
    * Last Modified: 17/03/2013
 4
 5
 6
   package Data_Structures;
 7
 8
   import java.util.Date;
9
10
   /**
    * @author Chris Savill, chs17@aber.ac.uk
11
12
   public class Record {
13
14
15
       private Event event;
16
       private char competitorStatus;
       private int checkpoint;
17
       private int competitorNumber;
18
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
         st Oparam competitorNumber The number of the competitor.
```

```
26
         * Oparam 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
         * Oparam checkpoint The number of the checkpoint.
38
39
         * @param competitorNumber The number of the competitor.
40
         * Oparam time The time of the record.
41
         */
42
       public Record(int checkpoint, char status, int competitorNumber, Date
          time) {
43
           this.competitorStatus = status;
            this.checkpoint = checkpoint;
44
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
         * Oreturn 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
         * Oreturn 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
       public int getCompetitorNumber() {
73
74
            return competitorNumber;
75
76
77
78
         * Method to return the time being recorded.
79
80
         * Oreturn The time of the record.
81
         */
82
       public Date getTime() {
83
           return time;
84
       }
85 || }
```

Listing 15: FileHandler class.

```
/* 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;
   import java.io.FileWriter;
16
   import java.io.IOException;
17
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
   public class FileHandler {
30
31
32
33
        * Method to read in all the details for the nodes pertaining to an
34
35
         * Oparam fileName The file name required to access the file needed.
36
         * Oparam event The event instance.
         * Oreturn True if file loaded successfully, else false if it fails at
37
            any
         * point.
38
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
                         if (node.getType().equals("CP") || node.getType().equals
59
                             ("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) {
                 Logger.getLogger(FileHandler.class.getName()).log(Level.SEVERE,
79
                    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
         st Oparam fileName The file name required to access the file needed.
90
         * Oparam event The event instance.
91
         st Oreturn True if file loaded successfully, else false if it fails at
             any
92
         * point.
93
        public boolean readCourses(String fileName, Event event) throws
94
           IOException {
95
            String input;
96
            char courseLetter;
97
            int numberOfNodes;
98
            int[] nodes;
99
            String[] subStrings;
100
            String pattern = "(([A-Za-z]+)((\setminus s+\setminus d+)+)*)"; //Regular expression
                 for courses file.
101
```

```
102
            try {
103
                 BufferedReader reader = new BufferedReader(new FileReader(
                    fileName));
104
105
                 while ((input = reader.readLine()) != null) {
                     if (input.matches(pattern)) { //Checks to make sure the line
106
                          is in the right format.
                         subStrings = input.split("\\s+"); //Gets rid of
107
                             white space\ and\ separates\ the\ strings\ into\ substrings.
108
                         courseLetter = subStrings[0].charAt(0); //Retrieves the
                             course letter.
                         numberOfNodes = Integer.parseInt(subStrings[1]);
109
110
                         nodes = new int[numberOfNodes];
111
112
                         for (int counter = 0; counter < numberOfNodes; counter</pre>
                             ++) {
113
                             if (event.checkNodeExists(Integer.parseInt(
                                 subStrings[counter + 2]))) {
114
                                  nodes[counter] = Integer.parseInt(subStrings[
                                     counter + 2]);
115
                              } else {
                                  System.out.print("Invalid node in course file
116
                                     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.
                         event.getCourses().add(course); //Adds new course to
123
                             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 {
                     System.out.print("Loading in of courses unsuccessful. No
136
                        courses in file.\n\n");
137
                     reader.close();
138
                     return false;
                 }
139
            } catch (FileNotFoundException ex) {
140
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
         * Oparam event The event instance.
         * Oreturn True if file loaded successfully, else false if it fails at
154
             any
155
         * point.
156
        public boolean readCompetitors (String fileName, Event event) throws
157
           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
                while ((input = reader.readLine()) != null) {
168
169
                     if (input.matches(pattern)) { //Checks to make sure the line
                         is in the right format.
                         subStrings = input.split("\\s+"); //Gets rid of
170
                            whitespace and separates the strings into substrings.
                         competitorNumber = Integer.parseInt(subStrings[0]); //
171
                            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
                             System.out.print("Invalid course in competitor file
176
                                 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;</pre>
                                 counter++) {
                                 competitorName += " " + subStrings[counter]; //
185
                                     Concatanates name substrings together.
186
                             }
187
                         }
188
189
                         Competitor competitor = new Competitor(competitorNumber,
                             courseLetter, competitorName, event); /\!/\mathit{Creates} new
                             competitor with parameters read in.
                         event.getCompetitors().add(competitor); //Adds new
190
                            competitor to array list of competitors.
191
                         System.out.print("Invalid line format. Cancelling
192
                            loading of competitors.\n\n");
                         reader.close();
193
194
                         return false;
                     }
195
196
                }
197
```

```
198
                 if (!event.getCompetitors().isEmpty()) {
199
                     System.out.print("Loading in of competitors successful.\n\n"
                        );
200
                     reader.close();
201
                     return true;
202
                 } else {
203
                     System.out.print("Loading in of competitors unsuccessful. No
                          competitors in file.\n\n");
204
                     reader.close();
205
                     return false;
                 }
206
207
            } catch (FileNotFoundException ex) {
208
                 Logger.getLogger(FileHandler.class
209
                         .getName()).log(Level.SEVERE, null, ex);
210
211
212
            System.out.print("Could not open file that contains competitors.\n\n
                ");
213
            return false;
        }
214
215
216
        /**
217
         * Method to read in all the details for the checkpoint times pertaining
218
         * an event.
219
220
         st Oparam fileName The file name required to access the file needed.
         * Oparam event The event instance.
221
222
         st Oreturn True if file loaded successfully, else false if it fails at
             any
         * point.
223
224
         */
225
        public boolean readTimes(String fileName, Event event) throws
           IOException, ParseException {
226
            String input;
227
            int currentLineNumber = 0;
228
            int lastLineNumber = event.getLastLineRead();
229
            char competitorStatus;
230
            int competitorNumber;
            int nodeNumber;
231
232
            String[] subStrings;
233
            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.
234
            SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");
235
            Date time:
236
            \verb| event.getRecords().clear(); // \textit{Empties array list}. \\
237
238
239
            try {
240
                 FileChannel channel = new RandomAccessFile(fileName, "rw").
                    getChannel(); //Creates a channel for the file.
241
                 FileLock lock = channel.lock(); //Blocks/Halts thread until lock
                     aquired.
242
                 BufferedReader reader = new BufferedReader(new FileReader("
243
                    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.
                             nodeNumber = Integer.parseInt(subStrings[1]); //
251
                                 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
         st Oparam fileName The file name required to access the file needed.
291
         * Oparam record The record to be written.
```

```
292
         * @return True if file written to successfully, else false if it fails
         * any point.
293
294
         */
        public boolean appendTimeRecord(String fileName, Record record) {
295
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 || }
```

Listing 16: TypeWindow class.

```
1 \parallel
   /* File Name: TypeWindow.java
    *\ \textit{Description: TypeWindow GUI class using swing.}
 2
 3
    * First Created: 17/03/2013
    * Last Modified: 18/03/2013
 4
 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;
   import javax.swing.JButton;
15
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
    * @author Chris Savill, chs17@aber.ac.uk
23
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;
       private JLabel typeLabel;
31
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
         Oparam event The event instance.
40
41
      public TypeWindow(Event event) {
42
          this.event = event;
43
          medicalSelected = false;
44
45
          //Setup frame:
          typeFrame = new JFrame("Checkpoint Type Selection");
46
          typeFrame.setPreferredSize(new Dimension(300, 200));
47
          typeFrame.setLocation(400, 200);
48
49
          typeFrame.setLayout(new BorderLayout());
50
          typeFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Sets the
              default close operation
          typeFrame.setIconImage(new ImageIcon("horse.jpg").getImage()); //
51
             Loads an image and sets it as the frame icon
52
          53
54
          //Setup panels:
          typePanel = new JPanel(new BorderLayout()); //Creates new JPanel.
55
          typePanel.setBorder(new EmptyBorder(25, 25, 25));
56
             invisible border to simulate a padding effect
57
          typeFrame.add(typePanel, BorderLayout.NORTH); //Adds panel to frame
             and places it in NORTH container.
          bottomPanel = new JPanel();
58
59
          typeFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
             frame and places it in SOUTH container.
60
          61
62
          //Setup checkpoint panel components:
          typeLabel = new JLabel("Select Checkpoint Type Below: ");
63
          typePanel.add(typeLabel, BorderLayout.NORTH);
64
65
66
          time = new JRadioButton("Time Checkpoint");
67
          time.setActionCommand("time");
68
          time.addActionListener(this);
          time.setSelected(true); //Defaults this button to be selected.
69
70
          typePanel.add(time, BorderLayout.CENTER);
          medical = new JRadioButton("Medical Checkpoint");
71
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);
          next.addActionListener(this);
86
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
         * Oparam 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 17: SelectionWindow class.

```
1
   /* File Name: SelectionWindow.java
 2
    * Description: SelectionWindow GUI class using swing.
 3
    * First Created: 16/03/2013
    * Last Modified: 17/03/2013
 4
    */
 5
 6
   package GUI;
 7
   import Data_Structures.Competitor;
8
   import Data_Structures.Event;
9
   import Data_Structures.Node;
10
   import java.awt.BorderLayout;
11
12
   import java.awt.Color;
   import java.awt.Dimension;
13
   import java.awt.event.ActionEvent;
14
   import java.awt.event.ActionListener;
15
16
   import javax.swing.DefaultListModel;
   import javax.swing.ImageIcon;
17
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;
   import javax.swing.ScrollPaneConstants;
26
   import javax.swing.border.EmptyBorder;
   import javax.swing.border.LineBorder;
27
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;
       private int checkpoint;
37
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;
       private JList checkpointList, competitorList;
46
47
       private JScrollPane checkpointListScrollBar, competitorListScrollBar;
48
       private JButton next;
49
50
       /**
51
        st Constructor for SelectionWindow GUI class, sets up and runs GUI.
52
        * Oparam event The event instance.
53
        * Oparam type The type of the checkpoint.
54
        * Oparam 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:
           selectionFrame = new JFrame("Checkpoint and Competitor Selection");
63
           selectionFrame.setLocation(400, 200);
64
65
           selectionFrame.setLayout(new BorderLayout());
66
           selectionFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //
              Sets the default close operation
           selectionFrame.setIconImage(new ImageIcon("horse.jpg").getImage());
67
              //Loads an image and sets it as the frame icon
68
           69
70
           //Setup panels:
           checkpointPanel = new JPanel(new BorderLayout()); //Creates new
71
              JPanel.
72
           checkpointPanel.setBorder(new EmptyBorder(10, 25, 10, 25));
              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.
           bottomPanel = new JPanel();
77
78
           selectionFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
```

```
frame and places it in SOUTH container.
79
           80
81
           //Setup checkpoint panel components:
           checkpointLabel = new JLabel("Select Checkpoint Below: ");
82
           checkpointPanel.add(checkpointLabel, BorderLayout.NORTH);
83
84
85
           checkpointListModel = new DefaultListModel();
           checkpointList = new JList(checkpointListModel);
86
           checkpointList.setBorder(new LineBorder(Color.BLACK));
87
88
           checkpointPanel.add(checkpointList, BorderLayout.CENTER);
89
           checkpointList.addListSelectionListener(this);
90
           checkpointListScrollBar = new JScrollPane(checkpointList);
91
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
        st 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)) {
                     checkpointListModel.addElement(currentCheckpoint.getNumber()
138
                         + ": " + currentCheckpoint.getType());
139
                }
140
            }
141
        }
142
143
144
         st 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()
                         + " Course: " + currentCompetitor.getCourse() + "
151
                            Name: " + currentCompetitor.getName());
152
            }
153
        }
154
155
        /**
156
         * Method to handle actions performed.
157
         * Oparam evt The event triggered.
         */
158
159
        @Override
        public void actionPerformed(ActionEvent evt) {
160
161
            String actionCommand = evt.getActionCommand();
162
163
            if (actionCommand.equals("Next")) {
                 if (checkpointSelected == true && competitorSelected == true) {
164
165
                     selectionFrame.setVisible(false);
                     TimeWindow timeWindow = new TimeWindow (event, checkpoint,
166
                        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
         * Oparam 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)) {
                     checkpoint = event.retrieveCheckpointNumber(type, list.
186
                        getSelectedIndex(), list.getModel().getSize());
187
                     checkpointSelected = true;
188
                } else if (list.equals(competitorList)) {
189
                     competitor = event.getCompetitors().get(list.
```

Listing 18: TimeWindow class.

```
/* File Name: TimeWindow.java
 2
    st 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;
   import java.awt.event.ActionListener;
14
15
   import java.io.IOException;
   import java.text.ParseException;
16
   import java.util.Calendar;
17
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;
   import javax.swing.JPanel;
26
27
   import javax.swing.JSpinner;
   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;
       private JLabel timeLabel;
44
45
       private JButton submit;
       private Date date;
46
       private SpinnerDateModel spinnerModel;
47
48
       private JSpinner spinner;
49
       private JSpinner.DateEditor dateEditor;
50
51
52
         st Constructor for TimeWindow GUI class that sets up and launches the
            GUI.
53
```

```
54
        * Oparam event The event instance.
55
        * Oparam checkpoint The checkpoint number.
56
        * Oparam type The checkpoint type.
57
        * @param competitor The competitor number.
        * Oparam selectionFrame The JFrame this transitioned from.
58
59
        st Oparam typeFrame The JFrame that is reopened after this JFrame closes
60
       public TimeWindow(Event event, int checkpoint, String type, int
61
          competitor, JFrame selectionFrame, JFrame typeFrame) {
62
           selectionFrame.dispose();
63
64
           this.typeFrame = typeFrame;
65
           this.event = event;
           this.checkpoint = checkpoint;
66
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();
           } else {
76
77
               status = 0; //Comeptitor status not a medical related status.
           }
78
79
           timeFrame.setLocation(400, 200);
80
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));
              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();
           timeFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
91
              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:
           submit = new JButton("Submit Checkpoint Record");
107
```

```
108
           submit.setPreferredSize(new Dimension(225, 30));
109
           bottomPanel.add(submit);
110
           submit.addActionListener(this);
           111
112
113
           //Finialise frame setup:
           timeFrame.pack();
114
115
           timeFrame.setVisible(true); //Makes the frame visible
           116
       }
117
118
119
        /**
120
        * Method to handle actions performed.
121
122
         * Oparam 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))
                       JOptionPane.showMessageDialog(timeFrame, "Failed to load
131
                           time records from file.");
132
133
               } catch (IOException | ParseException ex) {
                   Logger.getLogger(TimeWindow.class.getName()).log(Level.
134
                      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
                   fileHandler.appendTimeRecord(event.getFileNames()[3], record
143
144
                   JOptionPane.showMessageDialog(timeFrame, "Time record
                      successfully added.");
145
               } else {
146
                   JOptionPane.showMessageDialog(timeFrame, "Non-valid record.
                      Record will not added.");
               }
147
148
149
               timeFrame.dispose(); //Closes frame and releases resourses.
               this.dispose(); //Releases resources.
150
151
               TypeWindow typeFrame = new TypeWindow(event);
152
153
           }
154
       }
155
156
157
        * Method to get the user to select the status of the competitor at the
158
         * medical checkpoint.
159
160
         * Oreturn The status of the competitor at the medical checkpoint.
161
```

```
162
        public int getMedicalOptions() {
163
            String[] options = new String[]{"Arriving", "Departing", "Excluded"
                };
164
165
            int selection = JOptionPane.showOptionDialog(timeFrame, "Is the
                competitor being marked as 'Arriving',"
166
                     + " 'Departing' or as 'Excluded' on medical grounds?", "
                        Medical Marking", JOptionPane.DEFAULT_OPTION,
167
                     JOptionPane.PLAIN_MESSAGE, null, options, options[0]);
168
169
            if (selection == 0) {
                return 1; //Competitor status to be set to arriving at medical
170
                    checkpoint.
            } else if (selection == 1) {
171
172
                return 2; //Competitor status to be set to departing medical
                    checkpoint.
173
            } else if (selection == 2) {
174
                return 3; //Competitor status to be set to excluded based on
                    medical grounds.
175
            }
176
177
            return 0;
178
        }
179 || }
```

## 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
- 9 Files created by execution of Event Creation Program
- 10 Clean build and compilation of Event Manager Program

```
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .clean-
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-
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
     -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
      -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
      -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
     -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
      -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
       -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.
```

| 3: Display how many competitors are out on the courses.

| 1: Query competitor for current location/status. | 2: Display how many competitors have not started yet.

```
| 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.
| 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
                                                      | Course | Last Recorded Checkpoint | Presumed Location |
                              Name
______
                                                                                           TN - 01
   001 | Ace Abbey
```

| 1   | 003 | Ace Fudge                               | l A | 14                                      | TN - 18                                 |
|-----|-----|---|-----|---|---|
| 1   | 004 | Amber Abbey                             | l C | 13                                      | TC - 13                                 |
| - 1 | 005 | Amber Fudge                             | E   | 13                                      | TN - 13                                 |
| 1   | 006 | April Abbey                             | l D | 09                                      | TN - 02                                 |
| 1   | 007 | April Fudge                             | I В | 14                                      | A - 14                                  |
| 1   | 800 | Ash Abbey                               | F   | 13                                      | TN - 13                                 |
| - 1 | 009 | Ash Fudge                               | l D | 09                                      | TN - 02                                 |
| 1   | 010 | Asti Abbey                              | l A | 07                                      | TN - 15                                 |
|     | 011 | Asti Fudge                              | l A | 07                                      | TN - 15                                 |
|     | 012 | Autumn Abbey                            | l C | 07                                      | TN - 15                                 |
| - 1 | 013 | Autumn Fudge                            | I В | 07                                      | TN - 08                                 |
| 1   | 014 | Barfields Marco Abbey                   | l A | 07                                      | TN - 08                                 |
| 1   | 016 | Barfields Marco Fudge                   | F   | 13                                      | TN - 13                                 |
| 1   | 017 | Basil Abbey                             | I В | 07                                      | TN - 08                                 |
| 1   | 018 | Basil Fudge                             | l A | 05                                      | TN - 07                                 |
| 1   | 019 | Beatrice Abbey                          | l C | 05                                      | TN - 07                                 |
|     | 020 | Beatrice Fudge                          | A   | 05                                      | TN - 05                                 |
| - 1 | 022 | Beau Abbey                              | l D | 05                                      | TN - 05                                 |
|     | 023 | Beau Fudge                              | l C | 05                                      | TN - 05                                 |
| - 1 | 024 | Bella Abbey                             | B   | 04                                      | TN - 04                                 |
| - 1 | 026 | Bella Fudge                             | F   | 09                                      | TN - 12                                 |
|     | 027 | Black Jack Abbey                        | F   | 09                                      | TN - 12                                 |
| - 1 | 028 | Black Jack Fudge                        | A   | 04                                      | TN - 04                                 |
| - 1 | 030 | Blue Abbey                              | B   | 04                                      | TN - 04                                 |
| - 1 | 031 | Blue Fudge                              | B   | 01                                      | TN - 03                                 |
| 1   | 032 | Bobby Abbey                             | l A | 04                                      | TC - 04                                 |
|     | 034 | Bobby Fudge                             | l E | 01                                      | TN - 11                                 |
| 1   | 035 | Bubbles Abbey                           | l C | 01                                      | TN - 02                                 |
| 1   | 036 | Bubbles Fudge                           | l D | 01                                      | TN - 02                                 |
| 1   | 038 | Captain Abbey                           | l A | 01                                      | TN - 02                                 |
| 1   | 039 | Captain Fudge                           | I В | 01                                      | TN - 01                                 |
| 1   | 040 | Chalkie Abbey                           | l D | 01                                      | TN - 01                                 |
| 1   | 041 | Chalkie Fudge                           | F   | 01                                      | TN - 01                                 |
| 1   | 042 | Copper Abbey                            | E   | 01                                      | TN - 01                                 |
| 1   | 044 | Copper Fudge                            | В   | 01                                      | TN - 01                                 |
| 1   | 045 | Diamond Abbey                           | l C | 01                                      | TN - 01                                 |
| 1   | 046 | Diamond Fudge                           | В   | 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.
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.
   | 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.

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

Printing competitors that have not yet started...

| =   | :====== | === |                | === |        | === | =======  | == |
|-----|---------|-----|----------------|-----|--------|-----|----------|----|
| I   | Number  | ١   | Name           | 1   | Course | 1   | Location | 1  |
| =   |         | === |                |     |        |     |          | := |
| - 1 | 095     | - 1 | Ruby Abbey     | -   | F      | 1   | NS       | -  |
| -   | 097     | - 1 | Ruby Fudge     |     | С      | -   | NS       |    |
| -   | 098     | - 1 | Sapphire Abbey | -   | C      | -   | NS       | -  |

| 1   | 100   | Sapphire Fudge | 1 | F | 1     | NS | 1  |
|-----|-------|----------------|---|---|-------|----|----|
| 1   | 101   | Scarlet Abbey  | Ī | С | 1     | NS | 1  |
|     | 102   | Scarlet Fudge  | 1 | F | 1     | NS |    |
|     | 103   | sienna Abbey   | 1 | D | 1     | NS |    |
|     | 106   | sienna Fudge   | 1 | В | 1     | NS |    |
|     | 107   | Silver Abbey   | 1 | F | 1     | NS |    |
|     | 108   | Silver Fudge   | 1 | Α | 1     | NS |    |
|     | 109   | Smokey Abbey   | 1 | Α | 1     | NS |    |
|     | 110   | Smokey Fudge   |   | D | 1     | NS |    |
|     | 111   | Snowy Abbey    |   | E | 1     | NS |    |
|     | 113   | Snowy Fudge    |   | C | 1     | NS |    |
|     | 114   | sonic Abbey    |   | Α | 1     | NS |    |
|     | 115   | sonic Fudge    |   | D | 1     | NS |    |
|     | 117   | Summer Abbey   |   | Α | 1     | NS |    |
|     | 118   | Summer Fudge   |   | E | 1     | NS |    |
|     | 121   | Tango Abbey    |   | В | 1     | NS |    |
|     | 122   | Tango Fudge    | 1 | Α | - 1   | NS |    |
|     | 123   | Topaz Abbey    | 1 | В | - 1   | NS |    |
|     | 124   | Topaz Fudge    | 1 | F | - 1   | NS |    |
|     | 126   | Zizou Abbey    | 1 | D | - 1   | NS |    |
|     | 127   | Zizou Fudge    | 1 | F | 1     | NS |    |
| === | ===== |                |   |   | ===== |    | == |

Key: NS = Not Started.

Number of Competitors not started yet: 24 out of 102

Current Event Time: 11:39.

Press enter to continue.

- | 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.
- $\mid$  4: Display how many competitors have completed their course successfully.
- $\mid$  5: Read in a file of times at which competitors have reached time checkpoints.
- $\mid$  6: Display the result times for the successfully completed.

| = | ==== | ======  |       |             | :   | ===== |      | .====================================== |
|---|------|---------|-------|-------------|-----|-------|------|---|
|   |      |         |       |             |     |       |      |   |
|   | 8:   | Exit pr | ogran | n.          |     |       |      |   |
|   | 7:   | Display | the   | competitors | who | have  | been | excluded.                               |

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

Printing competitors that are out on a course...

| ===== | ==== |                       |          |                          |                   |
|-------|------|-----------------------|----------|--------------------------|-------------------|
| Numb  | er   | Name                  | Course   | Last Recorded Checkpoint | Presumed Location |
| ===== | ==== |                       | =======: |                          |                   |
| 01    |      | Asti Abbey            | l A      | 13                       | TN - 01           |
| 01    | 1    | Asti Fudge            | l A      | 13                       | TN - 01           |
| 01    | 4    | Barfields Marco Abbey | l A      | 13                       | TN - 13           |
| 01    | 8    | Basil Fudge           | l A      | 13                       | TN - 13           |
| 02    | 0    | Beatrice Fudge        | l A      | 13                       | TN - 13           |
| 02    | 8    | Black Jack Fudge      | l A      | 14                       | TN - 20           |
| 03    | 2    | Bobby Abbey           | l A      | 14                       | TN - 17           |
| 03    | 8    | Captain Abbey         | l A      | 17                       | TN - 22           |
| 03    | 9    | Captain Fudge         | l В      | 13                       | TN - 13           |
| 1 04  | 4    | Copper Fudge          | l B      | 14                       | TN - 14           |
| 1 04  | 5    | Diamond Abbey         | l C      | 13                       | TN - 01           |
| 04    | 9    | Ebony Abbey           | l B      | 14                       | TN - 17           |
| 05    | 0    | Ebony Fudge           | l C      | 13                       | TN - 13           |
| 05    | 1    | Ginger Abbey          | l C      | 13                       | TN - 13           |
| 05    | 2    | Ginger Fudge          | l F      | 13                       | TN - 01           |
| 05    | 5    | Goldie Fudge          | l E      | 13                       | TN - 13           |
| 05    | 6    | Honey Abbey           | l F      | 13                       | TN - 01           |
| 05    | 7    | Honey Fudge           | l C      | 07                       | TN - 16           |
| 05    | 8    | Izzy Abbey            | l A      | 07                       | TN - 17           |
| 06    | 0    | Jasmine Abbey         | l A      | 07                       | TN - 15           |
| 06    | 1    | Jasmine Fudge         | l F      | 13                       | TN - 13           |
| 06    | 2    | Lady Abbey            | l D      | l 09                     | TN - 11           |
| 06    | 4    | Lady Fudge            | l В      | 07                       | TN - 08           |
| 06    | 5    | Lady Tara Abbey       | l C      | 07                       | TN - 08           |
| 06    | 6    | Lady Tara Fudge       | l В      | 07                       | TN - 08           |
| I 06  | 7    | Lemon Abbey           | l В      | 07                       | TN - 08           |
| I 06  | 8    | Lemon Fudge           | l E      | l 09                     | TN - 15           |
| I 06  | 9    | Lord Abbey            | l F      | 13                       | TN - 13           |

| I   | 070   | Lord Fudge     |     | Ε     | l | 09                                      |   | TN - 15  | ı   |
|-----|-------|----------------|-----|-------|---|---|---|----------|-----|
| - 1 | 071   | Lucky Abbey    |     | Α     | 1 | 05                                      | 1 | TN - 05  |     |
| 1   | 074   | Lucky Fudge    | 1   | E     | I | 09                                      | 1 | TN - 09  | - 1 |
|     | 076   | Lord Abbey     | 1   | D     | 1 | 05                                      |   | TC - 05  | 1   |
|     | 077   | Lord Fudge     | 1   | В     | 1 | 04                                      |   | TN - 04  | 1   |
|     | 078   | Maddy Abbey    | 1   | F     | 1 | 09                                      |   | TN - 12  | 1   |
| 1   | 079   | Maddy Fudge    | 1   | Α     | I | 04                                      | 1 | TN - 04  | - 1 |
|     | 080   | Magic Abbey    | 1   | D     | 1 | 01                                      |   | TN - 03  | 1   |
|     | 081   | Magic Fudge    | 1   | D     | 1 | 01                                      |   | TN - 03  | 1   |
|     | 083   | Major Abbey    | 1   | Α     | 1 | 01                                      |   | TN - 03  | 1   |
|     | 085   | Major Fudge    | 1   | Α     | 1 | 01                                      |   | TN - 02  | 1   |
|     | 086   | Mattie Abbey   | 1   | В     | 1 | 01                                      | 1 | TN - 02  |     |
|     | 087   | Mattie Fudge   | 1   | Α     | 1 | 01                                      | 1 | TN - 02  | - 1 |
|     | 089   | Prince Abbey   | 1   | В     | 1 | 01                                      | 1 | TN - 01  | - 1 |
|     | 090   | Prince Fudge   | 1   | Α     | 1 | 01                                      | 1 | TN - 01  | - 1 |
| 1   | 091   | Princess Abbey | 1   | В     | I | 01                                      | 1 | TN - 01  | - 1 |
|     | 092   | Princess Fudge | 1   | В     | 1 | 01                                      | 1 | TN - 01  | - 1 |
|     | 093   | Rosie Abbey    | 1   | D     | 1 | 01                                      | 1 | TN - 01  |     |
| - 1 | 094   | Rosie Fudge    | 1   | В     | 1 | 01                                      | 1 | 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.

```
| | 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    | ==  | ====================================== | E      | <br>  CC |
| 003    | - 1 | Ace Fudge                              | l A    | l CC l   |
| 004    | - 1 | Amber Abbey                            | l C    | l CC l   |
| l 005  | - 1 | Amber Fudge                            | l E    | l CC l   |
| l 006  | - 1 | April Abbey                            | l D    | l CC l   |
| 007    | - 1 | April Fudge                            | I В    | l CC l   |
| 800    | - 1 | Ash Abbey                              | l F    | l CC l   |
| l 009  | - 1 | Ash Fudge                              | l D    | l CC l   |
| 012    | - 1 | Autumn Abbey                           | l C    | l CC l   |
| 013    | - 1 | Autumn Fudge                           | I В    | l CC l   |
| 016    | - 1 | Barfields Marco Fudge                  | l F    | l CC l   |
| 017    | -   | Basil Abbey                            | I В    | l CC l   |
| 019    | - 1 | Beatrice Abbey                         | l C    | l CC l   |
| 022    | - 1 | Beau Abbey                             | l D    | l CC l   |
| 024    | -   | Bella Abbey                            | I В    | l CC l   |
| 026    | - 1 | Bella Fudge                            | l F    | l CC l   |
| 027    | - 1 | Black Jack Abbey                       | l F    | l CC l   |
| 030    | - 1 | Blue Abbey                             | I В    | l CC l   |
| 031    | - 1 | Blue Fudge                             | I В    | l CC l   |
| l 034  | - 1 | Bobby Fudge                            | l E    | l CC l   |
| 035    | - [ | Bubbles Abbey                          | l C    | l CC l   |
| 040    | - [ | Chalkie Abbey                          | l D    | l CC l   |
| 042    | - [ | Copper Abbey                           | l E    | l CC l   |
| 047    | - 1 | Dinky Abbey                            | l E    | l CC l   |
| l 048  | I   | Dinky Fudge                            | l F    | I CC I   |

Number of Competitors completed course successfully: 25 out of 102

Current Event Time: 11:39.

Press enter to continue.

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

Printing results...

| ==  | ====== | == |                       | ==  | ======= | == |       | ==  |
|-----|--------|----|-----------------------|-----|---------|----|-------|-----|
| 1   | Number | I  | Name                  | I   | Status  |    | Time  | 1   |
| ==  |        | == |                       | ==: |         | == | ===== | ==  |
|     | 001    |    | Ace Abbey             |     | CC      |    | 02:04 | ı   |
|     | 003    | -  | Ace Fudge             | - 1 | CC      |    | 03:52 | 1   |
|     | 004    | -  | Amber Abbey           | -   | CC      |    | 02:37 | - 1 |
| - 1 | 005    |    | Amber Fudge           |     | CC      |    | 02:11 | - 1 |
| - 1 | 006    |    | April Abbey           |     | CC      |    | 02:03 | - 1 |
| - 1 | 007    |    | April Fudge           |     | CC      |    | 02:46 | - 1 |
| - 1 | 800    |    | Ash Abbey             |     | CC      |    | 01:56 | - 1 |
| - 1 | 009    |    | Ash Fudge             |     | CC      |    | 01:58 | - 1 |
| - 1 | 012    |    | Autumn Abbey          |     | CC      |    | 02:30 | - 1 |
|     | 013    | -  | Autumn Fudge          | - 1 | CC      |    | 02:53 | -   |
|     | 016    | -  | Barfields Marco Fudge | - 1 | CC      |    | 01:55 | -   |
| - 1 | 017    |    | Basil Abbey           |     | CC      |    | 02:49 | - 1 |
| - 1 | 019    |    | Beatrice Abbey        |     | CC      |    | 02:27 | - 1 |
| - 1 | 022    | -  | Beau Abbey            | -   | CC      |    | 02:02 | - 1 |
| - 1 | 024    | -  | Bella Abbey           | -   | CC      |    | 02:54 | - 1 |

| 6: Dis 7: Dis 8: Exi ====== lease s Printin |  | ====================================== | =====<br>=====<br>tatus | =====<br>=====<br>  At | =====<br>=====<br>: Time | === |
|---|--|--|-------------------------|------------------------|--------------------------|-----|
| 6: Dis 7: Dis 8: Exi ====== lease s Printin | select from one of the options above (number): 7   |  |                         | ====                   |                          | ==  |
| 6: Dis<br>7: Dis<br>8: Exi                  | select from one of the options above (number): 7   | ======                                 |                         | ====                   | ====                     | ==  |
| 6: Dis<br>7: Dis<br>8: Exi                  | it program.  |  | -====                   | ====                   | ====                     | ==  |
| 6: Dis<br>7: Dis                            | it program.  | ======                                 |                         | ====                   | ====                     | ==  |
| 6: Dis<br>7: Dis                            |  |  |                         |                        |                          |     |
| 6: Dis                                      | splay the competitors who have been excluded.  |  |                         |                        |                          |     |
|   | splay the result times for the successfully comple   | tea.                                   |                         |                        |                          |     |
|   | ad in a file of times at which competitors have re   | ached time                             |                         | •                      | nts.                     |     |
|   | splay how many competitors are out on the courses.<br>splay how many competitors have completed their co | urse succ                              | essful                  | ly.                    |                          |     |
| 2: Dis                                      | splay how many competitors have not started yet.   |  |                         |                        |                          |     |
| 1: Que                                      | ery competitor for current location/status.  |  |                         |                        |                          |     |
| =====                                       | ====== MAIN MENU =======   | =======                                |                         | =====                  |                          | ==  |
|   |  |  |                         |                        |                          |     |
| ress en                                     | nter to continue.  |  |                         |                        |                          |     |
| ırrent                                      | Event Time: 11:39.   |  |                         |                        |                          |     |
| ımber c                                     | of Competitors completed course successfully: 25 o   | ut of 102                              |                         |                        |                          |     |
| =====                                       | ,  | =======                                |                         | =====                  |                          | ==  |
| 04 <i>7</i><br>048                          | Dinky Abbey<br>  Dinky Fudge   | <br>                                   | CC                      | •                      | )2:10<br>)1:54           |     |
| 042<br>047                                  | Copper Abbey   | <br>                                   | CC                      | •                      | )2:05<br>)2:10           |     |
| 040   | Chalkie Abbey  | i                                      | CC                      |                        | 2:02                     | İ   |
| 035   | Bobby Fudge<br>  Bubbles Abbey   | <br>                                   | CC                      | •                      | )2:03<br>)2:32           |     |
|   | Blue Fudge   | 1                                      | CC                      | •                      | 2:44                     |     |
| 031<br>034                                  |  | 1                                      | CC                      | •                      | )1:49<br>)2:43           |     |
| 034   | Black Jack Abbey<br>  Blue Abbey   | ı                                      |                         |                        |                          |     |

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

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.

```
| 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
- 13 Log file contents