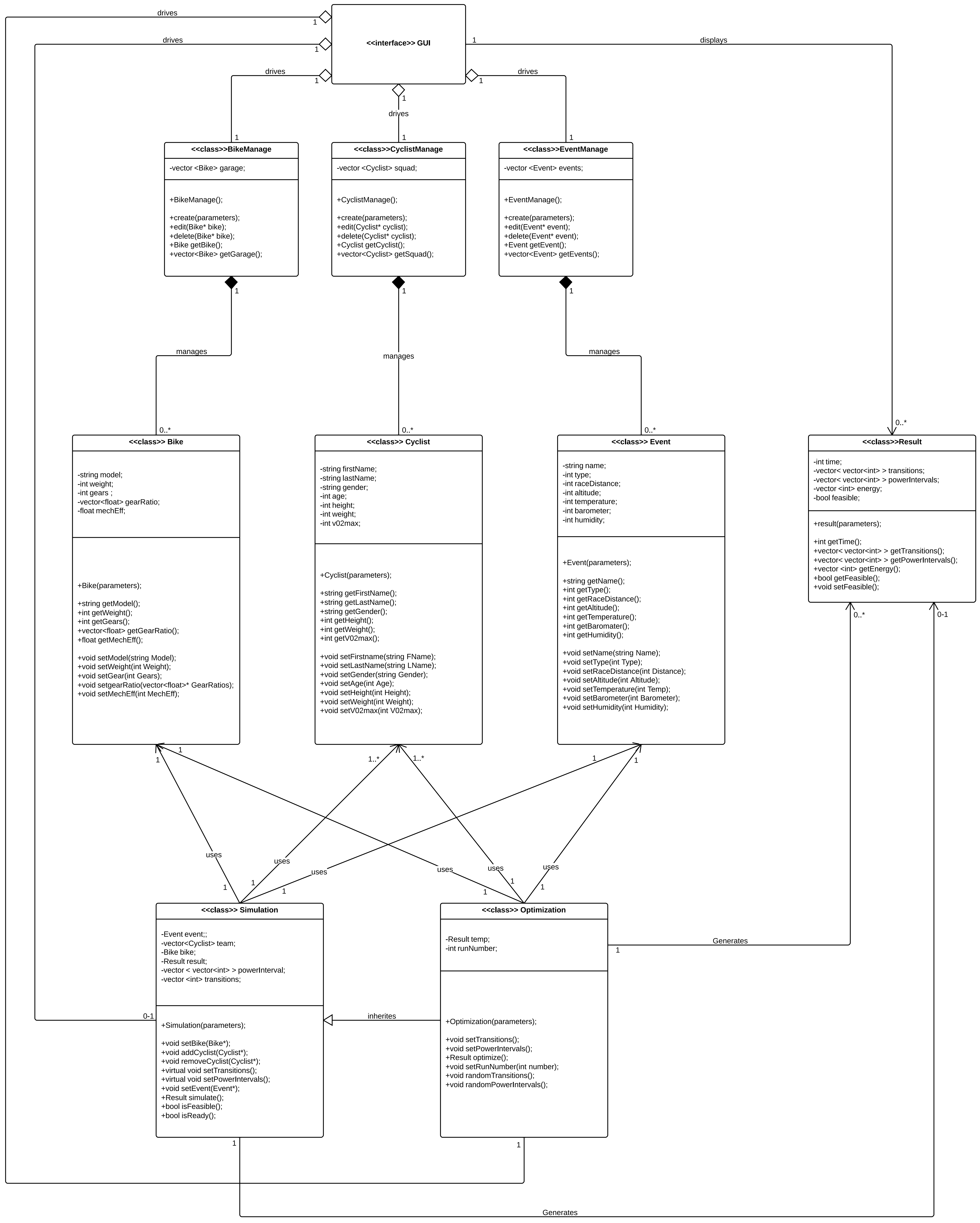
ISE - Assignment 2
Class Diagram

No Name Team:
Xiang Guo a1077337
Patrick Mann a1646630



```
<<interface>> GUI
```

```
<<class>>BikeManage
-vector <Bike> garage;

+BikeManage();

+create(parameters);
+edit(Bike* bike);
+delete(Bike* bike);
+Bike getBike();
+vector<Bike> getGarage();
```

```
<<class>>CyclistManage
-vector <Cyclist> squad;

+CyclistManage();

+create(parameters);
+edit(Cyclist* cyclist);
+delete(Cyclist* cyclist);
+Cyclist getCyclist();
+vector<Cyclist> getSquad();
```

```
<<class>>EventManage
-vector <Event> events;

+EventManage();

+create(parameters);
+edit(Event* event);
+delete(Event* event);
+Event getEvent();
+vector<Event> getEvents();
```

```
<<class>> Bike
-string model;
-int weight;
-int gears ;
-vector<float> gearRatio;
-float mechEff;

+Bike(parameters);

+string getModel();
+int getWeight();
+int getGears();
+vector<float> getGearRatio();
+float getMechEff();

+void setModel(string Model);
+void setWeight(int Weight);
+void setGear(int Gears);
+void setgearRatio(vector<float>* GearRatios);
+void setMechEff(int MechEff);
```

```
<<class>> Cyclist
-string firstName;
-string lastName;
-string gender;
-int age;
-int height;
-int weight;
-int v02max;

+Cyclist(parameters);

+string getFirstName();
+string getLastName();
+string getGender();
+int getHeight();
+int getWeight();
+int getV02max();

+void setFirstname(string FName);
+void setLastName(string LName);
+void setGender(string Gender);
+void setAge(int Age);
+void setHeight(int Height);
+void setWeight(int Weight);
+void setV02max(int V02max);
```

```
<<class>> Event
-string name;
-int type;
-int raceDistance;
-int altitude;
-int temperature;
-int barometer;
-int humidity;

+Event(parameters);

+string getName();
+int getType();
+int getRaceDistance();
+int getAltitude();
+int getTemperature();
+int getBaromater();
+int getHumidity();

+void setName(string Name);
+void setType(int Type);
+void setRaceDistance(int Distance);
+void setAltitude(int Altitude);
+void setTemperature(int Temp);
+void setBarometer(int Barometer);
+void setHumidity(int Humidity);
```

```
<<class>>Result
-int time;
-vector< vector<int> > transitions;
-vector< vector<int> > powerIntervals;
-vector <int> energy;
-bool feasible;

+result(parameters);

+int getTime();
+vector< vector<int> > getTransitions();
+vector< vector<int> > getPowerIntervals();
+vector <int> getEnergy();
+bool getFeasible();
+void setFeasible();
```

```
<<class>> Simulation
-Event event;;
-vector<Cyclist> team;
-Bike bike;
-Result result;
-vector < vector<int> > powerInterval;
-vector <int> transitions;

+Simulation(parameters);

+void setBike(Bike*);
+void addCyclist(Cyclist*);
+void removeCyclist(Cyclist*);
+virtual void setTransitions();
+virtual void setPowerIntervals();
+void setEvent(Event*);
+Result simulate();
+bool isFeasible();
+bool isReady();
```

```
<<class>> Optimization
-Result temp;
-int runNumber;

+Optimization(parameters);

+void setTransitions();
+void setPowerIntervals();
+Result optimize();
+void setRunNumber(int number);
+void randomTransitions();
+void randomPowerIntervals();
```

Overall Assumptions or design decisions.

The distances for each event will be manually entered during event creation since the tracks can vary between 250 and 330 meters.

Number of cyclists in each team is checked by Simulate(), if number is more or less than regulation bool isReady() is set to false and simulation will not run. The GUI should then indicate that a parameter is incorrect.

Strategy consists of Transitions and Power Intervals. These are entered in the GUI then stored in Simulation(), transitions as a vector of integers and power intervals as a 2d vector. For optimization(), a range is entered for each and a random number generator, bounded by the ranges specified will be used to set the transition and interval times for each simulation.

isReady() in Simulation() will check to make sure that only femal riders will participate in women's events. Same for male riders and male events. It will check also that only a single rider is entered for individual events. And make sure that a team event is entered by a team no less than 2 and no larger than 9.

Simulation() or a single run inside optimization will generate a result. isFeasible() will try to determine whether a strategy is feasible or not by comparing each Cyclist's maximum energy potential against the energy used in the results. Max energy potential is calculated using age, gender, weight, height and v02max of each cyclist. isFeasible() will check to make sure that the required amount of riders finished(did not exceed max energy) the race.

Inside the optimization(), each run stores results in a temporary Result object. It is then checked for feasibility. If not feasible it is discared, if ture then time of temporary result is compared to previously stored one. If new result is faster and feasible, it overrides the previous best, else it is discarded.

```
List of classes, attributes and methods

Bike
        +Bike(string Model, int Weight, int Gears, vector<float>* gearRatio, float mechEff);
        -string model;                  /* model of bike */
        -int weight;                    /* weight of bike - grams should be sufficient*/
        -int gears ;                    /* number of gears on bike, 1 for track, up to 20 for road */
        -vector<float> gearRatio;       /* vector of gear ratios */
        -float mechEff;                 /* mech efficiency of bike - percentage*/

        +string getModel();             /* gets */
        +int getWeight();
        +int getGears();
        +vector<float> getGearRatio();
        +float getMechEff();

        +void setModel(string Model);   /* sets */
        +void setWeight(int Weight);
        +void setGear(int Gears);
        +void setgearRatio(vector<float>* GearRatios);
        +void setMechEff(int MechEff);



Cyclist
        +Cyclist(string fName, stringlName, string Gender, int Age, int Height, int Weight, int VO2max)
        -string firstName;              /* Cyclist first name */
        -string lastName;               /* Cyclist last name */
        -string gender;                 /* Cyclist gender*/
        -int age;                       /* age in years */
        -int height;                    /* Cyclist height in cm */
        -int weight;                    /* Cyclist weight in kg */
        -int vO2max;                    /* Cyclist maximal oxygen consumption */

        +string getFirstName();         /* gets */
        +string getLastName();
        +string getGender();
        +int getHeight();
        +int getWeight();
        +int getVO2max();

        +void setFirstname(string FName);       /* sets */
        +void setLastName(string LName);
        +void setGender(string Gender);
        +void setAge(int Age);
        +void setHeight(int Height);
        +void setWeight(int Weight);
        +void setVO2max(int VO2max);



Event
        +Event(string Name, int Type, int RaceDist, int Alt, int Temp, int Baro, int Humidity, int Riders);
        -string name;                   /* Name of event */
        -int type;                      /* different numbers represent different event type, eg 0 for track individual,
                                           1 for track team, 2 for road individual, 3 for road team */
        -int raceDistance;              /* total race distance */
        -int altitude;                  /* altitude in metres */
        -int temperature;               /* temperature in degrees Celsius */
        -int barometer;                 /* barometric pressure in hPa */
        -int humidity;                  /* humidity as percentage */

        +string getName();      /* gets */
        +int getType();
        +int getRaceDistance();
        +int getAltitude();
        +int getTemperature();
        +int getBarometer();
        +int getHumidity();

        +void setName(string Name);     /* sets */
        +void setType(int Type);
        +void setRaceDistance(int Distance);
        +void setAltitude(int Altitude);
        +void setTemperature(int Temp);
        +void setBarometer(int Barometer);
        +void setHumidity(int Humidity);



BikeManage                      /* this class is used by the GUI to create of modify or delete bikes. */
        +BikeManage();
        +create(string Model, int Weight, int Gears, vector<float>* gearRatio, float mechEff);
                                /* Pushes new bike on to Garage vector */
        +edit(Bike* bike);
        +delete(Bike* bike);    /* finds and deletes a bike from Garage vector */
        -vector <bike> Garage;
        +Bike getBike();
        +vector<Bike> getGarage();



cyclistManage                   /* this class is used by the GUI to create of modify or delete cyclists. */
        +cyclistManage()
        +create(string fName, stringlName, string Gender, int Age, int Height, int Weight, int VO2max);
        +edit(Cyclist* cyclist);
        +delete(Cyclist* cyclist);
        -vector <Cyclist> squad;
        +Cyclist getCyclist();
        +vector<Cyclist> getSquad();



eventManage                     /* this class is used by the GUI to create of modify or delete Events. */
        +eventManage();
        +create(string Name, int Type, int RaceDist, int Alt, int Temp, int Baro, int Humidity, int Riders);
        +edit(Event* event);
        +delete(Event* event);
        -vector <Event> events;
        +Event getEvent();
        +vector<Event> getEvents();


Simulation
        +Simulation();
        -Event event;                   /* event to simulate */
        -vector<Cyclist> team;          /* team of cyclists */
        -Bike bike;                     /* Assuming that the entire team rides the same model of bike */
        -Result result;                 /* result object can be used by the GUI to display results to screen */
        -vector < vector<int> > powerInterval; /* 2 dimensional vector to store power interval info */
        -vector <int> transitions;      /* transition timing information */

        +void setBike(Bike*);           /* selects bike from Garage */
        +void addCyclist(Cyclist*);     /* add Cyclist to team from vector squad */
        +void removeCyclist(Cyclist*);  /* remove cyclist from team */
        +virtual void setTransitions();         /* GUI options determines what transition timing will be used */
        +virtual void setPowerIntervals();      /* GUI options determines what power intervals will be used */
        +void setEvent(Event*);         /* selects event from vector events */
        +Result simulate();             /* checks at all needed fields are filled, then calls run(), stores info in to result */
        +bool isReady();                /* Bool is used by simulate() to determine if all required parameters are entered, if
                                           bool is true then simualte will call run() */
        +bool isFeasible()              /* called by simulate() to see if any riders exceeded his/her energy potential using this strategy
                                           energy potential can be calculated using age, gender, weight, and VO2max. If any rider exceeded
                                           his/her max potention then strategy is not feasible. */


Optimization()                          /* inherit from Simulation */
        +Optimization();
        -Result temp;
        -int runNumber;                 /* number of times optimize() will call run() with different values for transitions and powerIntervals */
        +void setTransitions();         /* overloaded function that gets a range of values instead of a single value from GUI */
        +void setPowerIntervals();      /* overloaded function that gets a range of values instead of a single value from GUI */
        +Result optimize();             /* instead of simulate(), randomly generate values for transitions and powerIntervals bounded by range set
                                           from setTransitions() and setPowerInervals(), calls run() stores output in temp and compare to result
                                           if temp.getTime() < result.getTime() then result = temp */
        +void setRunNumber(int number); /* sets numbers of times to call run() */
        +void randomTransitions();      /* randomly generate values from range of values declared from setTransitions(); */
        +void randomPowerIntervals();   /* randomly generate values from range of values declared from setPowerIntervals(); */

Result
        +result(int Time, vector < vector<int> >* Transisitions, vector < vector<int> >* PowerIntervals, vector<int>* Energy, bool Feasible);
```

```
-int time;                              /* time for cyclists in simulation to complete event */
-vector< vector<int> > transitions;     /* 2 dimensional transition vector that was used in the simulation */
-vector< vector<int> > powerIntervals;  /* 2 dimensional powerInterval vector that was used in the simulation */
-vector <int> energy;                   /* how much energy used by each rider as number of calories */
-bool feasible;                         /* used by isFeasible() to indicate whether results is feasible */

+int getTime();                         /* gets */
+vector< vector<int> > getTransitions();
+vector< vector<int> > getPowerIntervals();
+vector <int> getEnergy();
+bool getFeasible();
+void setFeasible();
```