CS22510 Assignment 1
Runners and Riders
”Out and About”

Chris Savill
chs17@aber.ac.uk

March 20, 2013

Contents

1 Description of three programs

1.1 Event Creation Program

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.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: creator.h
 * Description: Header file for the starter function declarations.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#ifndef CREATOR_H
#define CREATOR_H

#include <memory>
#include "event.h"

bool get_acceptance(); //Function to get the user's input for accepting or
    rejecting their inputs.
bool checkCourseExists(char letter, Event *event); //Member function that
    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
    program menu.

#endif  /* CREATOR_H */
```

Listing 2: Main method and menu file.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: competitor.cpp
 * Description: cpp file that contains function definitions for the start-up
     of the event creation program.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#include "creator.h"
#include <iostream>
#include <cstdlib>
#include <limits>

using namespace std;

/* Main function that just calls a function that takes over. */
int main(int argc, char** argv) {
    Event *event = new Event();
    ecp_menu(event);

    return 0;
}
```

```cpp

/* Function to get the user's input for accepting or rejecting their inputs. */
bool get_acceptance() {
    char option;

    do {
        cout << "If yes press 'y' then 'Enter'" << endl << "If no press 'n' then 'Enter'" << endl;
        cin.clear();
        option = cin.get();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        if (option == 'y') return true;
        else if (option == 'n') return false;
        else cout << "Invalid option selected" << endl;
    } while (option != 'y' && option != 'n');
}

/* Function that displays the main menu for the event creation program. */
void ecp_menu(Event *event) {
    int option; //Field to store the user's option input.

    do {
        cout << "*********************************************************" << endl;
        cout << "*  Runners and Riders Event Creation Program Main Menu  *" << endl;
        cout << "*********************************************************" << endl;
        cout << "*              1. Add Competitor to Event               *" << endl;
        cout << "*              2. Add Course to Event                   *" << endl;
        cout << "*              3. Export Event to File                  *" << endl;
        cout << "*              4. Export Competitors to File            *" << endl;
        cout << "*              5. Export Courses to File                *" << endl;
        cout << "*              6. Exit Event Creation Program           *" << endl;
        cout << "*********************************************************" << endl << endl;

        cout << "Please enter in an option from the above an press 'Enter': ";
        cin.clear();
        cin >> option;
        cin.ignore();

        switch (option) {
            case 1:
                event->add_competitor();
                break;
            case 2:
                event->add_course();
                break;
            case 3:
                event->export_event();
                break;
            case 4:
                event->export_competitors();
                break;
            case 5:
                event->export_courses();
                break;
            case 6:
                delete(event);
                cout << "Exiting program..." << endl << endl;
                break;
            default:
                cout << "Please enter in a valid option." << endl << endl;
        }
    } while (option != 6);
}
```

Listing 3: Header file Event class.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: event.h
 * Description: Header file for the Event class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#ifndef EVENT_H
#define EVENT_H

#include <memory>
#include "competitor.h"
#include "course.h"
#include <vector>
#include <cstdlib>
#include <iostream>

#define MAX_EVENT_NAME_LENGTH 79
#define MAX_DATE_LENGTH 19

class Competitor;
class Course;

class Event {
private:
    std::string name; //Name of the event.
    std::string date; //Date of the event.
    std::string start_time; //Start time of the event.
    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
       event.

    void set_name(); //Member function to get the user to input the events
       name.
    void set_date(); //Member function to get the user to input the date of
       the event.
    void set_start_time(); //Member function to get the user to input the
       start time of the event.

public:
    Event();
    ~Event();
    std::vector<Course*>* getCourses(); //Member function that returns a
       pointer to the vector of courses.
    void add_competitor(); //Member function that will handle adding a
       competitor to the event.
    void add_course(); //Member function that will handle adding a course to
        the event.
    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
       exporting of the array of competitors to a '.txt' file.
    void export_courses(); //Member function that will handle the exporting
       of the array of courses to a '.txt' file.
};

#endif  /* EVENT_H */
```

Listing 4: Cpp file for Event class.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: event.cpp
 * Description: cpp file that contains member function definitions for the
    event class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

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

using namespace std;

/* Member function that returns a pointer to the vector of courses. */
vector<Course*>* Event::getCourses() {
    return courses;
}

/* Member function to get the user to input the events name. */
void Event::set_name() {
    bool name_chosen = false;
    string name;

    do {
        do {
            cout << "Please enter in the name for the event (no more than 79
                characters): ";
            cin.clear();
            getline(cin, name);
        } while (name.length() > MAX_EVENT_NAME_LENGTH);

        cout << endl << endl << "Are you happy with the name: '" << name <<
            "'?" << endl;
        name_chosen = get_acceptance();
    } while (name_chosen == false);

    this->name = name;
}

/* Member function to get the user to input the date of the event. */
void Event::set_date() {
```

```cpp
    bool date_chosen = false;
    string date;

    do {
        do {
            cout << endl << endl << "Please enter in the date for the event (no more than 19 characters): ";
            cin.clear();
            getline(cin, date);
        } while (date.length() > MAX_DATE_LENGTH);

        cout << endl << endl << "Are you happy with the date: '" << date << "'?" << endl;
        date_chosen = get_acceptance();
    } while (date_chosen == false);

    this->date = date;
}

/* Member function to get the user to input the start time of the event. */
void Event::set_start_time() {
    bool start_time_chosen = false;
    bool valid_hours = false;
    bool valid_minutes = false;
    char input[3];
    int hours;
    int minutes;
    string start_time;
    string string_hours;
    string string_minutes;

    do {
        do {
            cout << endl << endl << "Please enter in the start time for the event with the 24 hour format 'HH:MM', hours first: ";
            cin.clear();
            cin >> input;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << endl;

            if (isdigit(input[0]) && isdigit(input[1])) { //Ensures the input has 2 digits.
                hours = atoi(input); //Converts the digits into an int and stores it in hours.

                if (hours <= 23 && hours >= 00) { //Makes sure that the hours are in 24-hour format.
                    cout << "Valid hours entered." << endl << endl;
                    valid_hours = true;
                }
            } else cout << "Invalid hours entered, please enter in a value between 00 and 23 inclusive." << endl << endl;
        } while (valid_hours == false);

        do {
            cout << endl << endl << "Please now enter in the minutes: ";
            cin.clear();
            cin >> input;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << endl;

            if (isdigit(input[0]) && isdigit(input[1])) {
                minutes = atoi(input);

                if (minutes <= 59 && minutes >= 00) { //Makes sure minutes are valid.
                    cout << "Valid minutes entered." << endl << endl;
                    valid_minutes = true;
                }
            } else cout << "Invalid minutes entered, please enter in a value between 00 and 59 inclusive." << endl << endl;
        } while (valid_minutes == false);

        cout << endl << endl << "Are you happy with the start time: '" << hours << ":" << minutes << "'?" << endl;
        start_time_chosen = get_acceptance();
    } while (start_time_chosen == false);

    ostringstream string_retriever; //Converts ints into strings.
    string_retriever << hours;
    string_hours = string_retriever.str();
    string_retriever.str(""); //Clears the string stream.
    string_retriever << minutes;
    string_minutes = string_retriever.str();

    start_time = string_hours + ":" + string_minutes; //Concatenates the final time into HH:MM format.
    this->start_time = start_time;
}

/* Member function that will handle adding a competitor to the event.
 * @param number The current competitor number.
 */
void Event::add_competitor() {
    if (courses->empty()) cout << "No courses exist for competitor course selection. Please create a course first." << endl << endl;
    else {
        Competitor *competitor = new Competitor((competitors->size() + 1), this);
        competitors->push_back(competitor);
        cout << "New competitor added to event." << endl << endl;
        cout << "Competitor number: " << competitors->back()->get_number();
        cout << "Competitor name: " << competitors->back()->get_name() << endl;
        cout << "Course: " << competitors->back()->get_course() << endl;
    }
}

/* Member function that will handle adding a course to the event. */
void Event::add_course() {
    Course *course = new Course(this);
    courses->push_back(course);
    cout << "New course added to event." << endl << endl;
    cout << "Course letter: " << courses->back()->get_letter() << endl;
    cout << "Number of course nodes: " << courses->back()->get_number_of_nodes() << endl;
    cout << "Nodes: " << courses->back()->get_node(0);

    for (int counter = 1; counter < courses->back()->get_number_of_nodes(); counter++) {
        cout << ", " << courses->back()->get_node(counter);
    }

    cout << endl << endl;
}

```

```cpp
/* Member function that will handle exporting the name, date and start_time of the event to a '.txt' file. */
void Event::export_event() {
    ofstream competitors_file;
    competitors_file.open("name.txt", ios::out);

    if (competitors_file.is_open()) {
        competitors_file << this->name << "\n" << this->date << "\n" << this->start_time;
        competitors_file.close();
        cout << "Event successfully exported to 'name.txt'." << endl << endl;
    } else cout << "File 'name.txt' could not be written." << endl;
}

/* Member function that will handle the exporting of the array of competitors to a '.txt' file. */
void Event::export_competitors() {
    if (competitors->empty()) cout << "No competitors to export. Exporting cancelled." << endl << endl;
    else {
        ofstream competitors_file;
        competitors_file.open("entrants.txt", ios::out);

        if (competitors_file.is_open()) {
            for (int counter = 0; counter < this->competitors->size(); counter++) {
                competitors_file << this->competitors->at(counter)->get_number() << " " << this->competitors->at(counter)->get_course()
                        << " " << this->competitors->at(counter)->get_name() << "\n";
            }

            competitors_file.close();
            cout << "Competitors successfully exported to 'entrants.txt'." << endl << endl;
        } else cout << "File 'entrants.txt' could not be written." << endl;
    }
}

/* Member function that will handle the exporting of the array of courses to a '.txt' file. */
void Event::export_courses() {
    if (courses->empty()) cout << "No courses to export. Exporting cancelled." << endl << endl;
    else {
        ofstream courses_file;
        courses_file.open("courses.txt", ios::out);

        if (courses_file.is_open()) {
            for (int counter = 0; counter < this->courses->size(); counter++) {
                courses_file << this->courses->at(counter)->get_letter() << " " << this->courses->at(counter)->get_number_of_nodes();

                for (int counter2 = 0; counter2 < this->courses->at(counter)->get_number_of_nodes(); counter2++) {
                    courses_file << " " << this->courses->at(counter)->get_node(counter2);
                }
                courses_file << "\n";
            }

            courses_file.close();
            cout << "Courses successfully exported to 'courses.txt'." <<
                endl << endl;
        } else cout << "File 'courses.txt' could not be written." << endl;
    }
}

/* Constructor for Event class. */
Event::Event() {
    competitors = new vector<Competitor* > ();
    courses = new vector<Course* > ();
    set_name();
    set_date();
    set_start_time();

    cout << "Event name: " << this->name << endl;
    cout << "Event date: " << this->date << endl;
    cout << "Event start time: " << this->start_time << endl << endl;
}

/* Destructor for Event class. */
Event::~Event() {
    delete(competitors);
    delete(courses);
}
```

Listing 5: Header file for Course class.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: course.h
 * Description: Header file for the Course class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#ifndef COURSE_H
#define COURSE_H

#include <memory>
#include <vector>

class Event;

class Course {
private:
    char letter; //The courses unique identification letter for an event.
    int number_of_nodes; //The number of nodes the course contains.
    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
        available to select from, read in from the 'nodes.txt' file.

    void set_letter(Event *event); //Member function that will set the
        letter of the course.
    void set_number_of_nodes(); //Member function that will set the number
        of nodes of the course.
    bool read_nodes_available(); //Member function that reads in the nodes
        from the 'nodes.txt' file and adds them to the nodes available array.
    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
         new node being selected matches the last node added.
    bool check_node_exists(int number); //Member function that checks that
       the node being added exists in the array of nodes available.

public:
    char get_letter(); //Member function to return a course's letter.
    int get_number_of_nodes(); //Member function to return a course's number
        of nodes.
    int get_node(int index); //Member function to return a node from the
       course's vector of nodes.
    Course(Event *event);
    ~Course();
};

#endif  /* COURSE_H */
```

Listing 6: Cpp file for Course class.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: course.cpp
 * Description: cpp file that contains member function definitions for the
     course class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

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

using namespace std;

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

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

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

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

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

/* Member function that will set the letter of the course. */
```

```cpp
void Course::set_letter(Event *event) {
    bool valid_letter = false;
    bool letter_chosen = false;
    char letter;

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

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

        cout << endl << "Are you happy with the course letter: '" << letter << "'?" << endl;
        letter_chosen = get_acceptance();
    } while (letter_chosen == false);

    this->letter = letter;
}

/* Member function that will set the number of nodes of the course. */
void Course::set_number_of_nodes() {
    bool number_chosen = false;
    int number;

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

        cout << endl << endl << "Are you happy with the number of nodes: '" << number << "'?" << endl;
        number_chosen = get_acceptance();
    } while (number_chosen == false && number > 0);

    this->number_of_nodes = number;
}

/* Member function that reads in the nodes from the 'nodes.txt' file and adds them to the nodes available array. */
bool Course::read_nodes_available() {
    ifstream nodes_file;
    string input;
    int node_number;

    nodes_file.open("nodes.txt", ios::in);

    if (nodes_file.is_open()) {
        while (getline(nodes_file, input)) { //Keep reading until EOF
reached.
            stringstream int_retriever(input); //Retrieves int from the string stream.
            int_retriever >> node_number; //Stores the int in node_number.
            this->nodes_available->push_back(node_number);
        }

        nodes_file.close();
        cout << "Nodes from 'nodes.txt' read in successfully." << endl;
        cout << "Nodes read in: " << nodes_available->at(0);
        for (int counter = 1; counter < nodes_available->size(); counter++) cout << ", " << nodes_available->at(counter);
        cout << endl << endl;
    } else cout << "File 'nodes.txt' could not be opened. Please check file is in correct directory and permissions." << endl;
}

/* Member function that adds a new node to the course. */
void Course::add_node() {
    bool number_chosen = false;
    string input;
    int number = 0;

    do {
        do {
            cout << "Please enter in the node number you wish to add to the course: ";
            getline(cin, input);
            stringstream int_retriever(input);
            int_retriever >> number;
        } while (duplicated_last_node(number) || !check_node_exists(number)); //Makes sure that the number entered doesn't match the last number entered and that it does exist.

        cout << endl << endl << "Are you happy with the node number: '" << number << "'?" << endl;
        number_chosen = get_acceptance();
    } while (number_chosen == false);

    this->nodes->push_back(number);
}

/* Member function to check if the new node being selected matches the last node added. */
bool Course::duplicated_last_node(int number) {
    if (!nodes->empty()) { //Only checks if there are nodes present.
        if (number == nodes->back()) {
            cout << "Node matches last node. Please choose a different node number to add." << endl;
            return true;
        }
    }

    return false; //Returns false if the number entered and the last number entered don't match.
}

/* Member function that checks that the node being added exists in the array of node available. */
bool Course::check_node_exists(int number) {
    for (int counter = 0; counter < this->nodes_available->size(); counter++) {
        if (number == this->nodes_available->at(counter)) return true;
    }

    cout << "Node does not exist, please choose a different node number to
        add." << endl;
    return false; //Returns false if the number entered does not exist in
        the vector of nodes available.
}

/* Constructor for Course class. */
Course::Course(Event *event) {
    this->nodes = new vector<int>();
    this->nodes_available = new vector<int>();

    if (read_nodes_available()) {
        set_letter(event);
        set_number_of_nodes();

        for (int counter = 0; counter < number_of_nodes - 1; counter++) {
            add_node();
        }

        nodes->push_back(nodes->front()); //Adds the last node, matching the
            first node to the course.
    } else cout << "Nodes could not be read in from 'nodes.txt' file. Course
        creation cancelled." << endl << endl;
}

/* Destructor for Course class. */
Course::~Course() {
    delete(nodes);
    delete(nodes_available);
}
```

Listing 7: Header file for Competitor class.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: competitor.h
 * Description: Header file for the Competitor class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#ifndef COMPETITOR_H
#define COMPETITOR_H

#include <memory>
#include <string>

#define MAX_COMPETITOR_NAME_LENGTH 51 //Includes null terminator \0.

class Event;

class Competitor {
private:
    int number; //The competitor's unique identification number for an event
        .
    std::string name; //The competitor's name.
    char course; //The course letter the competitor is entering in for.

    void set_number(int number); //Member function that will set the number
        of the competitor.
    void set_name(); //Member function that will set the name of the
        competitor.
    void set_course(Event *event); //Member function that will set the
        course letter for the competitor.

public:
    int get_number(); //Member function to return a competitor's number.
    std::string get_name(); //Member function to return a competitor's name.
    char get_course(); //Member function to return a competitor's course.
    Competitor(int number, Event *event);
};

#endif  /* COMPETITOR_H */
```

Listing 8: Cpp file for Competitor class.

```
/*
 * Author: Chris Savill, chs17@aber.ac.uk
 * File Name: competitor.cpp
 * Description: cpp file that contains member function definitions for the
    competitor class.
 * First Created: 11/03/2013
 * Last Modified: 14/03/2013
 */

#include "competitor.h"
#include "creator.h"
#include <ctype.h>
#include <iostream>
#include <limits>

using namespace std;

/* Member function to return a competitor's number. */
int Competitor::get_number() {
    return this->number;
}

/* Member function to return a competitor's name. */
string Competitor::get_name() {
    return this->name;
}

/* Member function to return a competitor's course. */
char Competitor::get_course() {
    return this->course;
}

/* Member function that will set the number of the competitor.
 * @param number The number for the competitor.
 */
void Competitor::set_number(int number) {
    this->number = number;
}

/* Member function that will set the name of the competitor. */
void Competitor::set_name() {
    bool name_chosen = false;
    string name;

    do {
        do {
            cout << endl << endl << "Please enter in the name for the competitor (no more than 50 characters): ";
            getline(cin, name);
        } while (name.length() > MAX_COMPETITOR_NAME_LENGTH);

        cout << endl << endl << "Are you happy with the name: '" << name << "'?" << endl;

        name_chosen = get_acceptance();

    } while (name_chosen == false);

    this->name = name;
}

/* Member function that will set the course letter for the competitor. */
void Competitor::set_course(Event *event) {
    bool valid_letter = false;
    bool letter_chosen = false;
    char letter;

    do {
        do {
            cout << endl << endl << "List of courses available for the competitor to enter on: " << event->getCourses()->front()->get_letter();

            if (event->getCourses()->size() > 1) { //Only prints out other courses if the size of the vector > 1.
                 for (int counter = 1; counter < event->getCourses()->size(); counter++)
                     cout << ", " << event->getCourses()->at(counter)->get_letter();
            }

            cout << endl << endl << "Please enter in the letter of the course that the competitor is entering: ";
            cin.clear(); //Resets the input stream flags.
            letter = cin.get(); //Gets a single character.
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); //Clears the input stream.

            if (isalpha(letter) && checkCourseExists(letter, event))
                valid_letter = true; //Makes sure character is a letter and that it corresponds to a course that exists.
            else {
                 cout << "Please enter in a valid course letter." << endl << endl;
                 valid_letter = false;
            }
        } while (valid_letter == false);

        cout << endl << "Are you happy with the course letter: '" << letter << "'?" << endl;
        letter_chosen = get_acceptance();
    } while (letter_chosen == false);

    this->course = letter;
}

/* Constructor for Competitor class.
 * @param number The number for the new competitor.
 */
```

```cpp
Competitor::Competitor(int number, Event *event) {
    set_number(number);
    cout << "Competitor number: " << this->number << endl;
    set_name();
    cout << "Competitor name: " << this->name << endl;
    set_course(event);
    cout << "Competitor course:" << this-> course << endl;
}
```

3 Clean build and compilation of Event Creation Program

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


CLEAN SUCCESSFUL (total time: 217ms)

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


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

4 Run through of Event Creation Program

5 Files created by execution of Event Creation Program

6 Code for Checkpoint Manager Program

Listing 9: Launcher class.

```java
/* File Name: Launcher.java
 * Description: Launcher class which handles the initial launching of the
    Checkpoint Manager Program.
 * First Created: 15/03/2013
 * Last Modified: 19/03/2013
 */
package Data_Structures;

import GUI.TypeWindow;
import java.io.IOException;
import javax.swing.JOptionPane;

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

    /**
     * Main method that checks that the right number of arguments were
        received
     * and calls methods to load the file required and launch the GUI.
     *
     * @param args String array of arguments, should be a list of file names
        .
     */
    public static void main(String[] args) throws IOException {
        if (args.length < 4) {
            JOptionPane.showMessageDialog(null, "Invalid number of file
                names supplied required for program to run.\n\n"
                    + "File names required for:\nFile containing nodes\nFile
                        containing courses\nFile containing entrants\n"
                    + "File to retrieve time records and write time records
                        to.\n\n"
                    + "Now exiting program.");
        } else {
            Event event = new Event(args);

            if (event.loadCycle(args)) {
                JOptionPane.showMessageDialog(null, "Data files loaded
                    successfully.");
                TypeWindow typeWindow = new TypeWindow(event);
            } else {
                System.out.print("Exiting Program...\n");
            }
        }
    }
}
```

Listing 10: Event class.

```java
/* File Name: Manager.java
```

```
 * Description: Event class which stores all members and functions
     pertaining to an event.
 * First Created: 15/03/2013
 * Last Modified: 18/03/2013
 */
package Data_Structures;

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

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

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

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

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

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

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

```java
    }

    /**
     * Method to return array list of records.
     *
     * @return The array list of records.
     */
    public ArrayList<Record> getRecords() {
        return records;
    }

    /**
     * Method to get the last line read number.
     *
     * @return The line read from the times file.
     */
    public int getLastLineRead() {
        return lastLineRead;
    }

    /**
     * Method to return the array of file names.
     *
     * @return The string array of file names.
     */
    public String[] getFileNames() {
        return fileNames;
    }

    /**
     * Method to set the last line read number.
     *
     * @param lineNumber The line read from the times file.
     */
    public void setLastLineRead(int lineNumber) {
        this.lastLineRead = lineNumber;
    }

    /**
     * Method to set the last time recorded.
     *
     * @param time The last time recorded.
     */
    public void setLastRecordedTime(Date time) {
        this.lastRecordedTime = time;
    }

    /**
     * Method to call a series of methods to load in the data required by
          the
     * program.
     *
     * @param args The list of filenames to load the required data into the
     * system.
     * @return Successful/Unsuccessful.
     */
    public boolean loadCycle(String[] fileNames) throws IOException {
        this.fileNames = fileNames;

        FileHandler fileReader = new FileHandler();

        if (fileReader.readNodes(fileNames[0], this)) {
            if (fileReader.readCourses(fileNames[1], this)) {
                if (fileReader.readCompetitors(fileNames[2], this)) {
                    return true;
                } else {
                    System.out.print("Failed to load competitors. Program
                        Exiting.\n");
                }
            } else {
                System.out.print("Failed to load courses. Program Exiting.\n
                    ");
            }
        } else {
            System.out.print("Failed to load nodes. Program Exiting.\n");
        }

        return false;
    }

    /**
     * Method that checks if the node number passed in exists in the array
        list
     * of nodes loaded in.
     *
     * @param number The number to be compared with.
     * @return True if node exists else false.
     */
    public boolean checkNodeExists(int number) {
        for (int counter = 0; counter < nodes.size(); counter++) {
            if (number == nodes.get(counter).getNumber()) {
                return true;
            } //Nodes exists.
        }

        return false; //Returns false if the node number passed in does not
            exist in the array list of nodes.
    }

    /**
     * Method that checks if the course letter passed in exists in the array
     * list of courses loaded in.
     *
     * @param letter The letter to be compared with.
     * @return True if course exists else false.
     */
    public boolean checkCourseExists(char letter) {
        for (int counter = 0; counter < courses.size(); counter++) {
            if (letter == courses.get(counter).getLetter()) {
                return true;
            } //Course exists.
        }

        return false; //Returns false if the course letter passed in does
            not exist in the array list of courses.
    }

    /**
     * Method to let the know event instance know that a time file does now
     * exist.
     *
     */
    public void setTimesFilesExistsTrue() {
        timeFileExists = true;
    }

    /**
     * Method to find a competitor and return it.
     *
     * @param competitorNumber The number of the competitor being looked for
       .
     * @return The competitor matched.
     */
    public Competitor retrieveCompetitor(int competitorNumber) {
        for (Competitor competitor : competitors) {
            if (competitor.getNumber() == competitorNumber) {
                return competitor;
            }
        }
        return null;
    }

    /**
     * Method to find a course and return it.
     *
     * @param courseLetter The course being looked for.
     * @return The course matched.
     */
    public Course retrieveCourse(char courseLetter) {
        for (Course course : courses) {
            if (course.getLetter() == courseLetter) {
                return course;
            }
        }
        return null;
    }

    /**
     * Method to retrieve the checkpoint number.
     *
     * @param type The type of the checkpoint.
     * @param listIndex The index of the list element.
     * @param numberOfElements The size of the list.
     * @return The checkpoint number being looked for.
     */
    public int retrieveCheckpointNumber(String type, int listIndex, int
       numberOfElements) {
        int[] checkpointArray = new int[numberOfElements];
        int arrayIndex = 0;

        for (int counter = 0; counter < checkpoints.size(); counter++) {
            if (checkpoints.get(counter).getType().equals(type)) {
                checkpointArray[arrayIndex++] = checkpoints.get(counter).
                   getNumber();
            }
        }

        return checkpointArray[listIndex];
    }

    /**
     * Method to check if the new record is valid.
     *
     * @param checkpoint The checkpoint number.
     * @param status The status.
     * @param competitorNumber The competitor's number.
     * @param time The time of the record.
     * @return True is record is valid, else false.
     */
    public boolean checkNewRecord(int checkpoint, int status, int competitorNumber, Date time) {
        Competitor competitor = retrieveCompetitor(competitorNumber);

        if (timeFileExists != false) {
            if (time.before(lastRecordedTime)) {
                System.out.println("\nInvalid time.");
                return false;
            }
        }

        if (competitor.getStatus() == 'I' || competitor.getStatus() == 'E') {
            System.out.println("\nCompetitor already excluded.");
            return false; //Should not be updated as competitor already excluded.
        } else if (status == 2 || status == 3) {
            if (competitor.getStatus() != 'A') {
                System.out.println("\nCompetitor hasn't arrived at a medical checkpoint yet.");
                return false; //Competitor cannot be departing or be exclude from a medical checkpoint they haven't arrived at.
            } else {
                return true;
            }
        } else if (status == 0) {
            if (competitor.getStatus() != 'A') {
                return true;
            } else {
                System.out.println("\nCompetitor is still being examined at a medical checkpoint.");
                return false; //Competitor cannot be at a time checkpoint when should be at a medical checkpoint being examined.
            }
        } else if (status == 1) {
            return true;
        }

        return false;
    }

    /**
     * Method to determine the final status to be written to the time record file.
     * @param checkpoint The checkpoint number.
     * @param status The status.
     * @param competitorNumber The competitor's number.
     * @return The final status for the record.
     */
    public char determineFinalStatus(int checkpoint, int status, int competitorNumber) {
        Competitor competitor = retrieveCompetitor(competitorNumber);

        if (competitor.getStatus() == 'N') {
            if (checkpoint != competitor.getCheckpoints()[competitor.getCheckpointIndex()]) {
                return 'I';
            } else if (status == 0) {
                return 'T';
            } else if (status == 1) {
                return 'A';
            }
        } else if (competitor.getStatus() == 'A') {
            if (status == 2) {
                return 'D';
            } else if (status == 3) {
                return 'E';
            }
        } else if (checkpoint != competitor.getCheckpoints()[competitor.
            getCheckpointIndex() + 1]) {
            return 'I';
        } else {
            if (status == 0) {
                return 'T';
            } else if (status == 1) {
                return 'A';
            } else if (status == 2) {
                return 'D';
            } else if (status == 3) {
                return 'E';
            }
        }

        System.out.print("\n\nInvalid final status, returning 'I'.\n");
        return 'I';
    }

    /**
     * Constructor to initialise the event.
     */
    public Event(String[] fileNames) {
        competitors = new ArrayList<Competitor>();
        nodes = new ArrayList<Node>();
        checkpoints = new ArrayList<Node>();
        courses = new ArrayList<Course>();
        records = new ArrayList<Record>();
        lastLineRead = 0;
        timeFileExists = false;
    }
}
```

Listing 11: Node class.

```java
/* File Name: Node.java
 * Description: Node class which stores all members and functions pertaining
     to a node.
 * First Created: 15/03/2013
 * Last Modified: 15/03/2013
 */
package Data_Structures;

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

    private int number;
    private String type;

    /**
     * Constructor to initialise Node.
     *
     * @param number The number of the node.
     * @param type The type of the node.
     */
    public Node(int number, String type) {
        this.number = number;
        this.type = type;
    }

    /**
     * Method to return the node's number.
     *
     * @return The node number.
     */
    public int getNumber() {
        return number;
    }

    /**
     * Method to return the node's type.
     * @return The type of the node.
     */
    public String getType() {
        return type;
    }
}
```

Listing 12: Course class.

```java
/* File Name: Couse.java
 * Description: Course class which stores all members and functions
   pertaining to a course.
 * First Created: 15/03/2013
 * Last Modified: 17/03/2013
 */
package Data_Structures;

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

    private char letter;
    private int numberOfNodes;
    private int[] nodes;

    /**
     * Constructor to initialise course.
     *
     * @param letter The course letter identifier.
     * @param numberOfNodes The number of nodes the course contains.
     * @param nodes The array of nodes the course contains.
     */
    public Course(char letter, int numberOfNodes, int[] nodes) {
        this.letter = letter;
        this.numberOfNodes = numberOfNodes;
        this.nodes = nodes;
    }

    /**
     * Method to return the course letter.
     */
    public char getLetter() {
        return letter;
    }

    /**
     * Method to return the number of nodes the course contains.
     */
    public int getNumberOfNodes() {
        return numberOfNodes;
    }

    /**
     * Method to return the array of nodes the course contains.
     */
    public int[] getNodes() {
        return nodes;
    }
}
```

Listing 13: Competitor class.

```java
/* File Name: Competitor.java
 * Description: Competitor class which stores all members and functions
    pertaining to a competitor.
 * First Created: 15/03/2013
 * Last Modified: 18/03/2013
 */
package Data_Structures;

import java.util.ArrayList;

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

    private String name;
    private int number;
    private char course;
    private char status;
    private int[] checkpoints;
    private int checkpointIndex;

    /**
     * Constructor to initialise competitor.
     *
     * @param number The competitor's number.
     * @param course The competitor's course.
     * @param name The competitor's name.
     */
    public Competitor(int number, char course, String name, Event event) {
        this.number = number;
        this.course = course;
        this.name = name;
        this.checkpoints = setCheckpoints(event);
        this.checkpointIndex = 0;
        this.status = 'N'; //Not started yet.
    }

    /**
     * Method to return the competitor's number.
     *
     * @return The number of the competitor.
     */
    public int getNumber() {
        return number;
    }

    /**
```

```
     * Method to return the course the competitor is entered on.
     *
     * @return The course the competitor entered in on.
     */
    public char getCourse() {
        return course;
    }

    /**
     * Method to return the competitor's name.
     *
     * @return The name of the competitor.
     */
    public String getName() {
        return name;
    }

    /**
     * Method to return the competitor's status.
     *
     * @return The status of the competitor.
     */
    public char getStatus() {
        return status;
    }

    /**
     * Method to return the index of the last checkpoint the competitor
         arrived
     * at.
     *
     * @return The index of the last checkpoint the competitor arrived at.
     */
    public int getCheckpointIndex() {
        return checkpointIndex;
    }

    /**
     * Method to return the int array of checkpoints.
     *
     * @return The int array of checkpoints.
     */
    public int[] getCheckpoints() {
        return checkpoints;
    }

    /**
     * Method to get the nodes which are recordable checkpoints (non-
         junction
     * nodes).
     *
     * @param event The event instance.
     * @return The int array of checkpoints.
     */
    private int[] setCheckpoints(Event event) {
        ArrayList<Integer> checkpointsList = new ArrayList<Integer>();
        Course courseReference = event.retrieveCourse(course);

        for (int counter = 0; counter < courseReference.getNumberOfNodes();
            counter++) {
            for (int counter2 = 0; counter2 < event.getNodes().size();
                counter2++) {
                if ((!event.getNodes().get(counter2).getType().equals("JN"))
                        && (event.getNodes().get(counter2).getNumber() ==
                            courseReference.getNodes()[counter])) {
                    checkpointsList.add(event.getNodes().get(counter2).
                        getNumber());
                    break;
                }
            }
        }

        int[] intList = new int[checkpointsList.size()];

        for (int counter = 0; counter < checkpointsList.size(); counter++) {
            intList[counter] = checkpointsList.get(counter).intValue();
        }

        return intList;
    }

    /**
     * Method to set the status of the competitor.
     *
     * @param status The current status of the competitor.
     */
    public void setStatus(char status) {
        this.status = status;
    }

    /**
     * Method to increment the checkpoint index by 1.
     */
    public void incrementCheckpointIndex() {
        checkpointIndex++;
    }
}
```

Listing 14: Record class.

```
/* File Name: Record.java
 * Description: Record class which stores all members and functions
     pertaining to checking a competitor in at a checkpoint.
 * First Created: 15/03/2013
 * Last Modified: 17/03/2013
 */
package Data_Structures;

import java.util.Date;

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

    private Event event;
    private char competitorStatus;
    private int checkpoint;
    private int competitorNumber;
    private Date time;

    /**
     * Constructor to initialise record data when read in from file.
     *
     * @param checkpoint The number of the checkpoint.
     * @param competitorNumber The number of the competitor.
```

```java
     * @param time The time of the record.
     */
    public Record(char status, int checkpoint, int competitorNumber, Date
        time) {
        this.competitorStatus = status;
        this.checkpoint = checkpoint;
        this.competitorNumber = competitorNumber;
        this.time = time;
    }

    /**
     * Constructor to initialise record data when recorded through GUI.
     *
     * @param checkpoint The number of the checkpoint.
     * @param competitorNumber The number of the competitor.
     * @param time The time of the record.
     */
    public Record(int checkpoint, char status, int competitorNumber, Date
        time) {
        this.competitorStatus = status;
        this.checkpoint = checkpoint;
        this.competitorNumber = competitorNumber;
        this.time = time;
    }

    /**
     * Method to return the status of the competitor as marked by the
     * checkpoint.
     *
     * @return The status of the competitor.
     */
    public char getCompetitorStatus() {
        return competitorStatus;
    }

    /**
     * Method to return the checkpoint number being recorded.
     *
     * @return The checkpoint number.
     */
    public int getCheckpointNumber() {
        return checkpoint;
    }

    /**
     * Method to return the competitor number being recorded.
     *
     * @return The competitor number.
     */
    public int getCompetitorNumber() {
        return competitorNumber;
    }

    /**
     * Method to return the time being recorded.
     *
     * @return The time of the record.
     */
    public Date getTime() {
        return time;
    }
}
```

Listing 15: FileHandler class.

```java
/* File Name: FileHandler.java
 * Description: FileHandler class which stores methods to handle the reading
     of files.
 * First Created: 15/03/2013
 * Last Modified: 18/03/2013
 */
package File_Handling;

import Data_Structures.Competitor;
import Data_Structures.Course;
import Data_Structures.Event;
import Data_Structures.Node;
import Data_Structures.Record;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.channels.FileChannel;
import java.nio.channels.FileLock;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

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

    /**
     * Method to read in all the details for the nodes pertaining to an
        event.
     *
     * @param fileName The file name required to access the file needed.
     * @param event The event instance.
     * @return True if file loaded successfully, else false if it fails at
        any
     * point.
     */
    public boolean readNodes(String fileName, Event event) throws
       IOException {
        String input;
        int nodeNumber;
        String nodeType;
        String[] subStrings;
        String pattern = "(\\d+\\s+([A-Z]{2})$)"; //Regular expression for
           nodes file.

        try {
            BufferedReader reader = new BufferedReader(new FileReader(
               fileName));

            while ((input = reader.readLine()) != null) {
                if (input.matches(pattern)) { //Checks to make sure the line
                     is in the right format.
                    subStrings = input.split("\\s+"); //Gets rid of
                        whitespace and separates the two sides into two
                        substrings.
                    nodeNumber = Integer.parseInt(subStrings[0]); // Retrieves the node number by parsing the string into an int.
                    nodeType = subStrings[1]; //Retrieves the node type.

                    Node node = new Node(nodeNumber, nodeType); //Creates new node with parameters read in.
                    event.getNodes().add(node); //Adds new node to array list of nodes.

                    if (node.getType().equals("CP") || node.getType().equals("MC")) {
                        event.getCheckpoints().add(node); //Adds new node to array list of checkpoints if the node is of type "CP or "MC".
                    }
                } else {
                    System.out.print("Invalid line format. Cancelling loading of nodes.\n\n");
                    reader.close();
                    return false;
                }
            }

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

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

    /**
     * Method to read in all the details for the courses pertaining to an event.
     *
     * @param fileName The file name required to access the file needed.
     * @param event The event instance.
     * @return True if file loaded successfully, else false if it fails at any
     * point.
     */
    public boolean readCourses(String fileName, Event event) throws IOException {
        String input;
        char courseLetter;
        int numberOfNodes;
        int[] nodes;
        String[] subStrings;
        String pattern = "(([A-Za-z]+)((\\s+\\d+)+)$)"; //Regular expression for courses file.

        try {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));

            while ((input = reader.readLine()) != null) {
                if (input.matches(pattern)) { //Checks to make sure the line is in the right format.
                    subStrings = input.split("\\s+"); //Gets rid of whitespace and separates the strings into substrings.
                    courseLetter = subStrings[0].charAt(0); //Retrieves the course letter.
                    numberOfNodes = Integer.parseInt(subStrings[1]);
                    nodes = new int[numberOfNodes];

                    for (int counter = 0; counter < numberOfNodes; counter++) {
                        if (event.checkNodeExists(Integer.parseInt(subStrings[counter + 2]))) {
                            nodes[counter] = Integer.parseInt(subStrings[counter + 2]);
                        } else {
                            System.out.print("Invalid node in course file found. Cancelling loading of courses\n\n");
                            reader.close();
                            return false;
                        }
                    }

                    Course course = new Course(courseLetter, numberOfNodes, nodes); //Creates new course with parameters read in.
                    event.getCourses().add(course); //Adds new course to array list of courses.
                } else {
                    System.out.print("Invalid line format. Cancelling loading of courses\n\n");
                    reader.close();
                    return false;
                }
            }

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

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

    /**
     * Method to read in all the details for the competitors pertaining to an
     * event.
     *
     * @param fileName The file name required to access the file needed.
     * @param event The event instance.
     * @return True if file loaded successfully, else false if it fails at any
     * point.
     */
    public boolean readCompetitors(String fileName, Event event) throws IOException {
        String input;
        int competitorNumber;
        char courseLetter;
        String[] subStrings;
        String competitorName;
        String pattern = "(\\d+\\s+[A-Za-z]((\\s+[A-Za-z]{1}[a-z]+)+)$)"; // Regular expression for competitors file.

        try {
            BufferedReader reader = new BufferedReader(new FileReader(fileName));

            while ((input = reader.readLine()) != null) {
                if (input.matches(pattern)) { //Checks to make sure the line is in the right format.
                    subStrings = input.split("\\s+"); //Gets rid of whitespace and separates the strings into substrings.
                    competitorNumber = Integer.parseInt(subStrings[0]); // Retrieves the competitor number by parsing the string into an int.

                    if (event.checkCourseExists(subStrings[1].charAt(0))) {
                        courseLetter = subStrings[1].charAt(0); //Retrieves the course the competitor is entering in on.
                    } else {
                        System.out.print("Invalid course in competitor file found. Cancelling loading of competitors.\n\n");
                        reader.close();
                        return false;
                    }

                    competitorName = subStrings[2];

                    if (subStrings.length > 3) {
                        for (int counter = 3; counter < subStrings.length; counter++) {
                            competitorName += " " + subStrings[counter]; // Concatanates name substrings together.
                        }
                    }

                    Competitor competitor = new Competitor(competitorNumber, courseLetter, competitorName, event); //Creates new competitor with parameters read in.
                    event.getCompetitors().add(competitor); //Adds new competitor to array list of competitors.
                } else {
                    System.out.print("Invalid line format. Cancelling loading of competitors.\n\n");
                    reader.close();
                    return false;
                }
            }

```

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

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

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

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

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

            BufferedReader reader = new BufferedReader(new FileReader("cp_times.txt"));

            while ((input = reader.readLine()) != null) {
                currentLineNumber++;
                if (currentLineNumber > lastLineNumber) {
                    if (input.matches(pattern)) { //Checks to make sure the line is in the right format.
```

```java
                        subStrings = input.split("[\\s+]"); //Gets rid of whitespace and separates the strings into substrings.
                        competitorStatus = subStrings[0].charAt(0); //Retrieves competitor status.
                        nodeNumber = Integer.parseInt(subStrings[1]); //Retrieves the node number by parsing the string into an int.
                        competitorNumber = Integer.parseInt(subStrings[2]); //Retrieves the competitor number by parsing the string into an int.
                        time = formatter.parse(subStrings[3]); //Retrieves the time being recorded and formats it into 24 hour HH:MM.

                        Competitor competitor = event.retrieveCompetitor(competitorNumber);
                        if (competitor.getStatus() == 'T') {
                            competitor.incrementCheckpointIndex(); //Increments the competitor's checkpoint intdex by 1.
                        }

                        Record record = new Record(competitorStatus, nodeNumber, competitorNumber, time); //Creates new record with parameters read in.
                        event.getRecords().add(record); //Adds new record to array list of records.
                        competitor.setStatus(competitorStatus); //Updates competitor's status.

                        event.setLastLineRead(currentLineNumber);
                        event.setLastRecordedTime(time);
                    } else {
                        System.out.print("Invalid line format. Cancelling loading of times.\n\n");
                        reader.close();
                        lock.release();
                        channel.close();
                        return false;
                    }
                }
            }

            event.setTimesFilesExistsTrue(); //Lets the event instance know that an event does exist.
            reader.close(); //Closes reader.
            lock.release(); //Releases file lock.
            channel.close(); //Closes channel ensuring lock release and release of resources.
            return true;
        } catch (FileNotFoundException ex) {
            System.out.print("Could not open file that contains times.\n\n");
        }
        return false;
    }

    /**
     * Method to write a record on a line in the time records file.
     *
     * @param fileName The file name required to access the file needed.
     * @param record The record to be written.
     * @return True if file written to successfully, else false if it fails
        at
     * any point.
     */
    public boolean appendTimeRecord(String fileName, Record record) {
        SimpleDateFormat formatter = new SimpleDateFormat("HH:mm");

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

            FileWriter writer = new FileWriter(fileName, true); //True sets
                append mode.
            writer.write(record.getCompetitorStatus() + " " + record.
                getCheckpointNumber()
                    + " " + record.getCompetitorNumber() + " " + formatter.
                        format(record.getTime()) + "\n");
            writer.close();
            lock.release();
            channel.close();
            return true;
        } catch (IOException ex) {
            System.out.print("\nCould not open file for writing.\n\n");
        }
        return false;
    }
}
```

Listing 16: TypeWindow class.

```
/* File Name: TypeWindow.java
 * Description: TypeWindow GUI class using swing.
 * First Created: 17/03/2013
 * Last Modified: 18/03/2013
 */
package GUI;

import Data_Structures.Event;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ButtonGroup;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.border.EmptyBorder;

/**
 * @author Chris Savill, chs17@aber.ac.uk
 */
public class TypeWindow extends JFrame implements ActionListener {

    private Event event;
    private boolean medicalSelected;
    private JFrame typeFrame;
    private JPanel typePanel, bottomPanel;
    private JLabel typeLabel;
    private JRadioButton time, medical;
```

```java
    private ButtonGroup typeGroup;
    private JButton next;

    /**
     * Constructor for TypeWindow GUI class that sets up and launches GUI.
     *
     * @param event The event instance.
     */
    public TypeWindow(Event event) {
        this.event = event;
        medicalSelected = false;

        //Setup frame:
        typeFrame = new JFrame("Checkpoint Type Selection");
        typeFrame.setPreferredSize(new Dimension(300, 200));
        typeFrame.setLocation(400, 200);
        typeFrame.setLayout(new BorderLayout());
        typeFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Sets the default close operation
        typeFrame.setIconImage(new ImageIcon("horse.jpg").getImage()); // Loads an image and sets it as the frame icon
        ///////////////////////////////////////////////////////////////

        //Setup panels:
        typePanel = new JPanel(new BorderLayout()); //Creates new JPanel.
        typePanel.setBorder(new EmptyBorder(25, 25, 25, 25));  //Sets an invisible border to simulate a padding effect
        typeFrame.add(typePanel, BorderLayout.NORTH); //Adds panel to frame and places it in NORTH container.
        bottomPanel = new JPanel();
        typeFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to frame and places it in SOUTH container.
        ///////////////////////////////////////////////////////////////

        //Setup checkpoint panel components:
        typeLabel = new JLabel("Select Checkpoint Type Below: ");
        typePanel.add(typeLabel, BorderLayout.NORTH);

        time = new JRadioButton("Time Checkpoint");
        time.setActionCommand("time");
        time.addActionListener(this);
        time.setSelected(true); //Defaults this button to be selected.
        typePanel.add(time, BorderLayout.CENTER);
        medical = new JRadioButton("Medical Checkpoint");
        medical.setActionCommand("medical");
        medical.addActionListener(this);
        medical.setSelected(false);
        typePanel.add(medical, BorderLayout.SOUTH);

        typeGroup = new ButtonGroup(); //Creates a group for the radio buttons to prevent both from being selected.
        typeGroup.add(time);
        typeGroup.add(medical);
        ///////////////////////////////////////////////////////////////

        //Setup bottom panel components:
        next = new JButton("Next");
        next.setPreferredSize(new Dimension(100, 50));
        bottomPanel.add(next);
        next.addActionListener(this);
        ///////////////////////////////////////////////////////////////

        //Finialise frame setup:
```

```
        typeFrame.pack();
        typeFrame.setVisible(true); //Makes the frame visible
        ///////////////////////////////////////////////////////////////
    }

    /**
     * Method to handle actions performed.
     *
     * @param evt The event triggered.
     */
    @Override
    public void actionPerformed(ActionEvent evt) {
        String actionCommand = evt.getActionCommand();

        switch (actionCommand) {
            case "Next":
                if (medicalSelected == true) {
                    typeFrame.setVisible(false);
                    SelectionWindow selectionWindow = new SelectionWindow(
                        event, "MC", typeFrame);
                } else {
                    typeFrame.setVisible(false);
                    SelectionWindow selectionWindow = new SelectionWindow(
                        event, "CP", typeFrame);
                }

                typeFrame.dispose();
                this.dispose();
                break;
            case "time":
                medicalSelected = false;
                break;
            case "medical":
                medicalSelected = true;
                break;
        }
    }
}
```

Listing 17: SelectionWindow class.

```
/* File Name: SelectionWindow.java
 * Description: SelectionWindow GUI class using swing.
 * First Created: 16/03/2013
 * Last Modified: 17/03/2013
 */
package GUI;

import Data_Structures.Competitor;
import Data_Structures.Event;
import Data_Structures.Node;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.DefaultListModel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
```

```java
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

/**
 * @author Chris Savill, chs17@aber.ac.uk
 */
public class SelectionWindow extends JFrame implements ActionListener,
    ListSelectionListener {

    private Event event;
    private int checkpoint;
    private String type;
    private int competitor;
    private boolean checkpointSelected = false;
    private boolean competitorSelected = false;
    private JFrame selectionFrame, typeFrame;
    private JPanel checkpointPanel, competitorPanel, bottomPanel;
    private JLabel checkpointLabel, competitorLabel;
    private DefaultListModel checkpointListModel, competitorListModel;
    private JList checkpointList, competitorList;
    private JScrollPane checkpointListScrollBar, competitorListScrollBar;
    private JButton next;

    /**
     * Constructor for SelectionWindow GUI class, sets up and runs GUI.
     * @param event The event instance.
     * @param type The type of the checkpoint.
     * @param typeFrame The JFrame this transitioned from.
     */
    public SelectionWindow(Event event, String type, JFrame typeFrame) {
        typeFrame.dispose();
        this.typeFrame = typeFrame;
        this.event = event;
        this.type = type;

        //Setup frame:
        selectionFrame = new JFrame("Checkpoint and Competitor Selection");
        selectionFrame.setLocation(400, 200);
        selectionFrame.setLayout(new BorderLayout());
        selectionFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //
            Sets the default close operation
        selectionFrame.setIconImage(new ImageIcon("horse.jpg").getImage());
            //Loads an image and sets it as the frame icon
        ///////////////////////////////////////////////////////////////

        //Setup panels:
        checkpointPanel = new JPanel(new BorderLayout()); //Creates new
            JPanel.
        checkpointPanel.setBorder(new EmptyBorder(10, 25, 10, 25));  //Sets
            an invisible border to simulate a padding effect
        selectionFrame.add(checkpointPanel, BorderLayout.WEST); //Adds panel
             to frame and places it in WEST container.
        competitorPanel = new JPanel(new BorderLayout());
        competitorPanel.setBorder(new EmptyBorder(10, 25, 10, 25));
        selectionFrame.add(competitorPanel, BorderLayout.EAST); //Adds panel
             to frame and places it in EASTcontainer.
        bottomPanel = new JPanel();
        selectionFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
```

```
            frame and places it in SOUTH container.
        ///////////////////////////////////////////////////////////////

        //Setup checkpoint panel components:
        checkpointLabel = new JLabel("Select Checkpoint Below: ");
        checkpointPanel.add(checkpointLabel, BorderLayout.NORTH);

        checkpointListModel = new DefaultListModel();
        checkpointList = new JList(checkpointListModel);
        checkpointList.setBorder(new LineBorder(Color.BLACK));
        checkpointPanel.add(checkpointList, BorderLayout.CENTER);
        checkpointList.addListSelectionListener(this);

        checkpointListScrollBar = new JScrollPane(checkpointList);
        checkpointListScrollBar.setPreferredSize(new Dimension(50, 100));
        checkpointListScrollBar.setVerticalScrollBarPolicy(
           ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED); //Adds
           vertical scrollbar to JList
        checkpointListScrollBar.setHorizontalScrollBarPolicy(
           ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED); //Adds
           horizontal scrollbar to JList
        checkpointPanel.add(checkpointListScrollBar);
        ///////////////////////////////////////////////////////////////

        //Setup competitor panel components:
        competitorLabel = new JLabel("Select Competitor Below: ");
        competitorPanel.add(competitorLabel, BorderLayout.NORTH);

        competitorListModel = new DefaultListModel();
        competitorList = new JList(competitorListModel);
        competitorList.setBorder(new LineBorder(Color.BLACK));
        competitorPanel.add(competitorList, BorderLayout.CENTER);
        competitorList.addListSelectionListener(this);

        competitorListScrollBar = new JScrollPane(competitorList);
        competitorListScrollBar.setPreferredSize(new Dimension(400, 300));
        competitorListScrollBar.setVerticalScrollBarPolicy(
           ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED); //Adds
           vertical scrollbar to JList
        competitorListScrollBar.setHorizontalScrollBarPolicy(
           ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED); //Adds
           horizontal scrollbar to JList
        competitorPanel.add(competitorListScrollBar);
        ///////////////////////////////////////////////////////////////

        //Setup bottom panel components:
        next = new JButton("Next");
        next.setPreferredSize(new Dimension(100, 50));
        bottomPanel.add(next);
        next.addActionListener(this);
        ///////////////////////////////////////////////////////////////

        //Finialise frame setup:
        addCheckpoints();
        addCompetitors();
        selectionFrame.pack();
        selectionFrame.setVisible(true); //Makes the frame visible
        ///////////////////////////////////////////////////////////////
    }

    /**
     * Method that adds the checkpoint checkpoints to the checkpoint JList
     */
```

```java
    public void addCheckpoints() {
        checkpointListModel.removeAllElements();

        for (Node currentCheckpoint : event.getCheckpoints()) {
            if (currentCheckpoint.getType().equals(type)) {
                checkpointListModel.addElement(currentCheckpoint.getNumber()
                    + ": " + currentCheckpoint.getType());
            }
        }
    }

    /**
     * Method that adds the competitors to the competitor JList
     */
    public void addCompetitors() {
        competitorListModel.removeAllElements();

        for (Competitor currentCompetitor : event.getCompetitors()) {
            competitorListModel.addElement("Competitor: " +
               currentCompetitor.getNumber()
                    + "   Course: " + currentCompetitor.getCourse() + "
                       Name: " + currentCompetitor.getName());
        }
    }

    /**
     * Method to handle actions performed.
     * @param evt The event triggered.
     */
    @Override
    public void actionPerformed(ActionEvent evt) {
        String actionCommand = evt.getActionCommand();

        if (actionCommand.equals("Next")) {
            if (checkpointSelected == true && competitorSelected == true) {
                selectionFrame.setVisible(false);
                TimeWindow timeWindow = new TimeWindow(event, checkpoint,
                    type, competitor, selectionFrame, typeFrame);
                selectionFrame.dispose();
                this.dispose();
            } else {
                JOptionPane.showMessageDialog(selectionFrame, "Please select
                     both a checkpoint and competitor.");
            }
        }
    }

    /**
     * Method to handle values changing in a JList.
     * @param evt The event triggered.
     */
    @Override
    public void valueChanged(ListSelectionEvent evt) {

        if (!evt.getValueIsAdjusting()) {
            JList list = (JList) evt.getSource();

            if (list.equals(checkpointList)) {
                checkpoint = event.retrieveCheckpointNumber(type, list.
                    getSelectedIndex(), list.getModel().getSize());
                checkpointSelected = true;
            } else if (list.equals(competitorList)) {
                competitor = event.getCompetitors().get(list.
                        getSelectedIndex()).getNumber();
                    competitorSelected = true;
                }
            }
        }
}
```

Listing 18: TimeWindow class.

```
/* File Name: TimeWindow.java
 * Description: TimeWindow GUI class using swing.
 * First Created: 16/03/2013
 * Last Modified: 17/03/2013
 */
package GUI;

import Data_Structures.Event;
import Data_Structures.Record;
import File_Handling.FileHandler;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.text.ParseException;
import java.util.Calendar;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.SpinnerDateModel;
import javax.swing.border.EmptyBorder;

/**
 * @author Chris Savill, chs17@aber.ac.uk
 */
public class TimeWindow extends JFrame implements ActionListener {

    private Event event;
    private FileHandler fileHandler;
    private int checkpoint;
    private String type;
    private int competitor;
    private int status;
    private JFrame timeFrame, typeFrame;
    private JPanel timePanel, bottomPanel;
    private JLabel timeLabel;
    private JButton submit;
    private Date date;
    private SpinnerDateModel spinnerModel;
    private JSpinner spinner;
    private JSpinner.DateEditor dateEditor;

    /**
     * Constructor for TimeWindow GUI class that sets up and launches the
        GUI.
     *
```

```java
     * @param event The event instance.
     * @param checkpoint The checkpoint number.
     * @param type The checkpoint type.
     * @param competitor The competitor number.
     * @param selectionFrame The JFrame this transitioned from.
     * @param typeFrame The JFrame that is reopened after this JFrame closes
        .
     */
    public TimeWindow(Event event, int checkpoint, String type, int
       competitor, JFrame selectionFrame, JFrame typeFrame) {
        selectionFrame.dispose();

        this.typeFrame = typeFrame;
        this.event = event;
        this.checkpoint = checkpoint;
        this.type = type;
        this.competitor = competitor;
        fileHandler = new FileHandler();

        //Setup frame:
        timeFrame = new JFrame("Time Of Record");

        if (type.equals("MC")) {
            status = getMedicalOptions();
        } else {
            status = 0; //Comeptitor status not a medical related status.
        }

        timeFrame.setLocation(400, 200);
        timeFrame.setLayout(new BorderLayout());
        timeFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Sets the
            default close operation
        timeFrame.setIconImage(new ImageIcon("horse.jpg").getImage()); //
           Loads an image and sets it as the frame icon
        //////////////////////////////////////////////////////////////

        //Setup panels:
        timePanel = new JPanel(new BorderLayout()); //Creates new JPanel.
        timePanel.setBorder(new EmptyBorder(10, 25, 10, 25));  //Sets an
           invisible border to simulate a padding effect
        timeFrame.add(timePanel, BorderLayout.WEST); //Adds panel to frame
           and places it in WEST container.
        bottomPanel = new JPanel();
        timeFrame.add(bottomPanel, BorderLayout.SOUTH); //Adds panel to
           frame and places it in SOUTH container.
        //////////////////////////////////////////////////////////////

        //Setup checkpoint panel components:
        timeLabel = new JLabel("Select Time Below: ");
        timePanel.add(timeLabel, BorderLayout.NORTH);

        date = new Date();
        spinnerModel = new SpinnerDateModel(date, null, null, Calendar.
           HOUR_OF_DAY);
        spinner = new JSpinner(spinnerModel);
        dateEditor = new JSpinner.DateEditor(spinner, "HH:mm"); //24-hour
           format.
        spinner.setEditor(dateEditor);
        timePanel.add(spinner, BorderLayout.CENTER);
        //////////////////////////////////////////////////////////////

        //Setup bottom panel components:
        submit = new JButton("Submit Checkpoint Record");
        submit.setPreferredSize(new Dimension(225, 30));
        bottomPanel.add(submit);
        submit.addActionListener(this);
        ///////////////////////////////////////////////////////////////

        //Finialise frame setup:
        timeFrame.pack();
        timeFrame.setVisible(true); //Makes the frame visible
        ///////////////////////////////////////////////////////////////
    }

    /**
     * Method to handle actions performed.
     *
     * @param evt The event triggered.
     */
    @Override
    public void actionPerformed(ActionEvent evt) {
        String actionCommand = evt.getActionCommand();

        if (actionCommand.equals("Submit Checkpoint Record")) {
            try {
                if (!fileHandler.readTimes(event.getFileNames()[3], event)) {
                     JOptionPane.showMessageDialog(timeFrame, "Failed to load time records from file.");
                }
            } catch (IOException | ParseException ex) {
                Logger.getLogger(TimeWindow.class.getName()).log(Level.SEVERE, null, ex);
            }

            if (event.checkNewRecord(checkpoint, status, competitor, (Date) spinner.getValue())) {
                char finalStatus = event.determineFinalStatus(checkpoint, status, competitor);

                Record record = new Record(checkpoint, finalStatus, competitor, (Date) spinner.getValue());
                event.getRecords().add(record);

                fileHandler.appendTimeRecord(event.getFileNames()[3], record);
                JOptionPane.showMessageDialog(timeFrame, "Time record succesfully added.");
            } else {
                JOptionPane.showMessageDialog(timeFrame, "Non-valid record. Record will not added.");
            }

            timeFrame.dispose(); //Closes frame and releases resourses.
            this.dispose(); //Releases resources.
            TypeWindow typeFrame = new TypeWindow(event);

        }
    }

    /**
     * Method to get the user to select the status of the competitor at the
     * medical checkpoint.
     *
     * @return The status of the competitor at the medical checkpoint.
     */
    public int getMedicalOptions() {
        String[] options = new String[]{"Arriving", "Departing", "Excluded"
            };

        int selection = JOptionPane.showOptionDialog(timeFrame, "Is the
            competitor being marked as 'Arriving',"
                + " 'Departing' or as 'Excluded' on medical grounds?", "
                    Medical Marking", JOptionPane.DEFAULT_OPTION,
                JOptionPane.PLAIN_MESSAGE, null, options, options[0]);

        if (selection == 0) {
            return 1; //Competitor status to be set to arriving at medical
                checkpoint.
        } else if (selection == 1) {
            return 2; //Competitor status to be set to departing medical
                checkpoint.
        } else if (selection == 2) {
            return 3; //Competitor status to be set to excluded based on
                medical grounds.
        }

        return 0;
    }
}
```

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-
   conf
make[1]: Entering directory ‘/home/clsavill/GitHub/Runners_and_Riders_3_Part
   /Event_Manager_Program’
rm -f -r build/Debug
rm -f dist/Debug/GNU-Linux-x86/event_manager_program
make[1]: Leaving directory ‘/home/clsavill/GitHub/Runners_and_Riders_3_Part/
   Event_Manager_Program’


CLEAN SUCCESSFUL (total time: 57ms)

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

```
gcc    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/main.o.d -o build/Debug/
   GNU-Linux-x86/main.o main.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/event.o.d
gcc    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/event.o.d -o build/Debug
   /GNU-Linux-x86/event.o event.c
mkdir -p build/Debug/GNU-Linux-x86
rm -f build/Debug/GNU-Linux-x86/tracks.o.d
gcc    -c -g -MMD -MP -MF build/Debug/GNU-Linux-x86/tracks.o.d -o build/
   Debug/GNU-Linux-x86/tracks.o tracks.c
mkdir -p dist/Debug/GNU-Linux-x86
gcc     -o dist/Debug/GNU-Linux-x86/event_manager_program build/Debug/GNU-
   Linux-x86/loader.o build/Debug/GNU-Linux-x86/logger.o build/Debug/GNU-
   Linux-x86/updater.o build/Debug/GNU-Linux-x86/courses.o build/Debug/GNU-
   Linux-x86/competitors.o build/Debug/GNU-Linux-x86/nodes.o build/Debug/GNU
   -Linux-x86/main.o build/Debug/GNU-Linux-x86/event.o build/Debug/GNU-Linux
   -x86/tracks.o
make[2]: Leaving directory '/home/clsavill/GitHub/Runners_and_Riders_3_Part/
   Event_Manager_Program'
make[1]: Leaving directory '/home/clsavill/GitHub/Runners_and_Riders_3_Part/
   Event_Manager_Program'


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

11 Run through of Event Manager Program

```
Event Monitoring Program Launching...

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

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

Event file loaded in successfully.
Event loading finished.


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

Nodes file loaded in successfully.
Node loading finished.


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

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

Tracks file loaded in successfully.
Track loading finished.


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

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

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

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

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

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

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

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


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

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

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

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


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

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

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

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

Press enter to continue.


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

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

 Printing competitors that are out on a course...

========================================================================================================================
| Number |                          Name                          | Course | Last Recorded Checkpoint | Presumed Location |
========================================================================================================================
|   001  | Ace Abbey                                              |   E    |           13             |     TN - 01       |
|   003   | Ace Fudge                                            |    A    |            14            |      TN - 18       |
|   004   | Amber Abbey                                          |    C    |            13            |      TC - 13       |
|   005   | Amber Fudge                                          |    E    |            13            |      TN - 13       |
|   006   | April Abbey                                          |    D    |            09            |      TN - 02       |
|   007   | April Fudge                                          |    B    |            14            |       A - 14       |
|   008   | Ash Abbey                                            |    F    |            13            |      TN - 13       |
|   009   | Ash Fudge                                            |    D    |            09            |      TN - 02       |
|   010   | Asti Abbey                                           |    A    |            07            |      TN - 15       |
|   011   | Asti Fudge                                           |    A    |            07            |      TN - 15       |
|   012   | Autumn Abbey                                         |    C    |            07            |      TN - 15       |
|   013   | Autumn Fudge                                         |    B    |            07            |      TN - 08       |
|   014   | Barfields Marco Abbey                                |    A    |            07            |      TN - 08       |
|   016   | Barfields Marco Fudge                                |    F    |            13            |      TN - 13       |
|   017   | Basil Abbey                                          |    B    |            07            |      TN - 08       |
|   018   | Basil Fudge                                          |    A    |            05            |      TN - 07       |
|   019   | Beatrice Abbey                                       |    C    |            05            |      TN - 07       |
|   020   | Beatrice Fudge                                       |    A    |            05            |      TN - 05       |
|   022   | Beau Abbey                                           |    D    |            05            |      TN - 05       |
|   023   | Beau Fudge                                           |    C    |            05            |      TN - 05       |
|   024   | Bella Abbey                                          |    B    |            04            |      TN - 04       |
|   026   | Bella Fudge                                          |    F    |            09            |      TN - 12       |
|   027   | Black Jack Abbey                                     |    F    |            09            |      TN - 12       |
|   028   | Black Jack Fudge                                     |    A    |            04            |      TN - 04       |
|   030   | Blue Abbey                                           |    B    |            04            |      TN - 04       |
|   031   | Blue Fudge                                           |    B    |            01            |      TN - 03       |
|   032   | Bobby Abbey                                          |    A    |            04            |      TC - 04       |
|   034   | Bobby Fudge                                          |    E    |            01            |      TN - 11       |
|   035   | Bubbles Abbey                                        |    C    |            01            |      TN - 02       |
|   036   | Bubbles Fudge                                        |    D    |            01            |      TN - 02       |
|   038   | Captain Abbey                                        |    A    |            01            |      TN - 02       |
|   039   | Captain Fudge                                        |    B    |            01            |      TN - 01       |
|   040   | Chalkie Abbey                                        |    D    |            01            |      TN - 01       |
|   041   | Chalkie Fudge                                        |    F    |            01            |      TN - 01       |
|   042   | Copper Abbey                                         |    E    |            01            |      TN - 01       |
|   044   | Copper Fudge                                         |    B    |            01            |      TN - 01       |
|   045   | Diamond Abbey                                        |    C    |            01            |      TN - 01       |
|   046   | Diamond Fudge                                        |    B    |            01            |      TN - 01       |
==========================================================================================================================

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

Number of Competitors out on course: 38 out of 102

Current Event Time: 9:26.

Press enter to continue.


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

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

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

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

Press enter to continue.


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

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

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

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

Press enter to continue.


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

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

 Printing competitors that have not yet started...

=================================================================================
| Number  |                       Name                      | Course | Location |
=================================================================================
|   095   | Ruby Abbey                                      |   F    |    NS    |
|   097   | Ruby Fudge                                      |   C    |    NS    |
|   098   | Sapphire Abbey                                  |   C    |    NS    |
|   100   | Sapphire Fudge                                         |   F    |    NS    |
|   101   | Scarlet Abbey                                          |   C    |    NS    |
|   102   | Scarlet Fudge                                          |   F    |    NS    |
|   103   | sienna Abbey                                           |   D    |    NS    |
|   106   | sienna Fudge                                           |   B    |    NS    |
|   107   | Silver Abbey                                           |   F    |    NS    |
|   108   | Silver Fudge                                           |   A    |    NS    |
|   109   | Smokey Abbey                                           |   A    |    NS    |
|   110   | Smokey Fudge                                           |   D    |    NS    |
|   111   | Snowy Abbey                                            |   E    |    NS    |
|   113   | Snowy Fudge                                            |   C    |    NS    |
|   114   | sonic Abbey                                            |   A    |    NS    |
|   115   | sonic Fudge                                            |   D    |    NS    |
|   117   | Summer Abbey                                           |   A    |    NS    |
|   118   | Summer Fudge                                           |   E    |    NS    |
|   121   | Tango Abbey                                            |   B    |    NS    |
|   122   | Tango Fudge                                            |   A    |    NS    |
|   123   | Topaz Abbey                                            |   B    |    NS    |
|   124   | Topaz Fudge                                            |   F    |    NS    |
|   126   | Zizou Abbey                                            |   D    |    NS    |
|   127   | Zizou Fudge                                            |   F    |    NS    |
==================================================================================

Key: NS = Not Started.

Number of Competitors not started yet: 24 out of 102

Current Event Time: 11:39.

Press enter to continue.


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

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

 Printing competitors that are out on a course...

========================================================================================================================
| Number  |                        Name                        | Course | Last Recorded Checkpoint | Presumed Location |
========================================================================================================================
|   010   | Asti Abbey                                         |   A    |          13              |     TN - 01       |
|   011   | Asti Fudge                                         |   A    |          13              |     TN - 01       |
|   014   | Barfields Marco Abbey                              |   A    |          13              |     TN - 13       |
|   018   | Basil Fudge                                        |   A    |          13              |     TN - 13       |
|   020   | Beatrice Fudge                                     |   A    |          13              |     TN - 13       |
|   028   | Black Jack Fudge                                   |   A    |          14              |     TN - 20       |
|   032   | Bobby Abbey                                        |   A    |          14              |     TN - 17       |
|   038   | Captain Abbey                                      |   A    |          17              |     TN - 22       |
|   039   | Captain Fudge                                      |   B    |          13              |     TN - 13       |
|   044   | Copper Fudge                                       |   B    |          14              |     TN - 14       |
|   045   | Diamond Abbey                                      |   C    |          13              |     TN - 01       |
|   049   | Ebony Abbey                                        |   B    |          14              |     TN - 17       |
|   050   | Ebony Fudge                                        |   C    |          13              |     TN - 13       |
|   051   | Ginger Abbey                                       |   C    |          13              |     TN - 13       |
|   052   | Ginger Fudge                                       |   F    |          13              |     TN - 01       |
|   055   | Goldie Fudge                                       |   E    |          13              |     TN - 13       |
|   056   | Honey Abbey                                        |   F    |          13              |     TN - 01       |
|   057   | Honey Fudge                                        |   C    |          07              |     TN - 16       |
|   058   | Izzy Abbey                                         |   A    |          07              |     TN - 17       |
|   060   | Jasmine Abbey                                      |   A    |          07              |     TN - 15       |
|   061   | Jasmine Fudge                                      |   F    |          13              |     TN - 13       |
|   062   | Lady Abbey                                         |   D    |          09              |     TN - 11       |
|   064   | Lady Fudge                                         |   B    |          07              |     TN - 08       |
|   065   | Lady Tara Abbey                                    |   C    |          07              |     TN - 08       |
|   066   | Lady Tara Fudge                                    |   B    |          07              |     TN - 08       |
|   067   | Lemon Abbey                                        |   B    |          07              |     TN - 08       |
|   068   | Lemon Fudge                                        |   E    |          09              |     TN - 15       |
|   069   | Lord Abbey                                         |   F    |          13              |     TN - 13       |
|   070   | Lord Fudge                                          |   E   |           09           |     TN - 15       |
|   071   | Lucky Abbey                                         |   A   |           05           |     TN - 05       |
|   074   | Lucky Fudge                                         |   E   |           09           |     TN - 09       |
|   076   | Lord Abbey                                          |   D   |           05           |     TC - 05       |
|   077   | Lord Fudge                                          |   B   |           04           |     TN - 04       |
|   078   | Maddy Abbey                                         |   F   |           09           |     TN - 12       |
|   079   | Maddy Fudge                                         |   A   |           04           |     TN - 04       |
|   080   | Magic Abbey                                         |   D   |           01           |     TN - 03       |
|   081   | Magic Fudge                                         |   D   |           01           |     TN - 03       |
|   083   | Major Abbey                                         |   A   |           01           |     TN - 03       |
|   085   | Major Fudge                                         |   A   |           01           |     TN - 02       |
|   086   | Mattie Abbey                                        |   B   |           01           |     TN - 02       |
|   087   | Mattie Fudge                                        |   A   |           01           |     TN - 02       |
|   089   | Prince Abbey                                        |   B   |           01           |     TN - 01       |
|   090   | Prince Fudge                                        |   A   |           01           |     TN - 01       |
|   091   | Princess Abbey                                      |   B   |           01           |     TN - 01       |
|   092   | Princess Fudge                                      |   B   |           01           |     TN - 01       |
|   093   | Rosie Abbey                                         |   D   |           01           |     TN - 01       |
|   094   | Rosie Fudge                                         |   B   |           01           |     TN - 01       |
========================================================================================================================

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

Number of Competitors out on course: 47 out of 102

Current Event Time: 11:39.

Press enter to continue.


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

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

 Printing competitors that have finished...

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

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

```
Current Event Time: 11:39.

Press enter to continue.


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

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

 Printing results...

===================================================================================
| Number  |                         Name                         | Status |  Time   |
===================================================================================
|   001   | Ace Abbey                                            |   CC   |  02:04  |
|   003   | Ace Fudge                                            |   CC   |  03:52  |
|   004   | Amber Abbey                                          |   CC   |  02:37  |
|   005   | Amber Fudge                                          |   CC   |  02:11  |
|   006   | April Abbey                                          |   CC   |  02:03  |
|   007   | April Fudge                                          |   CC   |  02:46  |
|   008   | Ash Abbey                                            |   CC   |  01:56  |
|   009   | Ash Fudge                                            |   CC   |  01:58  |
|   012   | Autumn Abbey                                         |   CC   |  02:30  |
|   013   | Autumn Fudge                                         |   CC   |  02:53  |
|   016   | Barfields Marco Fudge                                |   CC   |  01:55  |
|   017   | Basil Abbey                                          |   CC   |  02:49  |
|   019   | Beatrice Abbey                                       |   CC   |  02:27  |
|   022   | Beau Abbey                                           |   CC   |  02:02  |
|   024   | Bella Abbey                                          |   CC   |  02:54  |
|   026   | Bella Fudge                                          |   CC   |   01:49  |
|   027   | Black Jack Abbey                                     |   CC   |   01:49  |
|   030   | Blue Abbey                                           |   CC   |   02:43  |
|   031   | Blue Fudge                                           |   CC   |   02:44  |
|   034   | Bobby Fudge                                          |   CC   |   02:03  |
|   035   | Bubbles Abbey                                        |   CC   |   02:32  |
|   040   | Chalkie Abbey                                        |   CC   |   02:03  |
|   042   | Copper Abbey                                         |   CC   |   02:05  |
|   047   | Dinky Abbey                                          |   CC   |   02:10  |
|   048   | Dinky Fudge                                          |   CC   |   01:54  |
====================================================================================

Number of Competitors completed course successfully: 25 out of 102

Current Event Time: 11:39.

Press enter to continue.


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

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

 Printing excluded...

====================================================================================
| Number  |                        Name                          | Status |  At Time |
====================================================================================
|   023   | Beau Fudge                                           |   EI   |   09:49  |
|   036   | Bubbles Fudge                                          |   EI   |   09:57  |
|   041   | Chalkie Fudge                                          |   EI   |   11:05  |
|   046   | Diamond Fudge                                          |   EI   |   11:13  |
|   059   | Izzy Fudge                                             |   EI   |   11:10  |
==========================================================================================

Number of Competitors excluded: 5 out of 102

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

Current Event Time: 11:39.

Press enter to continue.


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

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


Exiting Program...

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

12 Results list produced at the end of an event

13 Log file contents