COS 214 Project - Prancing Ponies

Generated by Doxygen 1.8.20

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Aggressive Class Reference

Concrete Strategy Participant of the Strategy design pattern.

`#include <Aggressive.h>`

Inheritance diagram for Aggressive:

Public Member Functions

- Aggressive ()
- ~Aggressive ()

  Constructor.
- string execute ()

  Destructor.
- string type ()

4.1.1 Detailed Description

Concrete Strategy Participant of the Strategy design pattern.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Aggressive()

```
Aggressive::Aggressive ( )
```

4.1.2.2 ~Aggressive()

```
Aggressive::~Aggressive ( )
```

Constructor.

4.1.3 Member Function Documentation

4.1.3.1 execute()

```
string Aggressive::execute ( )  [virtual]
```

Destructor.

Execute the strategy of the way the driver wants to race.

Returns

string that gives information about the tires given the strategy that is used.

Implements Strategy.

4.1.3.2 type()

```
string Aggressive::type ( )  [virtual]
```

Returns the way the driver wants to race

Returns

string that displays the drivers strategy.

Implements Strategy.

The documentation for this class was generated from the following files:

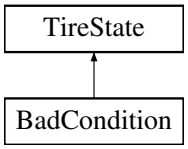
- Strategy/Aggressive.h
- Strategy/Aggressive.cpp

4.2 BadCondition Class Reference

The concrete state participant of the State design Pattern.

`#include <BadCondition.h>`

Inheritance diagram for BadCondition:



Public Member Functions

- BadCondition ()
- ∼BadCondition ()

  constructor
- bool handle (Tire ∗tire)

  destructor
- void changeTireState (Tire ∗tire)

  method to check if you should pit stop or not
- TireState ∗ clone ()

  method to change tire state

4.2.1 Detailed Description

The concrete state participant of the State design Pattern.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BadCondition()

```
BadCondition::BadCondition ( )
```

4.2.2.2 ∼BadCondition()

```
BadCondition::∼BadCondition ( )
```

constructor

4.2.3 Member Function Documentation

4.2.3.1 changeTireState()

```
void BadCondition::changeTireState (
            Tire * tire )  [virtual]
```

method to check if you should pit stop or not

Implements TireState.

4.2.3.2 clone()

```
TireState * BadCondition::clone ( )  [virtual]
```

method to change tire state

Implements TireState.

4.2.3.3 handle()

```
bool BadCondition::handle (
            Tire * tire )  [virtual]
```

destructor

Implements TireState.

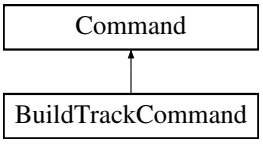
The documentation for this class was generated from the following files:

- TireState/BadCondition.h
- TireState/BadCondition.cpp

4.3 BuildTrackCommand Class Reference

```
#include <BuildTrackCommand.h>
```

Inheritance diagram for BuildTrackCommand:



Public Member Functions

- BuildTrackCommand ()
- BuildTrackCommand (string, int)

  constructor
- ~BuildTrackCommand ()
- void execute ()

  destructor
- ConcreteTrack * getTrack ()
- TrackBuilder * getTrackBuilder ()

4.3.1 Constructor & Destructor Documentation

4.3.1.1 BuildTrackCommand() [1/2]

```
BuildTrackCommand::BuildTrackCommand ( )
```

4.3.1.2 BuildTrackCommand() [2/2]

```
BuildTrackCommand::BuildTrackCommand (
            string n,
            int l )
```

constructor

Constructor

Parameters

location	
laps	

4.3.1.3 ~BuildTrackCommand()

```
BuildTrackCommand::~BuildTrackCommand ( )
```

4.3.2 Member Function Documentation

4.3.2.1 execute()

```
void BuildTrackCommand::execute ( )  [virtual]
```

destructor

Function that executes all the commands

Implements Command.

4.3.2.2 getTrack()

```
ConcreteTrack * BuildTrackCommand::getTrack ( )
```

Returns the track

Returns

ConcreteTrack

4.3.2.3 getTrackBuilder()

```
TrackBuilder * BuildTrackCommand::getTrackBuilder ( )
```

Returns the trackbuilder

Returns

TrackBuilder

The documentation for this class was generated from the following files:

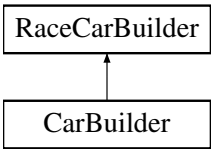
- Command/BuildTrackCommand.h
- Command/BuildTrackCommand.cpp

4.4 CarBuilder Class Reference

this class is the Concrete Builder in the Builder design Pattern

```
#include <CarBuilder.h>
```

Inheritance diagram for CarBuilder:



Public Member Functions

- CarBuilder ()
  Construct a new Car Builder object.
- ∼CarBuilder ()
  constructor
- void addChassis ()
  destructor
- void addSuspension ()
  add the suspension to the car
- void addWing ()
  add the wing to the car
- void addHub ()
  add the hub to the car
- void addEngine ()
  add the engine to the car
- void addTire (string compound)
  add the tire to the car and defines starting tire compound
- RaceCar ∗ getCar ()
  Get the Car object.
- CarPart ∗ getCarPart ()
  Get the Car Part object.

4.4.1 Detailed Description

this class is the Concrete Builder in the Builder design Pattern

this class is the Concrete Builder in the Builder design Pattern

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CarBuilder()

```
CarBuilder::CarBuilder ( )
```

Construct a new Car Builder object.

4.4.2.2 ∼CarBuilder()

```
CarBuilder::∼CarBuilder ( )
```

constructor

Destroy the Car Builder object

4.4.3 Member Function Documentation

4.4.3.1 addChassis()

```
void CarBuilder::addChassis ( )  [virtual]
```

destructor

add the chassis to car

Implements RaceCarBuilder.

4.4.3.2 addEngine()

```
void CarBuilder::addEngine ( )  [virtual]
```

add the engine to the car

Implements RaceCarBuilder.

4.4.3.3 addHub()

```
void CarBuilder::addHub ( )  [virtual]
```

add the hub to the car

Implements RaceCarBuilder.

4.4.3.4 addSuspension()

```
void CarBuilder::addSuspension ( )  [virtual]
```

add the suspension to the car

Implements RaceCarBuilder.

4.4.3.5 addTire()

```
void CarBuilder::addTire (
            string compound )  [virtual]
```

add the tire to the car and defines starting tire compound

Parameters

compound	

Implements RaceCarBuilder.

4.4.3.6 addWing()

```
void CarBuilder::addWing ( )  [virtual]
```

add the wing to the car

Implements RaceCarBuilder.

4.4.3.7 getCar()

```
RaceCar * CarBuilder::getCar ( )  [virtual]
```

Get the Car object.

Returns

RaceCar*

Implements RaceCarBuilder.

4.4.3.8 getCarPart()

```
CarPart * CarBuilder::getCarPart ( )
```

Get the Car Part object.

Returns

CarPart*

The documentation for this class was generated from the following files:

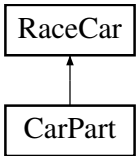
- CarComposite/CarBuilder.h
- CarComposite/CarBuilder.cpp

4.5 CarPart Class Reference

the composite participant of the Composite Design Pattern

`#include <CarPart.h>`

Inheritance diagram for CarPart:



Public Member Functions

- CarPart ()
  Construct a new Car Part object.
- CarPart (CarPart &)
  Construct a new Car Part object.
- ~CarPart ()
  Destroy the Car Part object.
- RaceCar * clone ()
  Creates a clone of CarPart and return it as RaceCar object.
- void addPart (RaceCar *part)
  add a part of the racecar to CarPart
- void removePart (RaceCar *part)
  removes a part from the CarPart list
- RaceCar * getPart ()
  Get the Part object.
- list< RaceCar * > getCarParts ()
  Get the Car Parts object.
- void lap ()
  calls the lap function of the tire part
- void degrade ()
  calls the
- void addCarTire (RaceCar *part)
  adds the tire part
- RaceCar * getCarTire ()
  Get the Car Tire object.
- int getTireGrip ()
  Get the Tire Grip object.
- string getName ()
  Get the driverName object.
- void setName (string name)
  Set the driverName object.
- int getPoints ()
  Get the Points object.
- void setPoints (int points)
  Set the Points object.
- void setPrint (bool shouldItPrint)
  Set the Print object.
- bool getPrint ()
  Get the Print object.

Protected Attributes

- list< RaceCar * > parts
- RaceCar * tire

a list of all the Race Car parts

Additional Inherited Members

4.5.1 Detailed Description

the composite participant of the Composite Design Pattern

4.5.2 Constructor & Destructor Documentation

4.5.2.1 CarPart() [1/2]

```
CarPart::CarPart ( )
```

Construct a new Car Part object.

4.5.2.2 CarPart() [2/2]

```
CarPart::CarPart (
            CarPart & oldCar )
```

Construct a new Car Part object.

4.5.2.3 ~CarPart()

```
CarPart::~CarPart ( )
```

Destroy the Car Part object.

4.5.3 Member Function Documentation

4.5.3.1 addCarTire()

```
void CarPart::addCarTire (
            RaceCar * part )
```

adds the tire part

Parameters

part	

4.5.3.2 addPart()

```
void CarPart::addPart (
            RaceCar * part )  [virtual]
```

add a part of the racecar to CarPart

Parameters

part	

Reimplemented from RaceCar.

4.5.3.3 clone()

```
RaceCar * CarPart::clone ( )  [virtual]
```

Creates a clone of CarPart and return it as RaceCar object.

Returns

RaceCar*

Implements RaceCar.

4.5.3.4 degrade()

```
void CarPart::degrade ( )  [virtual]
```

calls the

Reimplemented from RaceCar.

4.5.3.5 getCarParts()

```
list< RaceCar * > CarPart::getCarParts ( )  [virtual]
```

Get the Car Parts object.

Returns

list<RaceCar*>

Reimplemented from RaceCar.

4.5.3.6 getCarTire()

```
RaceCar * CarPart::getCarTire ( )
```

Get the Car Tire object.

Returns

RaceCar*

4.5.3.7 getName()

```
string CarPart::getName ( )  [virtual]
```

Get the driverName object.

Returns

string

Reimplemented from RaceCar.

4.5.3.8 getPart()

```
RaceCar * CarPart::getPart ( )
```

Get the Part object.

Returns

RaceCar*

4.5.3.9 getPoints()

```
int CarPart::getPoints ( )  [virtual]
```

Get the Points object.

Returns

int

Reimplemented from RaceCar.

4.5.3.10 getPrint()

```
bool CarPart::getPrint ( )
```

Get the Print object.

Returns

true
false

4.5.3.11 getTireGrip()

```
int CarPart::getTireGrip ( )  [virtual]
```

Get the Tire Grip object.

Returns

int

Reimplemented from RaceCar.

4.5.3.12 lap()

```
void CarPart::lap ( )  [virtual]
```

calls the lap function of the tire part

Reimplemented from RaceCar.

4.5.3.13 removePart()

```
void CarPart::removePart (
            RaceCar * part )
```

removes a part from the CarPart list

Parameters

part	

4.5.3.14 setName()

```
void CarPart::setName (
            string name )  [virtual]
```

Set the driverName object.

Parameters

name	

Reimplemented from RaceCar.

4.5.3.15 setPoints()

```
void CarPart::setPoints (
            int points )  [virtual]
```

Set the Points object.

Parameters

points	

Reimplemented from RaceCar.

4.5.3.16 setPrint()

```
void CarPart::setPrint (
            bool shouldItPrint )
```

Set the Print object.

Parameters

shouldItPrint	

4.5.4 Member Data Documentation

4.5.4.1 parts

```
list<RaceCar*> CarPart::parts  [protected]
```

4.5.4.2 tire

```
RaceCar* CarPart::tire  [protected]
```

a list of all the Race Car parts

The documentation for this class was generated from the following files:

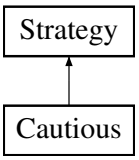
- CarComposite/CarPart.h
- CarComposite/CarPart.cpp

4.6 Cautious Class Reference

Concrete Strategy participant of the strategy design Pattern.

```
#include <Cautious.h>
```

Inheritance diagram for Cautious:



Public Member Functions

- Cautious ()
- ~Cautious ()
  constructor
- string execute ()
  destructor
- string type ()

4.6.1 Detailed Description

Concrete Strategy participant of the strategy design Pattern.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Cautious()

```
Cautious::Cautious ( )
```

4.6.2.2 ~Cautious()

```
Cautious::~Cautious ( )
```

constructor

4.6.3 Member Function Documentation

4.6.3.1 execute()

```
string Cautious::execute ( )  [virtual]
```

destructor

Execute the strategy of the way the driver wants to race.

Returns

string that gives information about the tires given the strategy that is used.

Implements Strategy.

4.6.3.2 type()

```
string Cautious::type ( )  [virtual]
```

Returns the way the driver wants to race

Returns

string that displays the drivers strategy.

Implements Strategy.

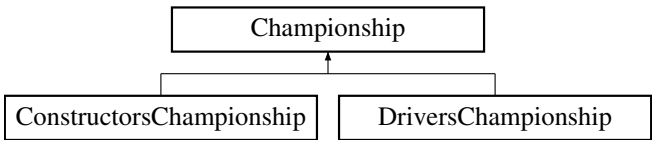
The documentation for this class was generated from the following files:

- Strategy/Cautious.h
- Strategy/Cautious.cpp

4.7 Championship Class Reference

`#include <Championship.h>`

Inheritance diagram for Championship:



Public Member Functions

- Championship (Team **, double **array, int drivers, int laps)
- virtual ~Championship ()
- void calculate ()

  destructor
- void logResults ()
- int * getTeamPoints ()
- virtual void print ()=0

Protected Attributes

- double ** arr
- int numDrivers
- int numLaps
- Team ** teams
- Results * driversResults
- TeamResults * teamResults
- int * pointList
- int pointAmount [20] = {25,18,15,12,10,8,6,4,2,1,0,0,0,0,0,0,0,0,0,0}

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Championship()

```
Championship::Championship (
            Team ** t,
            double ** array,
            int drivers,
            int laps )
```

Constructor

Parameters

Team	
array	of the times for each lap
drivers	amount
laps	amount

4.7.1.2 ~Championship()

```
Championship::~Championship ( )  [virtual]
```

4.7.2 Member Function Documentation

4.7.2.1 calculate()

```
void Championship::calculate ( )
```

destructor

Prints out the results of the race for each driver

4.7.2.2 getTeamPoints()

```
int* Championship::getTeamPoints ( )
```

Prints out the results of the race for each driver

4.7.2.3 logResults()

```
void Championship::logResults ( )
```

Prints out the results of the race for each driver

4.7.2.4 print()

```
virtual void Championship::print ( )  [pure virtual]
```

Prints out the results of the race for each driver

Implemented in DriversChampionship, and ConstructorsChampionship.

4.7.3 Member Data Documentation

4.7.3.1 arr

```
double** Championship::arr  [protected]
```

the double array of the times made by each driver for each lap

4.7.3.2 driversResults

```
Results* Championship::driversResults  [protected]
```

the drivers results for the race

4.7.3.3 numDrivers

```
int Championship::numDrivers  [protected]
```

the number of drivers driving the race

4.7.3.4 numLaps

```
int Championship::numLaps  [protected]
```

the number of laps that the race consists of

4.7.3.5 pointAmount

```
int Championship::pointAmount[20] = {25,18,15,12,10,8,6,4,2,1,0,0,0,0,0,0,0,0,0,0}  [protected]
```

array of points for each place

4.7.3.6 pointList

```
int* Championship::pointList  [protected]
```

the points list

4.7.3.7 teamResults

```
TeamResults* Championship::teamResults  [protected]
```

the team results of the race

4.7.3.8 teams

```
Team** Championship::teams  [protected]
```

List of participating teams

The documentation for this class was generated from the following files:

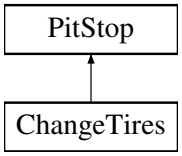
- Template/Championship.h
- Template/Championship.cpp

4.8 ChangeTires Class Reference

concrete observer participant of the Observer design pattern.

`#include <ChangeTires.h>`

Inheritance diagram for ChangeTires:



Public Member Functions

- ChangeTires (Tire *carTire)
- ~ChangeTires ()
- void update ()
  destructor

4.8.1 Detailed Description

concrete observer participant of the Observer design pattern.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ChangeTires()

```
ChangeTires::ChangeTires (
            Tire * carTire )
```

Constructor

Parameters

carTire	

4.8.2.2 ~ChangeTires()

```
ChangeTires::~ChangeTires ( )
```

4.8.3 Member Function Documentation

4.8.3.1 update()

```
void ChangeTires::update ( )  [virtual]
```

destructor

Implements PitStop.

The documentation for this class was generated from the following files:

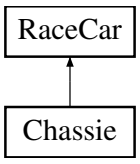
- Observer/ChangeTires.h
- Observer/ChangeTires.cpp

4.9 Chassie Class Reference

leaf participant of the Composite Design Pattern

```
#include <Chassie.h>
```

Inheritance diagram for Chassie:



Public Member Functions

- Chassie ()

  Construct a new Chassie object.
- ∼Chassie ()

  Destroy the Chassie object.
- RaceCar ∗ clone ()

  Clone function that returns a clone of the current Race Car.
- void degrade ()

  degrade the chassie

Additional Inherited Members

4.9.1 Detailed Description

leaf participant of the Composite Design Pattern

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Chassie()

```
Chassie::Chassie ( )
```

Construct a new Chassie object.

4.9.2.2 ∼Chassie()

```
Chassie::∼Chassie ( )
```

Destroy the Chassie object.

4.9.3 Member Function Documentation

4.9.3.1 clone()

```
RaceCar * Chassie::clone ( )  [virtual]
```

Clone function that returns a clone of the current Race Car.

Returns

a clone of the the Race Car

Implements RaceCar.

4.9.3.2 degrade()

```
void Chassie::degrade ( )  [virtual]
```

degrade the chassie

Reimplemented from RaceCar.

The documentation for this class was generated from the following files:

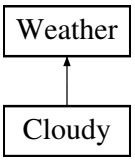
- CarComposite/Chassie.h
- CarComposite/Chassie.cpp

4.10 Cloudy Class Reference

concrete state participant of the state participant

```
#include <Cloudy.h>
```

Inheritance diagram for Cloudy:



Public Member Functions

- Cloudy ()
- virtual Weather * changeWeather ()

  constructor

4.10.1 Detailed Description

concrete state participant of the state participant

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Cloudy()

```
Cloudy::Cloudy ( )
```

4.10.3 Member Function Documentation

4.10.3.1 changeWeather()

```
Weather * Cloudy::changeWeather ( )  [virtual]
```

constructor

method to change the state of the weather

Returns

the weather state as it has changed.

Implements Weather.

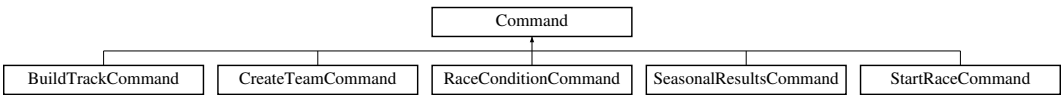
The documentation for this class was generated from the following files:

- StateWeather/Cloudy.h
- StateWeather/Cloudy.cpp

4.11 Command Class Reference

```
#include <Command.h>
```

Inheritance diagram for Command:



Public Member Functions

- virtual void execute ()=0

4.11.1 Member Function Documentation

4.11.1.1 execute()

```
virtual void Command::execute ( )  [pure virtual]
```

Function that executes all the commands

Implemented in StartRaceCommand, SeasonalResultsCommand, RaceConditionCommand, CreateTeamCommand, and BuildTrackCommand.

The documentation for this class was generated from the following file:

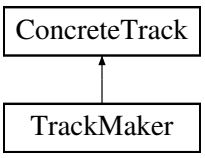
- Command/Command.h

4.12 ConcreteTrack Class Reference

The builder participant in the Builder design Pattern.

```
#include <ConcreteTrack.h>
```

Inheritance diagram for ConcreteTrack:



Public Member Functions

- ConcreteTrack ()
- virtual void addNinetyDegree (int)=0

  Constructor.
- virtual void addStraight (int)=0
- virtual void addHairpin (int)=0
- virtual void addS_section (int)=0
- virtual void addSlightTurn (int)=0
- virtual int getNumSections ()=0
- virtual vector< TrackSection > getTrack ()=0
- virtual void showTrack ()=0

4.12.1 Detailed Description

The builder participant in the Builder design Pattern.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 ConcreteTrack()

```
ConcreteTrack::ConcreteTrack ( )
```

4.12.3 Member Function Documentation

4.12.3.1 addHairpin()

```
virtual void ConcreteTrack::addHairpin (
            int  )  [pure virtual]
```

Implemented in TrackMaker.

4.12.3.2 addNinetyDegree()

```
virtual void ConcreteTrack::addNinetyDegree (
            int  )  [pure virtual]
```

Constructor.

interface for concrete builder

Implemented in TrackMaker.

4.12.3.3 addS_section()

```
virtual void ConcreteTrack::addS_section (
            int  )  [pure virtual]
```

Implemented in TrackMaker.

4.12.3.4 addSlightTurn()

```
virtual void ConcreteTrack::addSlightTurn (
            int  )  [pure virtual]
```

Implemented in TrackMaker.

4.12.3.5 addStraight()

```
virtual void ConcreteTrack::addStraight (
            int  )  [pure virtual]
```

Implemented in TrackMaker.

4.12.3.6 getNumSections()

```
virtual int ConcreteTrack::getNumSections ( )  [pure virtual]
```

Implemented in TrackMaker.

4.12.3.7 getTrack()

```
virtual vector<TrackSection> ConcreteTrack::getTrack ( )  [pure virtual]
```

Implemented in TrackMaker.

4.12.3.8 showTrack()

```
virtual void ConcreteTrack::showTrack ( )  [pure virtual]
```

Implemented in TrackMaker.

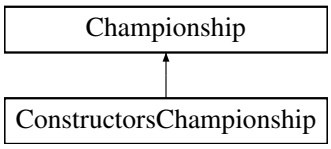
The documentation for this class was generated from the following files:

- Builder/ConcreteTrack.h
- Builder/ConcreteTrack.cpp

4.13 ConstructorsChampionship Class Reference

`#include <ConstructorsChampionship.h>`

Inheritance diagram for ConstructorsChampionship:



Public Member Functions

- ConstructorsChampionship (Team **t, double **array, int drivers, int laps)
- ~ConstructorsChampionship ()
- void print ()

Additional Inherited Members

4.13.1 Constructor & Destructor Documentation

4.13.1.1 ConstructorsChampionship()

```
ConstructorsChampionship::ConstructorsChampionship (
            Team ** t,
            double ** array,
            int drivers,
            int laps )
```

Constructor

Parameters

Team	
array	of the times for each lap
drivers	amount
laps	amount

4.13.1.2 ~ConstructorsChampionship()

```
ConstructorsChampionship::~ConstructorsChampionship ( )
```

4.13.2 Member Function Documentation

4.13.2.1 print()

```
void ConstructorsChampionship::print ( )  [virtual]
```

Prints out the results of the race for each driver

Implements Championship.

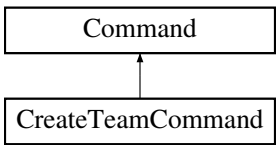
The documentation for this class was generated from the following files:

- Template/ConstructorsChampionship.h
- Template/ConstructorsChampionship.cpp

4.14 CreateTeamCommand Class Reference

```
#include <CreateTeamCommand.h>
```

Inheritance diagram for CreateTeamCommand:



Public Member Functions

- CreateTeamCommand ()
- ~CreateTeamCommand ()
  constructor
- void execute ()
  destructor
- Team ** getTeams ()
- void restoreTeams ()

4.14.1 Constructor & Destructor Documentation

4.14.1.1 CreateTeamCommand()

```
CreateTeamCommand::CreateTeamCommand ( )
```

4.14.1.2 ~CreateTeamCommand()

```
CreateTeamCommand::~CreateTeamCommand ( )
```

constructor

4.14.2 Member Function Documentation

4.14.2.1 execute()

```
void CreateTeamCommand::execute ( )  [virtual]
```

destructor

Function that executes all the commands

Implements Command.

4.14.2.2 getTeams()

```
Team ** CreateTeamCommand::getTeams ( )
```

Returns the teams

Returns

Team

4.14.2.3 restoreTeams()

```
void CreateTeamCommand::restoreTeams ( )
```

Sets the teams

Parameters

teams	taking part

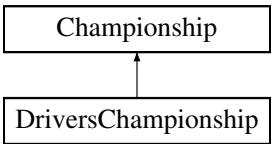
The documentation for this class was generated from the following files:

- Command/CreateTeamCommand.h
- Command/CreateTeamCommand.cpp

4.15 DriversChampionship Class Reference

`#include <DriversChampionship.h>`

Inheritance diagram for DriversChampionship:



Public Member Functions

- DriversChampionship (Team **t, double **arr, int drivers, int laps)
- ~DriversChampionship ()
- void print ()

  destructor

Additional Inherited Members

4.15.1 Constructor & Destructor Documentation

4.15.1.1 DriversChampionship()

```
DriversChampionship::DriversChampionship (
            Team ** t,
            double ** arr,
            int drivers,
            int laps )
```

Constructor

Parameters

Team	
array	of the times for each lap
drivers	amount
laps	amount

4.15.1.2 ~DriversChampionship()

```
DriversChampionship::~DriversChampionship ( )
```

4.15.2 Member Function Documentation

4.15.2.1 print()

```
void DriversChampionship::print ( )  [virtual]
```

destructor

Prints out the results of the race for each driver

Implements Championship.

The documentation for this class was generated from the following files:

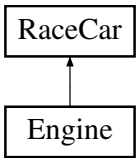
- Template/DriversChampionship.h
- Template/DriversChampionship.cpp

4.16 Engine Class Reference

leaf participant of the Composite design Pattern

```
#include <Engine.h>
```

Inheritance diagram for Engine:



Public Member Functions

- Engine ()

  Construct a new Engine object.
- ~Engine ()

  Destroy the Engine object.
- void degrade ()

  degrade the engine
- virtual RaceCar * clone ()

  Clone function that returns a clone of the current Race Car.

Additional Inherited Members

4.16.1 Detailed Description

leaf participant of the Composite design Pattern

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Engine()

```
Engine::Engine ( )
```

Construct a new Engine object.

4.16.2.2 ~Engine()

```
Engine::~Engine ( )
```

Destroy the Engine object.

4.16.3 Member Function Documentation

4.16.3.1 clone()

```
RaceCar * Engine::clone ( )  [virtual]
```

Clone function that returns a clone of the current Race Car.

Returns

a clone of the the Race Car

Implements RaceCar.

4.16.3.2 degrade()

```
void Engine::degrade ( )  [virtual]
```

degrade the engine

Reimplemented from RaceCar.

The documentation for this class was generated from the following files:

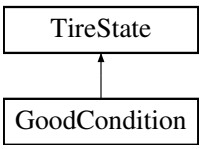
- CarComposite/Engine.h
- CarComposite/Engine.cpp

4.17 GoodCondition Class Reference

the concrete state participant of the State design Pattern

`#include <GoodCondition.h>`

Inheritance diagram for GoodCondition:



Public Member Functions

- GoodCondition ()
- ∼GoodCondition ()

  constructor
- bool handle (Tire ∗tire)

  destructor
- void changeTireState (Tire ∗tire)

  method to check if a pit stop is needed
- TireState ∗ clone ()

  method to change tire state

4.17.1 Detailed Description

the concrete state participant of the State design Pattern

4.17.2 Constructor & Destructor Documentation

4.17.2.1 GoodCondition()

`GoodCondition::GoodCondition ( )`

4.17.2.2 ∼GoodCondition()

`GoodCondition::∼GoodCondition ( )`

constructor

4.17.3 Member Function Documentation

4.17.3.1 changeTireState()

```
void GoodCondition::changeTireState (
            Tire * tire )  [virtual]
```

method to check if a pit stop is needed

Implements TireState.

4.17.3.2 clone()

```
TireState * GoodCondition::clone ( )  [virtual]
```

method to change tire state

Implements TireState.

4.17.3.3 handle()

```
bool GoodCondition::handle (
            Tire * tire )  [virtual]
```

destructor

Implements TireState.

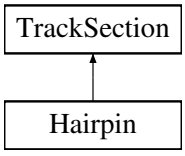
The documentation for this class was generated from the following files:

- TireState/GoodCondition.h
- TireState/GoodCondition.cpp

4.18 Hairpin Class Reference

```
#include <Hairpin.h>
```

Inheritance diagram for Hairpin:



Public Member Functions

- Hairpin (int)

  Construct a new Hairpin object.

Additional Inherited Members

4.18.1 Constructor & Destructor Documentation

4.18.1.1 Hairpin()

```
Hairpin::Hairpin (
            int d )
```

Construct a new Hairpin object.

Parameters

distance	

The documentation for this class was generated from the following files:

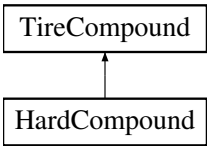
- Builder/Hairpin.h
- Builder/Hairpin.cpp

4.19 HardCompound Class Reference

concrete strategy participant of the strategy design pattern

```
#include <HardCompound.h>
```

Inheritance diagram for HardCompound:



Public Member Functions

- HardCompound ()
- ∼HardCompound ()

  constructor
- int getGrip ()

  destructor
- void setGrip (int grip)
- int getWear ()
- void setWear (int wear)
- double getRate ()

  set the wear of the tire
- TireCompound ∗ clone ()

  returns the rate at which the tire wears

Additional Inherited Members

4.19.1 Detailed Description

concrete strategy participant of the strategy design pattern

4.19.2 Constructor & Destructor Documentation

4.19.2.1 HardCompound()

```
HardCompound::HardCompound ( )
```

4.19.2.2 ∼HardCompound()

```
HardCompound::∼HardCompound ( )
```

constructor

4.19.3 Member Function Documentation

4.19.3.1 clone()

```
TireCompound * HardCompound::clone ( )  [virtual]
```

returns the rate at which the tire wears

clone the TireCompound

Returns

TireCompound clone

Implements TireCompound.

4.19.3.2 getGrip()

```
int HardCompound::getGrip ( )  [virtual]
```

destructor

get the grip of the tire

Returns

int grip of the car default value of 100

Implements TireCompound.

4.19.3.3 getRate()

```
double HardCompound::getRate ( )  [virtual]
```

set the wear of the tire

return the wear of the tire

Returns

double rate of the tires

Implements TireCompound.

4.19.3.4 getWear()

```
int HardCompound::getWear ( )  [virtual]
```

return the wear of the tire

Returns

int wear of the tire

Implements TireCompound.

4.19.3.5 setGrip()

```
void HardCompound::setGrip (
            int grip )  [virtual]
```

sets the grip of the tires

Parameters

grip	int

Implements TireCompound.

4.19.3.6 setWear()

```
void HardCompound::setWear (
            int wear )  [virtual]
```

sets the wear of the tire

Parameters

wear	int

Implements TireCompound.

The documentation for this class was generated from the following files:

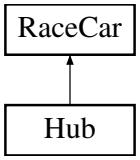
- TireCompoundStrategy/HardCompound.h
- TireCompoundStrategy/HardCompound.cpp

4.20 Hub Class Reference

leaf participant of the Composite Design pattern

`#include <Hub.h>`

Inheritance diagram for Hub:



Public Member Functions

- Hub ()

  Construct a new Hub object.
- ~Hub ()

  Destroy the Hub object.
- void degrade ()

  degrade the hub
- RaceCar * clone ()

  Clone function that returns a clone of the current Race Car.

Additional Inherited Members

4.20.1 Detailed Description

leaf participant of the Composite Design pattern

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Hub()

```
Hub::Hub ( )
```

Construct a new Hub object.

4.20.2.2 ~Hub()

```
Hub::~Hub ( )
```

Destroy the Hub object.

4.20.3 Member Function Documentation

4.20.3.1 clone()

```
RaceCar * Hub::clone ( )  [virtual]
```

Clone function that returns a clone of the current Race Car.

Returns

a clone of the the Race Car

Implements RaceCar.

4.20.3.2 degrade()

```
void Hub::degrade ( )  [virtual]
```

degrade the hub

Reimplemented from RaceCar.

The documentation for this class was generated from the following files:

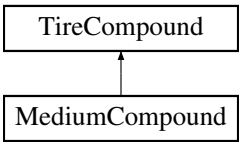
- CarComposite/Hub.h
- CarComposite/Hub.cpp

4.21 MediumCompound Class Reference

concrete strategy participant from the strategy design pattern

```
#include <MediumCompound.h>
```

Inheritance diagram for MediumCompound:



Public Member Functions

- MediumCompound ()
- ~MediumCompound ()
  constructor
- int getGrip ()
- void setGrip (int grip)
- int getWear ()
- void setWear (int wear)
- double getRate ()
- TireCompound * clone ()

Additional Inherited Members

4.21.1 Detailed Description

concrete strategy participant from the strategy design pattern

4.21.2 Constructor & Destructor Documentation

4.21.2.1 MediumCompound()

```
MediumCompound::MediumCompound ( )
```

4.21.2.2 ~MediumCompound()

```
MediumCompound::~MediumCompound ( )
```

constructor

4.21.3 Member Function Documentation

4.21.3.1 clone()

```
TireCompound * MediumCompound::clone ( )  [virtual]
```

clone the TireCompound

Returns

TireCompound clone

Implements TireCompound.

4.21.3.2 getGrip()

```
int MediumCompound::getGrip ( )  [virtual]
```

get the grip of the tire

Returns

int grip of the car default value of 100

Implements TireCompound.

4.21.3.3 getRate()

```
double MediumCompound::getRate ( )  [virtual]
```

return the wear of the tire

Returns

double rate of the tires

Implements TireCompound.

4.21.3.4 getWear()

```
int MediumCompound::getWear ( )  [virtual]
```

return the wear of the tire

Returns

int wear of the tire

Implements TireCompound.

4.21.3.5 setGrip()

```
void MediumCompound::setGrip (
            int grip )  [virtual]
```

sets the grip of the tires

Parameters

grip	int

Implements TireCompound.

4.21.3.6 setWear()

```
void MediumCompound::setWear (
            int wear )  [virtual]
```

sets the wear of the tire

Parameters

wear	int

Implements TireCompound.

The documentation for this class was generated from the following files:

- TireCompoundStrategy/MediumCompound.h
- TireCompoundStrategy/MediumCompound.cpp

4.22 Memento Class Reference

the memento participant of the memento design pattern

`#include <Memento.h>`

Public Member Functions

- Memento (Team ∗team)
- Memento (RaceCar ∗carOne, RaceCar ∗carTwo)
- ∼Memento ()
- TeamState ∗ getState ()

  destructor
- void setState (Team ∗team)
- void setState (RaceCar ∗carOne, RaceCar ∗carTwo)

4.22.1 Detailed Description

the memento participant of the memento design pattern

4.22.2 Constructor & Destructor Documentation

4.22.2.1 Memento() [1/2]

```
Memento::Memento (
            Team * team )
```

Constructor to set the team

Parameters

Team	object to store in the memento

4.22.2.2 Memento() [2/2]

```
Memento::Memento (
            RaceCar * carOne,
            RaceCar * carTwo )
```

Constructor that sets the cars of the team

Parameters

carOne	the first car of the team
carTwo	the second car of the team

4.22.2.3 ~Memento()

```
Memento::~Memento ( )
```

4.22.3 Member Function Documentation

4.22.3.1 getState()

```
TeamState * Memento::getState ( )
```

destructor

Returns the state of the team

Returns

TeamState pointer that is stored

4.22.3.2 setState() [1/2]

```
void Memento::setState (
            RaceCar * carOne,
            RaceCar * carTwo )
```

Sets the cars in the teams

Parameters

carOne	the first car of the team
carTwo	the second car of the team

4.22.3.3 setState() [2/2]

```
void Memento::setState (
            Team * team )
```

Sets the team to store

Parameters

Team	object to store in the memento

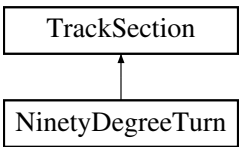
The documentation for this class was generated from the following files:

- Memento/Memento.h
- Memento/Memento.cpp

4.23 NinetyDegreeTurn Class Reference

`#include <NinetyDegreeTurn.h>`

Inheritance diagram for NinetyDegreeTurn:



Public Member Functions

- NinetyDegreeTurn (int)

  Construct a new Ninety Degree Turn object.

Additional Inherited Members

4.23.1 Constructor & Destructor Documentation

4.23.1.1 NinetyDegreeTurn()

```
NinetyDegreeTurn::NinetyDegreeTurn (
            int d )
```

Construct a new Ninety Degree Turn object.

Parameters

distance	

The documentation for this class was generated from the following files:

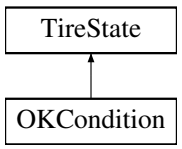
- Builder/NinetyDegreeTurn.h
- Builder/NinetyDegreeTurn.cpp

4.24 OKCondition Class Reference

The concrete state participant of the State design Pattern.

`#include <OKCondition.h>`

Inheritance diagram for OKCondition:



Public Member Functions

- OKCondition ()
- ∼OKCondition ()

  constructor
- bool handle (Tire ∗tire)

  destructor
- void changeTireState (Tire ∗tire)

  method to check if you should pit stop or not
- TireState ∗ clone ()

  method to change the state of the tires

4.24.1 Detailed Description

The concrete state participant of the State design Pattern.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 OKCondition()

```
OKCondition::OKCondition ( )
```

4.24.2.2 ~OKCondition()

```
OKCondition::~OKCondition ( )
```

constructor

4.24.3 Member Function Documentation

4.24.3.1 changeTireState()

```
void OKCondition::changeTireState (
            Tire * tire )  [virtual]
```

method to check if you should pit stop or not

Implements TireState.

4.24.3.2 clone()

```
TireState * OKCondition::clone ( )  [virtual]
```

method to change the state of the tires

Implements TireState.

4.24.3.3 handle()

```
bool OKCondition::handle (
            Tire * tire )  [virtual]
```

destructor

Implements TireState.

The documentation for this class was generated from the following files:

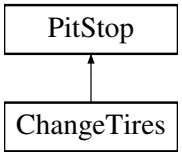
- TireState/OKCondition.h
- TireState/OKCondition.cpp

4.25 PitStop Class Reference

The observer participant of the observer design pattern.

`#include <PitStop.h>`

Inheritance diagram for PitStop:



Public Member Functions

- PitStop ()
- ~PitStop ()

  constructor
- virtual void update ()=0

  destructor

4.25.1 Detailed Description

The observer participant of the observer design pattern.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 PitStop()

`PitStop::PitStop ( )  [inline]`

4.25.2.2 ~PitStop()

`PitStop::~PitStop ( )  [inline]`

constructor

4.25.3 Member Function Documentation

4.25.3.1 update()

```
virtual void PitStop::update ( )  [pure virtual]
```

destructor

Implemented in ChangeTires.

The documentation for this class was generated from the following file:

- Observer/PitStop.h

4.26 Race Class Reference

context participant of the state design pattern

```
#include <Race.h>
```

Public Member Functions

- Race (string Location)
- ~Race ()
- void change ()
  destructor
- string getWeather ()
- void setWeather (Weather *)
  returns the weather

4.26.1 Detailed Description

context participant of the state design pattern

4.26.2 Constructor & Destructor Documentation

4.26.2.1 Race()

```
Race::Race (
            string Location )
```

Constructor that sets the location of the race

Parameters

location	of the race

4.26.2.2 ~Race()

```
Race::~Race ( )
```

4.26.3 Member Function Documentation

4.26.3.1 change()

```
void Race::change ( )
```

destructor

Changes the state of the weather depending on random conditions

4.26.3.2 getWeather()

```
string Race::getWeather ( )
```

get method to get the state of the weather

Returns

the weather

4.26.3.3 setWeather()

```
void Race::setWeather (
            Weather * w )
```

returns the weather

get method to get the state of the weather

Parameters

weather	pointer to set the weather state

The documentation for this class was generated from the following files:

- StateWeather/Race.h

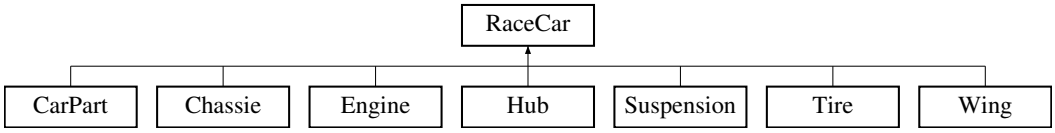
- StateWeather/Race.cpp

4.27 RaceCar Class Reference

This class is the product participant of the Builder Design pattern. Subject participant of the observer design Pattern.

`#include <RaceCar.h>`

Inheritance diagram for RaceCar:



Public Member Functions

- RaceCar ()

  Construct a new Race Car object.
- ~RaceCar ()

  Destroy the Race Car object.
- void request ()
- virtual void lap ()

  //the method to do a lap
- RaceCar * getChild ()

  Get the Child object.
- virtual void degrade ()

  virtual function of the degrade method
- virtual void addPart (RaceCar *car)

  virtual method to add parts to the race car
- virtual RaceCar * clone ()=0

  pure virtual function of the clone method that create a clone of the car
- virtual list< RaceCar * > getCarParts ()

  Get the Car Parts object.
- void addPitcrew (PitStop *pitcrew)

  attach observer
- void removePitCrew ()

  detach observer
- void notify ()

  notify observer
- Strategy * getStrategy () const

  Get the Strategy object.
- void setStrategy (Strategy *strat)

  Set the Strategy object.
- PitStop * getPitStops () const

  Get the Pit Stops object.
- void setPitStop (PitStop *pitstop)

  Set the Pit Stop object.

- bool carPitted ()
  check if the car has pitted
- bool strategyChanged ()
  check if the strategy has cha
- virtual int getTireGrip ()
  Get the Tire Grip object.
- int getCarTireGrip ()
  Get the Car Tire Grip object.
- virtual string getName ()
  Get the driverName object.
- virtual void setName (string name)
  Set the driverName object.
- string getDriverName ()
  Get the Driver Name object.
- void setDriverName (string name)
  Set the Driver Name object.
- virtual int getPoints ()
  Get the Points object.
- virtual void setPoints (int points)
  Set the Points object.
- int getCarPoints ()
  Get the Car Points object.
- void setCarPoints (int points)
  Set the Car Points object.

Public Attributes

- bool print = true

Protected Attributes

- Strategy * strategy = nullptr
  should it print
- PitStop * pitCrew = nullptr
  strategy
- int points = 0
  observer
- string driverName = ""
  driver points
- int tireGrip = 5
  driver name
- TireCompound * compound = nullptr
  the grip of the tire
- bool hasPitted = false
  the tire compound strategy
- bool changedStrat = false
  has it pitted
- string oldStrat = ""
  did the strategy change
- string newStrat = ""
  name of old strat before change

4.27.1 Detailed Description

This class is the product participant of the Builder Design pattern. Subject participant of the observer design Pattern.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 RaceCar()

```
RaceCar::RaceCar ( )
```

Construct a new Race Car object.

4.27.2.2 ~RaceCar()

```
RaceCar::~RaceCar ( )
```

Destroy the Race Car object.

4.27.3 Member Function Documentation

4.27.3.1 addPart()

```
void RaceCar::addPart (
            RaceCar * car )  [virtual]
```

virtual method to add parts to the race car

Parameters

car	

Reimplemented in CarPart.

4.27.3.2 addPitcrew()

```
void RaceCar::addPitcrew (
            PitStop * pitcrew )
```

attach observer

Parameters

pitcrew	

4.27.3.3 carPitted()

```
bool RaceCar::carPitted ( )
```

check if the car has pitted

Returns

true

false

4.27.3.4 clone()

```
virtual RaceCar* RaceCar::clone ( )  [pure virtual]
```

pure virtual function of the clone method that create a clone of the car

Returns

RaceCar∗

Implemented in Wing, Tire, Suspension, Hub, Engine, Chassie, and CarPart.

4.27.3.5 degrade()

```
void RaceCar::degrade ( )  [virtual]
```

virtual function of the degrade method

Reimplemented in Wing, Tire, Suspension, Hub, Engine, Chassie, and CarPart.

4.27.3.6 getCarParts()

```
list< RaceCar * > RaceCar::getCarParts ( )  [virtual]
```

Get the Car Parts object.

Returns

list<RaceCar*>

Reimplemented in CarPart.

4.27.3.7 getCarPoints()

```
int RaceCar::getCarPoints ( )
```

Get the Car Points object.

Returns

int

4.27.3.8 getCarTireGrip()

```
int RaceCar::getCarTireGrip ( )
```

Get the Car Tire Grip object.

Returns

int

4.27.3.9 getChild()

```
RaceCar * RaceCar::getChild ( )
```

Get the Child object.

Returns

RaceCar*

4.27.3.10 getDriverName()

```
string RaceCar::getDriverName ( )
```

Get the Driver Name object.

Returns

string

4.27.3.11 getName()

```
string RaceCar::getName ( )  [virtual]
```

Get the driverName object.

Returns

string

Reimplemented in CarPart.

4.27.3.12 getPitStops()

```
PitStop * RaceCar::getPitStops ( ) const
```

Get the Pit Stops object.

Returns

PitStop*

4.27.3.13 getPoints()

```
int RaceCar::getPoints ( )  [virtual]
```

Get the Points object.

Returns

int

Reimplemented in CarPart.

4.27.3.14 getStrategy()

```
Strategy * RaceCar::getStrategy ( ) const
```

Get the Strategy object.

Returns

Strategy*

4.27.3.15 getTireGrip()

```
int RaceCar::getTireGrip ( )  [virtual]
```

Get the Tire Grip object.

Returns

int

Reimplemented in CarPart.

4.27.3.16 lap()

```
void RaceCar::lap ( )  [virtual]
```

//the method to do a lap

Reimplemented in Tire, and CarPart.

4.27.3.17 notify()

```
void RaceCar::notify ( )
```

notify observer

4.27.3.18 removePitCrew()

```
void RaceCar::removePitCrew ( )
```

detach observer

4.27.3.19 request()

```
void RaceCar::request ( )
```

4.27.3.20 setCarPoints()

```
void RaceCar::setCarPoints (
            int points )
```

Set the Car Points object.

Parameters

points	

4.27.3.21 setDriverName()

```
void RaceCar::setDriverName (
            string name )
```

Set the Driver Name object.

Parameters

name	

4.27.3.22 setName()

```
void RaceCar::setName (
            string name )  [virtual]
```

Set the driverName object.

Parameters

name	

Reimplemented in CarPart.

4.27.3.23 setPitStop()

```
void RaceCar::setPitStop (
            PitStop * pitstop )
```

Set the Pit Stop object.

Parameters

pitstop	

4.27.3.24 setPoints()

```
void RaceCar::setPoints (
            int points )  [virtual]
```

Set the Points object.

Parameters

points	

Reimplemented in CarPart.

4.27.3.25 setStrategy()

```
void RaceCar::setStrategy (
            Strategy * strat )
```

Set the Strategy object.

Parameters

strat	

4.27.3.26 strategyChanged()

```
bool RaceCar::strategyChanged ( )
```

check if the strategy has cha

Returns

true

false

4.27.4 Member Data Documentation

4.27.4.1 changedStrat

```
bool RaceCar::changedStrat = false  [protected]
```

has it pitted

4.27.4.2 compound

```
TireCompound* RaceCar::compound = nullptr  [protected]
```

the grip of the tire

4.27.4.3 driverName

```
string RaceCar::driverName = ""  [protected]
```

driver points

4.27.4.4 hasPitted

```
bool RaceCar::hasPitted = false  [protected]
```

the tire compound strategy

4.27.4.5 newStrat

```
string RaceCar::newStrat = ""  [protected]
```

name of old strat before change

4.27.4.6 oldStrat

```
string RaceCar::oldStrat = ""  [protected]
```

did the strategy change

4.27.4.7 pitCrew

```
PitStop* RaceCar::pitCrew = nullptr  [protected]
```

strategy

4.27.4.8 points

```
int RaceCar::points = 0  [protected]
```

observer

4.27.4.9 print

```
bool RaceCar::print = true
```

4.27.4.10 strategy

```
Strategy* RaceCar::strategy = nullptr  [protected]
```

should it print

4.27.4.11 tireGrip

```
int RaceCar::tireGrip = 5  [protected]
```

driver name

The documentation for this class was generated from the following files:

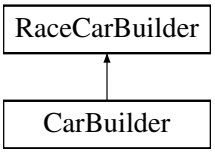
- CarComposite/RaceCar.h
- CarComposite/RaceCar.cpp

4.28 RaceCarBuilder Class Reference

The Builder participant of The BUilder Design Pattern.

```
#include <RaceCarBuilder.h>
```

Inheritance diagram for RaceCarBuilder:



Public Member Functions

- RaceCarBuilder ()
  Construct a new Race Car Builder object.
- ∼RaceCarBuilder ()
  Destroy the Race Car Builder object.
- virtual void addChassis ()=0
  add the chassis to the car
- virtual void addSuspension ()=0
  add the suspension
- virtual void addWing ()=0
  add the wing to the car
- virtual void addHub ()=0
  add the hub to the car
- virtual void addEngine ()=0
  add the engine to the car
- virtual void addTire (string)=0
  add the tire to the car
- virtual RaceCar ∗ getCar ()=0
  Get the Car object.

4.28.1 Detailed Description

The Builder participant of The BUilder Design Pattern.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 RaceCarBuilder()

```
RaceCarBuilder::RaceCarBuilder ( )
```

Construct a new Race Car Builder object.

4.28.2.2 ∼RaceCarBuilder()

```
RaceCarBuilder::∼RaceCarBuilder ( )
```

Destroy the Race Car Builder object.

4.28.3 Member Function Documentation

4.28.3.1 addChassis()

```
virtual void RaceCarBuilder::addChassis ( )  [pure virtual]
```

add the chassis to the car

Implemented in CarBuilder.

4.28.3.2 addEngine()

```
virtual void RaceCarBuilder::addEngine ( )  [pure virtual]
```

add the engine to the car

Implemented in CarBuilder.

4.28.3.3 addHub()

```
virtual void RaceCarBuilder::addHub ( )  [pure virtual]
```

add the hub to the car

Implemented in CarBuilder.

4.28.3.4 addSuspension()

```
virtual void RaceCarBuilder::addSuspension ( )  [pure virtual]
```

add the suspension

Implemented in CarBuilder.

4.28.3.5 addTire()

```
virtual void RaceCarBuilder::addTire (
            string  )  [pure virtual]
```

add the tire to the car

Implemented in CarBuilder.

4.28.3.6 addWing()

```
virtual void RaceCarBuilder::addWing ( )  [pure virtual]
```

add the wing to the car

Implemented in CarBuilder.

4.28.3.7 getCar()

```
virtual RaceCar* RaceCarBuilder::getCar ( )  [pure virtual]
```

Get the Car object.

Returns

RaceCar∗

Implemented in CarBuilder.

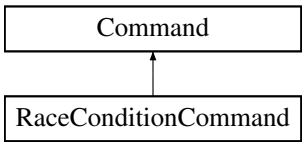
The documentation for this class was generated from the following files:

- CarComposite/RaceCarBuilder.h
- CarComposite/RaceCarBuilder.cpp

4.29 RaceConditionCommand Class Reference

```
#include <RaceConditionCommand.h>
```

Inheritance diagram for RaceConditionCommand:



Public Member Functions

- RaceConditionCommand ()
- RaceConditionCommand (string location)

  constructor
- ∼RaceConditionCommand ()
- void execute ()

  destructor
- Race ∗ getRaceWeather ()
- void setRaceWeather (Weather ∗itNeverRainOnRaceDay)

4.29.1 Constructor & Destructor Documentation

4.29.1.1 RaceConditionCommand() [1/2]

```
RaceConditionCommand::RaceConditionCommand ( )
```

4.29.1.2 RaceConditionCommand() [2/2]

```
RaceConditionCommand::RaceConditionCommand (
            string location )
```

constructor

Constructor

Parameters

location	

4.29.1.3 ~RaceConditionCommand()

```
RaceConditionCommand::~RaceConditionCommand ( )
```

4.29.2 Member Function Documentation

4.29.2.1 execute()

```
void RaceConditionCommand::execute ( )  [virtual]
```

destructor

Function that executes all the commands

Implements Command.

4.29.2.2 getRaceWeather()

```
Race * RaceConditionCommand::getRaceWeather ( )
```

get the race weather

Returns

the race weather

4.29.2.3 setRaceWeather()

```
void RaceConditionCommand::setRaceWeather (
            Weather * itNeverRainOnRaceDay )
```

Set the weather of the race

Parameters

itNeverRainOnRaceDay	

The documentation for this class was generated from the following files:

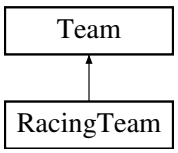
- Command/RaceConditionCommand.h
- Command/RaceConditionCommand.cpp

4.30 RacingTeam Class Reference

concrete prototype of the prototype design pattern

```
#include <RacingTeam.h>
```

Inheritance diagram for RacingTeam:



Public Member Functions

- RacingTeam ()
- RacingTeam (string tireCompound)

  constructor

- RacingTeam (RacingTeam &)
- ~RacingTeam ()
- void buildCar ()

  destructor
- void lap ()
- void setTireCompound (string tireCompound)
- RaceCar * getCarOne ()
- CarPart * getCarOnePart ()
- RaceCar * getCarTwo ()
- CarPart * getCarTwoPart ()
- void setCarOne (RaceCar *car1)
- void setCarTwo (RaceCar *car2)
- Team * clone ()
- int getTeamPoints ()
- void setTeamPoints ()
- void setTeamPoints (int points)
- string getTeamName ()
- void setTeamName (string name)
- Memento * createMemento ()
- void loadMemento (Memento *m)

Additional Inherited Members

4.30.1 Detailed Description

concrete prototype of the prototype design pattern

4.30.2 Constructor & Destructor Documentation

4.30.2.1 RacingTeam() [1/3]

```
RacingTeam::RacingTeam ( )
```

4.30.2.2 RacingTeam() [2/3]

```
RacingTeam::RacingTeam (
            string tireCompound )
```

constructor

Constrcutor That takes in the tire compound of the car(soft , medium , hard)

Parameters

string	tire compound

4.30.2.3 RacingTeam() [3/3]

```
RacingTeam::RacingTeam (
            RacingTeam & racingTeam )
```

Constrcutor That takes in the racing team

Parameters

racingTeam	tire compound

4.30.2.4 ~RacingTeam()

```
RacingTeam::~RacingTeam ( )
```

4.30.3 Member Function Documentation

4.30.3.1 buildCar()

```
void RacingTeam::buildCar ( )  [virtual]
```

destructor

Method that builds the car It adds all the elements to the car

Implements Team.

4.30.3.2 clone()

```
Team * RacingTeam::clone ( )  [virtual]
```

Abstract interface method that clones the team

Returns

Team object

Implements Team.

4.30.3.3 createMemento()

```
Memento * RacingTeam::createMemento ( )  [virtual]
```

gets the team name

Returns

memento of the

Implements Team.

4.30.3.4 getCarOne()

```
RaceCar * RacingTeam::getCarOne ( )  [virtual]
```

method that returns the first car of the team

Returns

RaceCar object of the first car of the team

Implements Team.

4.30.3.5 getCarOnePart()

```
CarPart * RacingTeam::getCarOnePart ( )  [virtual]
```

Implements Team.

4.30.3.6 getCarTwo()

```
RaceCar * RacingTeam::getCarTwo ( )  [virtual]
```

method that returns the second car of the team

Returns

RaceCar object of the second car of the team

Implements Team.

4.30.3.7 getCarTwoPart()

```
CarPart * RacingTeam::getCarTwoPart ( )  [virtual]
```

Implements Team.

4.30.3.8 getTeamName()

```
string RacingTeam::getTeamName ( )  [virtual]
```

gets the team name

Returns

string of the team name

Implements Team.

4.30.3.9 getTeamPoints()

```
int RacingTeam::getTeamPoints ( )  [virtual]
```

Abstract interface method that returns the teams points

Returns

int amount of the points the team has made

Implements Team.

4.30.3.10 lap()

```
void RacingTeam::lap ( )  [virtual]
```

Method that allows both cars of the team to do laps

Implements Team.

4.30.3.11 loadMemento()

```
void RacingTeam::loadMemento (
            Memento * m )  [virtual]
```

method that loads the memento that was previously stored and reinstates it

Returns

int amount of the points the team has made

Implements Team.

4.30.3.12 setCarOne()

```
void RacingTeam::setCarOne (
            RaceCar * car1 )  [virtual]
```

method that sets the first car of the team

Returns

RaceCar object of the first car of the team

Implements Team.

4.30.3.13 setCarTwo()

```
void RacingTeam::setCarTwo (
            RaceCar * car2 )  [virtual]
```

Abstract interface method that sets the second car of the team

Returns

RaceCar object of the first car of the team

Implements Team.

4.30.3.14 setTeamName()

```
void RacingTeam::setTeamName (
            string name )  [virtual]
```

gets the team name

Parameters

string	of the team name

Implements Team.

4.30.3.15 setTeamPoints() [1/2]

```
void RacingTeam::setTeamPoints ( )  [virtual]
```

method that sets the points of the team

Implements Team.

4.30.3.16 setTeamPoints() [2/2]

```
void RacingTeam::setTeamPoints (
            int points )  [virtual]
```

method that sets the points of the team

Parameters

int	of the points

Implements Team.

4.30.3.17 setTireCompound()

```
void RacingTeam::setTireCompound (
            string tireCompound )  [virtual]
```

method that sets the tire Compound(soft , hard , medium)

Parameters

tireCompound	

Implements Team.

The documentation for this class was generated from the following files:

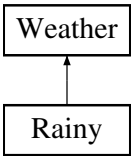
- Prototype/RacingTeam.h
- Prototype/RacingTeam.cpp

4.31 Rainy Class Reference

concrete state participant of the state participant

`#include <Rainy.h>`

Inheritance diagram for Rainy:



Public Member Functions

- Rainy ()
- virtual Weather * changeWeather ()

  constructor

4.31.1 Detailed Description

concrete state participant of the state participant

4.31.2 Constructor & Destructor Documentation

4.31.2.1 Rainy()

`Rainy::Rainy ( )`

4.31.3 Member Function Documentation

4.31.3.1 changeWeather()

```
Weather * Rainy::changeWeather ( )  [virtual]
```

constructor

method to change the state of the weather

Returns

the weather state as it has changed.

Implements Weather.

The documentation for this class was generated from the following files:

- StateWeather/Rainy.h
- StateWeather/Rainy.cpp

4.32 Results Struct Reference

The results Structure.

```
#include <Championship.h>
```

Public Attributes

- string driverName
- int team
  driver name
- int driver
  number of team
- double time
  number of driver
- double TeamTime
  time
- string teamName
  team time
- int points
  team name
- Team * teamObject
  amount of points recieved

4.32.1 Detailed Description

The results Structure.

4.32.2 Member Data Documentation

4.32.2.1 driver

```
int Results::driver
```

number of team

4.32.2.2 driverName

```
string Results::driverName
```

4.32.2.3 points

```
int Results::points
```

team name

4.32.2.4 team

```
int Results::team
```

driver name

4.32.2.5 teamName

```
string Results::teamName
```

team time

4.32.2.6 teamObject

```
Team* Results::teamObject
```

amount of points recieved

4.32.2.7 TeamTime

```
double Results::TeamTime
```

time

4.32.2.8 time

```
double Results::time
```

number of driver

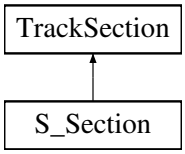
The documentation for this struct was generated from the following file:

- Template/Championship.h

4.33 S_Section Class Reference

```
#include <S_Section.h>
```

Inheritance diagram for S_Section:



Public Member Functions

- S_Section (int)
  Construct a new s section object.

Additional Inherited Members

4.33.1 Constructor & Destructor Documentation

4.33.1.1 S_Section()

```
S_Section::S_Section (
            int d )
```

Construct a new s section object.

Parameters

distance	

The documentation for this class was generated from the following files:

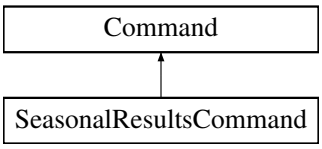
- Builder/S_Section.h
- Builder/S_Section.cpp

4.34 SeasonalResultsCommand Class Reference

concrete Command participant of the Command Design Pattern

`#include <SeasonalResultsCommand.h>`

Inheritance diagram for SeasonalResultsCommand:



Public Member Functions

- SeasonalResultsCommand (Team **)
- ~SeasonalResultsCommand ()
- void execute ()

default constructor

4.34.1 Detailed Description

concrete Command participant of the Command Design Pattern

4.34.2 Constructor & Destructor Documentation

4.34.2.1 SeasonalResultsCommand()

```
SeasonalResultsCommand::SeasonalResultsCommand (
            Team ** t )
```

Constructor

Parameters

teams	

4.34.2.2 ~SeasonalResultsCommand()

```
SeasonalResultsCommand::~SeasonalResultsCommand ( )
```

4.34.3 Member Function Documentation

4.34.3.1 execute()

```
void SeasonalResultsCommand::execute ( )  [virtual]
```

default constructor

Function that executes all the commands

Implements Command.

The documentation for this class was generated from the following files:

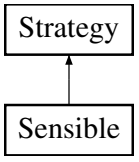
- Command/SeasonalResultsCommand.h
- Command/SeasonalResultsCommand.cpp

4.35 Sensible Class Reference

Concrete Strategy Participant of the Strategy design pattern.

```
#include <Sensible.h>
```

Inheritance diagram for Sensible:



Public Member Functions

- Sensible ()
- ~Sensible ()

  Constructor.
- string execute ()

  Destructor.
- string type ()

4.35.1 Detailed Description

Concrete Strategy Participant of the Strategy design pattern.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 Sensible()

```
Sensible::Sensible ( )
```

4.35.2.2 ~Sensible()

```
Sensible::~Sensible ( )
```

Constructor.

4.35.3 Member Function Documentation

4.35.3.1 execute()

```
string Sensible::execute ( )  [virtual]
```

Destructor.

Execute the strategy of the way the driver wants to race.

Returns

string that gives information about the tires given the strategy that is used.

Implements Strategy.

4.35.3.2 type()

```
string Sensible::type ( )  [virtual]
```

Returns the way the driver wants to race

Returns

string that displays the drivers strategy.

Implements Strategy.

The documentation for this class was generated from the following files:

- Strategy/Sensible.h
- Strategy/Sensible.cpp

4.36 SingletonChampionship Class Reference

```
#include <SingletonChampionship.h>
```

Public Member Functions

- void StartChampionship ()

Static Public Member Functions

- static SingletonChampionship * getInstance ()

Protected Member Functions

- SingletonChampionship ()
- ~SingletonChampionship ()

  constructor

4.36.1 Constructor & Destructor Documentation

4.36.1.1 SingletonChampionship()

```
SingletonChampionship::SingletonChampionship ( )  [protected]
```

4.36.1.2 ~SingletonChampionship()

```
SingletonChampionship::~SingletonChampionship ( )  [protected]
```

constructor

4.36.2 Member Function Documentation

4.36.2.1 getInstance()

```
SingletonChampionship * SingletonChampionship::getInstance ( )  [static]
```

method that returns a singleton object of the championship

Returns

SingletonChampionship

4.36.2.2 StartChampionship()

```
void SingletonChampionship::StartChampionship ( )
```

method to start the championship

The documentation for this class was generated from the following files:

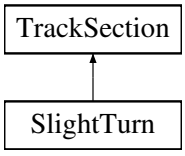
- Singleton/SingletonChampionship.h
- Singleton/SingletonChampionship.cpp

4.37 SlightTurn Class Reference

This is the leaf participant of the Composite Design Pattern.

```
#include <SlightTurn.h>
```

Inheritance diagram for SlightTurn:



Public Member Functions

- SlightTurn (int)

Additional Inherited Members

4.37.1 Detailed Description

This is the leaf participant of the Composite Design Pattern.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 SlightTurn()

```
SlightTurn::SlightTurn (
            int d )
```

The documentation for this class was generated from the following files:

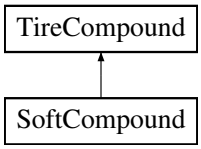
- Builder/SlightTurn.h
- Builder/SlightTurn.cpp

4.38 SoftCompound Class Reference

concrete strategy participant of the strategy design pattern

`#include <SoftCompound.h>`

Inheritance diagram for SoftCompound:



Public Member Functions

- SoftCompound ()
- ~SoftCompound ()
  constructor
- int getGrip ()
  destructor
- void setGrip (int grip)
- int getWear ()
- void setWear (int wear)
- double getRate ()
- TireCompound * clone ()

Additional Inherited Members

4.38.1 Detailed Description

concrete strategy participant of the strategy design pattern

4.38.2 Constructor & Destructor Documentation

4.38.2.1 SoftCompound()

```
SoftCompound::SoftCompound ( )
```

4.38.2.2 ~SoftCompound()

```
SoftCompound::~SoftCompound ( )
```

constructor

4.38.3 Member Function Documentation

4.38.3.1 clone()

```
TireCompound * SoftCompound::clone ( )  [virtual]
```

clone the TireCompound

Returns

TireCompound clone

Implements TireCompound.

4.38.3.2 getGrip()

```
int SoftCompound::getGrip ( )  [virtual]
```

destructor

get the grip of the tire

Returns

int grip of the car default value of 100

Implements TireCompound.

4.38.3.3 getRate()

```
double SoftCompound::getRate ( )  [virtual]
```

return the wear of the tire

Returns

double rate of the tires

Implements TireCompound.

4.38.3.4 getWear()

```
int SoftCompound::getWear ( )  [virtual]
```

return the wear of the tire

Returns

int wear of the tire

Implements TireCompound.

4.38.3.5 setGrip()

```
void SoftCompound::setGrip (
            int grip )  [virtual]
```

sets the grip of the tires

Parameters

grip	int

Implements TireCompound.

4.38.3.6 setWear()

```
void SoftCompound::setWear (
            int wear )  [virtual]
```

sets the wear of the tire

Parameters

wear	int

Implements TireCompound.

The documentation for this class was generated from the following files:

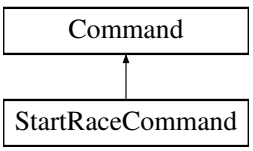
- TireCompoundStrategy/SoftCompound.h
- TireCompoundStrategy/SoftCompound.cpp

4.39 StartRaceCommand Class Reference

Concrete Command Participant of the Command Design Pattern.

`#include <StartRaceCommand.h>`

Inheritance diagram for StartRaceCommand:



Public Member Functions

- StartRaceCommand ()
- StartRaceCommand (Team **teams, BuildTrackCommand *)
  constructor
- StartRaceCommand (CreateTeamCommand *, BuildTrackCommand *)
- ~StartRaceCommand ()
- void execute ()

destructor

- Team ** getTeams ()
- void setTeams (Team **teams)
- RaceCar ** getCars ()
- void setCars (RaceCar **cars)
- BuildTrackCommand * getTrackBuilder ()
- void setTrackBuilder (BuildTrackCommand *)
- vector< TrackSection > getTrack ()
- void setTrack (vector< TrackSection >)

4.39.1 Detailed Description

Concrete Command Participant of the Command Design Pattern.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 StartRaceCommand() [1/3]

```
StartRaceCommand::StartRaceCommand ( )
```

4.39.2.2 StartRaceCommand() [2/3]

```
StartRaceCommand::StartRaceCommand (
            Team ** teams,
            BuildTrackCommand * t )
```

constructor

Constructor

Parameters

teams	
BuildtRackCommand	

4.39.2.3 StartRaceCommand() [3/3]

```
StartRaceCommand::StartRaceCommand (
            CreateTeamCommand * teamCom,
            BuildTrackCommand * t )
```

Constructor

Parameters

CreateTeamCommand	
BuildtRackCommand	

4.39.2.4 ~StartRaceCommand()

```
StartRaceCommand::~StartRaceCommand ( )
```

4.39.3 Member Function Documentation

4.39.3.1 execute()

```
void StartRaceCommand::execute ( )  [virtual]
```

destructor

Function that executes all the commands

Implements Command.

4.39.3.2 getCars()

```
RaceCar ** StartRaceCommand::getCars ( )
```

Returns the cars/drivers

Returns

RaceCar

4.39.3.3 getTeams()

```
Team ** StartRaceCommand::getTeams ( )
```

Returns the teams

Returns

Team

4.39.3.4 getTrack()

```
vector< TrackSection > StartRaceCommand::getTrack ( )
```

returns the track sections

Returns

the trac sections

4.39.3.5 getTrackBuilder()

```
BuildTrackCommand * StartRaceCommand::getTrackBuilder ( )
```

returns the track builder

Returns

the trackbuilder

4.39.3.6 setCars()

```
void StartRaceCommand::setCars (
            RaceCar ** cars )
```

Sets the drivers/cars

Parameters

cars	

4.39.3.7 setTeams()

```
void StartRaceCommand::setTeams (
            Team ** teams )
```

Sets the teams

Parameters

teams	taking part

4.39.3.8 setTrack()

```
void StartRaceCommand::setTrack (
            vector< TrackSection > t )
```

sets the track sections

Parameters

track	sections.

4.39.3.9 setTrackBuilder()

```
void StartRaceCommand::setTrackBuilder (
            BuildTrackCommand * t )
```

sets the buildtrackcommand

Parameters

BuildTrackCommand	

The documentation for this class was generated from the following files:

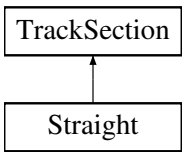
- Command/StartRaceCommand.h
- Command/StartRaceCommand.cpp

4.40 Straight Class Reference

This is the leaf participant of the Composite Design Pattern.

`#include <Straight.h>`

Inheritance diagram for Straight:



Public Member Functions

- Straight (int)

  Construct a new Straight object.

Additional Inherited Members

4.40.1 Detailed Description

This is the leaf participant of the Composite Design Pattern.

4.40.2 Constructor & Destructor Documentation

4.40.2.1 Straight()

```
Straight::Straight (
            int d )
```

Construct a new Straight object.

Parameters

disnatce	

The documentation for this class was generated from the following files:

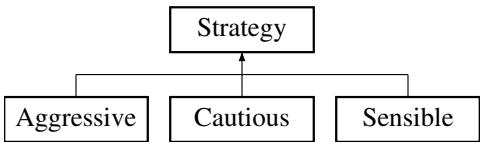
- Builder/Straight.h
- Builder/Straight.cpp

4.41 Strategy Class Reference

strategy participant of the Strategy design pattern

```
#include <Strategy.h>
```

Inheritance diagram for Strategy:



Public Member Functions

- Strategy ()
- virtual string execute ()=0

  constructor
- virtual string type ()=0

4.41.1 Detailed Description

strategy participant of the Strategy design pattern

4.41.2 Constructor & Destructor Documentation

4.41.2.1 Strategy()

```
Strategy::Strategy ( )  [inline]
```

4.41.3 Member Function Documentation

4.41.3.1 execute()

```
virtual string Strategy::execute ( )  [pure virtual]
```

constructor

Abstract interface to execute the strategy of the way the driver wants to race.

Returns

string that gives information about the tires given the strategy that is used.

Implemented in Sensible, Cautious, and Aggressive.

4.41.3.2 type()

```
virtual string Strategy::type ( )  [pure virtual]
```

Returns the way the driver wants to race

Returns

string that displays the drivers strategy.

Implemented in Sensible, Cautious, and Aggressive.

The documentation for this class was generated from the following file:

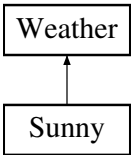
- Strategy/Strategy.h

4.42 Sunny Class Reference

concrete state participant of the state participant

`#include <Sunny.h>`

Inheritance diagram for Sunny:



Public Member Functions

- Sunny ()
- virtual Weather * changeWeather ()
  constructor

4.42.1 Detailed Description

concrete state participant of the state participant

4.42.2 Constructor & Destructor Documentation

4.42.2.1 Sunny()

```
Sunny::Sunny ( )
```

4.42.3 Member Function Documentation

4.42.3.1 changeWeather()

```
Weather * Sunny::changeWeather ( )  [virtual]
```

constructor

changes the state of the weather object

Returns

the weather state it has changed.

Implements Weather.

The documentation for this class was generated from the following files:

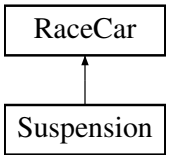
- StateWeather/Sunny.h
- StateWeather/Sunny.cpp

4.43 Suspension Class Reference

The leaf participant of the composite Design Pattern.

`#include <Suspension.h>`

Inheritance diagram for Suspension:



Public Member Functions

- Suspension ()

  Construct a new Suspension object.
- ∼Suspension ()

  Destroy the Suspension object.
- void degrade ()

  call the degrade method
- RaceCar ∗ clone ()

  Clone function that returns a clone of the current Race Car.

Additional Inherited Members

4.43.1 Detailed Description

The leaf participant of the composite Design Pattern.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 Suspension()

`Suspension::Suspension ( )`

Construct a new Suspension object.

4.43.2.2 ∼Suspension()

`Suspension::∼Suspension ( )`

Destroy the Suspension object.

4.43.3 Member Function Documentation

4.43.3.1 clone()

```
RaceCar * Suspension::clone ( )  [virtual]
```

Clone function that returns a clone of the current Race Car.

Returns

a clone of the the Race Car

Implements RaceCar.

4.43.3.2 degrade()

```
void Suspension::degrade ( )  [virtual]
```

call the degrade method

Reimplemented from RaceCar.

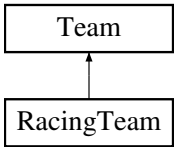
The documentation for this class was generated from the following files:

- CarComposite/Suspension.h
- CarComposite/Suspension.cpp

4.44 Team Class Reference

```
#include <Team.h>
```

Inheritance diagram for Team:



Public Member Functions

- Team ()
- Team (string tireCompound)

  default constructor
- ~Team ()
- virtual void buildCar ()=0

  destructor
- virtual void lap ()=0
- virtual void setTireCompound (string tireCompound)=0
- virtual RaceCar * getCarOne ()=0
- virtual CarPart * getCarOnePart ()=0
- virtual RaceCar * getCarTwo ()=0
- virtual CarPart * getCarTwoPart ()=0
- virtual void setCarOne (RaceCar *car1)=0
- virtual void setCarTwo (RaceCar *car2)=0
- virtual Team * clone ()=0
- virtual int getTeamPoints ()=0
- virtual void setTeamPoints ()=0
- virtual void setTeamPoints (int p)=0
- virtual string getTeamName ()=0
- virtual void setTeamName (string name)=0
- virtual Memento * createMemento ()=0
- virtual void loadMemento (Memento *m)=0

Protected Attributes

- CarBuilder * builder1
- CarBuilder * builder2
- RaceCar * car1
- CarPart * car1Part
- RaceCar * car2
- CarPart * car2Part
- string tireCompound
- int teamPoints = 0
- string teamName

4.44.1 Constructor & Destructor Documentation

4.44.1.1 Team() [1/2]

```
Team::Team ( )
```

4.44.1.2 Team() [2/2]

```
Team::Team (
            string tireCompound )
```

default constructor

Constructor that sets the tire compound

Parameters

tireCompound	wether the tire is soft medium or hard

4.44.1.3 ~Team()

```
Team::~Team ( )
```

4.44.2 Member Function Documentation

4.44.2.1 buildCar()

```
virtual void Team::buildCar ( )  [pure virtual]
```

destructor

Abstract interface Method that builds the car It adds all the elements to the car

Implemented in RacingTeam.

4.44.2.2 clone()

```
virtual Team* Team::clone ( )  [pure virtual]
```

Abstract interface method that clones the team

Returns

Team object

Implemented in RacingTeam.

4.44.2.3 createMemento()

```
virtual Memento* Team::createMemento ( )  [pure virtual]
```

Abstract interface method that creates a memento of the team

Returns

Memento of the team

Implemented in RacingTeam.

4.44.2.4 getCarOne()

```
virtual RaceCar* Team::getCarOne ( )  [pure virtual]
```

Abstract interface method that returns the first car of the team

Returns

RaceCar object of the first car of the team

Implemented in RacingTeam.

4.44.2.5 getCarOnePart()

```
virtual CarPart* Team::getCarOnePart ( )  [pure virtual]
```

Implemented in RacingTeam.

4.44.2.6 getCarTwo()

```
virtual RaceCar* Team::getCarTwo ( )  [pure virtual]
```

Abstract interface method that returns the second car of the team

Returns

RaceCar object of the second car of the team

Implemented in RacingTeam.

4.44.2.7 getCarTwoPart()

```
virtual CarPart* Team::getCarTwoPart ( )  [pure virtual]
```

Implemented in RacingTeam.

4.44.2.8 getTeamName()

```
virtual string Team::getTeamName ( )  [pure virtual]
```

Abstract interface method that returns the teams name

Returns

string of the teams name

Implemented in RacingTeam.

4.44.2.9 getTeamPoints()

```
virtual int Team::getTeamPoints ( )  [pure virtual]
```

Abstract interface method that returns the teams points

Returns

int amount of the points the team has made

Implemented in RacingTeam.

4.44.2.10 lap()

```
virtual void Team::lap ( )  [pure virtual]
```

Abstract interface Method that allows both cars of the team to do laps

Implemented in RacingTeam.

4.44.2.11 loadMemento()

```
virtual void Team::loadMemento (
            Memento * m )  [pure virtual]
```

Abstract interface method that loads the memento that was previously stored and reinstates it

Returns

int amount of the points the team has made

Implemented in RacingTeam.

4.44.2.12 setCarOne()

```
virtual void Team::setCarOne (
            RaceCar * car1 )  [pure virtual]
```

Abstract interface method that sets the first car of the team

Returns

RaceCar object of the first car of the team

Implemented in RacingTeam.

4.44.2.13 setCarTwo()

```
virtual void Team::setCarTwo (
            RaceCar * car2 )  [pure virtual]
```

Abstract interface method that sets the second car of the team

Returns

RaceCar object of the first car of the team

Implemented in RacingTeam.

4.44.2.14 setTeamName()

```
virtual void Team::setTeamName (
            string name )  [pure virtual]
```

Abstract interface method that sets the teams name

Parameters

name	of the team

Implemented in RacingTeam.

4.44.2.15 setTeamPoints() [1/2]

```
virtual void Team::setTeamPoints ( )  [pure virtual]
```

Abstract interface method that sets the teams points

Implemented in RacingTeam.

4.44.2.16 setTeamPoints() [2/2]

```
virtual void Team::setTeamPoints (
            int p )  [pure virtual]
```

Abstract interface method that sets the teams points

Implemented in RacingTeam.

4.44.2.17 setTireCompound()

```
virtual void Team::setTireCompound (
            string tireCompound )  [pure virtual]
```

Abstract interface method that sets the tire Compound(soft , hard , medium)

Parameters

tireCompound	

Implemented in RacingTeam.

4.44.3 Member Data Documentation

4.44.3.1 builder1

```
CarBuilder* Team::builder1  [protected]
```

the builder object to build the cars

4.44.3.2 builder2

```
CarBuilder* Team::builder2  [protected]
```

the builder object to build the cars

4.44.3.3 car1

```
RaceCar* Team::car1  [protected]
```

the first car that the car has

4.44.3.4 car1Part

```
CarPart* Team::car1Part  [protected]
```

the first car that the car has

4.44.3.5 car2

```
RaceCar* Team::car2  [protected]
```

the second car that the team has

4.44.3.6 car2Part

```
CarPart* Team::car2Part  [protected]
```

the first car that the car has

4.44.3.7 teamName

```
string Team::teamName  [protected]
```

the teams name

4.44.3.8 teamPoints

```
int Team::teamPoints = 0  [protected]
```

the points of the team

4.44.3.9 tireCompound

```
string Team::tireCompound  [protected]
```

the tire compound

The documentation for this class was generated from the following files:

- Prototype/Team.h
- Prototype/Team.cpp

4.45 TeamResult Struct Reference

the team results

`#include <SeasonalResultsCommand.h>`

Public Attributes

- Team * team
- string teamName

  the team
- int teamPoints

  team name

4.45.1 Detailed Description

the team results

4.45.2 Member Data Documentation

4.45.2.1 team

```
Team* TeamResult::team
```

4.45.2.2 teamName

```
string TeamResult::teamName
```

the team

4.45.2.3 teamPoints

```
int TeamResult::teamPoints
```

team name

The documentation for this struct was generated from the following file:

- Command/SeasonalResultsCommand.h

4.46 TeamResults Struct Reference

the results of the teams

`#include <Championship.h>`

Public Attributes

- int driver1Points
- int driver2Points

  the points driver one recieved
- int TeamPoints

  the points driver two recieved
- string teamName

  the points of both drivers summed together
- Team * teamObject

  the teams name

4.46.1 Detailed Description

the results of the teams

4.46.2 Member Data Documentation

4.46.2.1 driver1Points

`int TeamResults::driver1Points`

4.46.2.2 driver2Points

`int TeamResults::driver2Points`

the points driver one recieved

4.46.2.3 teamName

`string TeamResults::teamName`

the points of both drivers summed together

4.46.2.4 teamObject

```
Team* TeamResults::teamObject
```

the teams name

4.46.2.5 TeamPoints

```
int TeamResults::TeamPoints
```

the points driver two recieved

The documentation for this struct was generated from the following file:

- Template/Championship.h

4.47 TeamState Class Reference

Originator participant of the Memento Design Pattern.

```
#include <TeamState.h>
```

Public Member Functions

- TeamState (RaceCar *carOne, RaceCar *carTwo)
- TeamState (Team *teams)
- ~TeamState ()
- TeamState * getTeamState ()
- RaceCar * getCarOne ()
- RaceCar * getCarTwo ()
- string getTeamName ()
- int getTeamPoints ()
- Team * getTeam ()

4.47.1 Detailed Description

Originator participant of the Memento Design Pattern.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 TeamState() [1/2]

```
TeamState::TeamState (
            RaceCar * carOne,
            RaceCar * carTwo )
```

Constructor that sets the cars of the team

Parameters

carOne	the first car of the team
carTwo	the second car of the team

4.47.2.2 TeamState() [2/2]

```
TeamState::TeamState (
            Team * teams )
```

4.47.2.3 ~TeamState()

```
TeamState::~TeamState ( )
```

4.47.3 Member Function Documentation

4.47.3.1 getCarOne()

```
RaceCar * TeamState::getCarOne ( )
```

Returns the first race car of the team

Returns

CarOne of the team

4.47.3.2 getCarTwo()

```
RaceCar * TeamState::getCarTwo ( )
```

Returns the second race car of team

Returns

Car two of the team

4.47.3.3 getTeam()

```
Team * TeamState::getTeam ( )
```

Returns the team

Returns

team

4.47.3.4 getTeamName()

```
string TeamState::getTeamName ( )
```

Returns the name of the team

Returns

string of the teams name

4.47.3.5 getTeamPoints()

```
int TeamState::getTeamPoints ( )
```

Returns the points of the team

Returns

int of the teams points

4.47.3.6 getTeamState()

```
TeamState * TeamState::getTeamState ( )
```

Returns the state of the team

Returns

TeamState pointer that is stored

The documentation for this class was generated from the following files:

- Memento/TeamState.h
- Memento/TeamState.cpp

4.48 TeamStateCaretaker Class Reference

This is the caretaker participant in the memento design pattern.

`#include <TeamStateCaretaker.h>`

Public Member Functions

- TeamStateCaretaker (Memento ∗backupTeam)
- ∼TeamStateCaretaker ()
- Memento ∗ getBackupTeam ()

  destructor
- void setBackupTeam (Memento ∗backupTeam)

4.48.1 Detailed Description

This is the caretaker participant in the memento design pattern.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 TeamStateCaretaker()

```
TeamStateCaretaker::TeamStateCaretaker (
            Memento * backupTeam )
```

Constructor that sets the memento object

Parameters

backupteam	

4.48.2.2 ∼TeamStateCaretaker()

```
TeamStateCaretaker::~TeamStateCaretaker ( )
```

4.48.3 Member Function Documentation

4.48.3.1 getBackupTeam()

```
Memento * TeamStateCaretaker::getBackupTeam ( )
```

destructor

method that returns the mento of the team

Returns

Mento of the backup team

4.48.3.2 setBackupTeam()

```
void TeamStateCaretaker::setBackupTeam (
            Memento * backupTeam )
```

reinstate the team after a race (fix the cars or do a pitstop)

Parameters

backupteam	

The documentation for this class was generated from the following files:

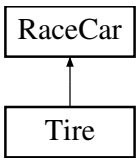
- Memento/TeamStateCaretaker.h
- Memento/TeamStateCaretaker.cpp

4.49 Tire Class Reference

concrete subject participant of the observer design pattern. The context participant of the State design Pattern

```
#include <Tire.h>
```

Inheritance diagram for Tire:



Public Member Functions

- Tire ()
- Tire (TireState ∗tState, TireCompound ∗type)

  default constructor
- Tire (string type)

  Constructor taking in the type of tire.
- ∼Tire ()
- RaceCar ∗ clone ()

  destructor
- TireState ∗ getState ()

  Get the State object.
- void setState (TireState ∗tState)

  Set the State object.
- void setType (string type)

  Set the Type object.
- void setType (TireCompound ∗type)

  Set the Type object.
- void lap ()

  let the car lap
- void degrade ()

  degrades the tires
- int getGrip ()

  Get the Grip object.
- void setGrip (int grip)

  Set the Grip object.
- int getWear ()

  Get the Wear object.
- void setWear (int wear)

  Set the Wear object.
- double getRate ()

  Get the Rate object of the tire compounds.
- string getNextTireCompound ()

  Get the Next Tire Compound object.

Additional Inherited Members

4.49.1 Detailed Description

concrete subject participant of the observer design pattern. The context participant of the State design Pattern

concrete subject participant of the observer design pattern

4.49.2 Constructor & Destructor Documentation

4.49.2.1 Tire() [1/3]

```
Tire::Tire ( )
```

4.49.2.2 Tire() [2/3]

```
Tire::Tire (
            TireState * tState,
            TireCompound * type )
```

default constructor

Constructor Taking in a tire state and the type of tire

Parameters

tireState	
type	

4.49.2.3 Tire() [3/3]

```
Tire::Tire (
            string type )
```

Constructor taking in the type of tire.

Parameters

type	

4.49.2.4 ~Tire()

```
Tire::~Tire ( )
```

4.49.3 Member Function Documentation

4.49.3.1 clone()

```
RaceCar * Tire::clone ( )  [virtual]
```

destructor

Clone function that returns a clone of the current Race Car

Returns

a clone of the the Race Car

Implements RaceCar.

4.49.3.2 degrade()

```
void Tire::degrade ( )  [virtual]
```

degrades the tires

Function to call each lap that degrades the tires.

Reimplemented from RaceCar.

4.49.3.3 getGrip()

```
int Tire::getGrip ( )
```

Get the Grip object.

Returns

int

4.49.3.4 getNextTireCompound()

```
string Tire::getNextTireCompound ( )
```

Get the Next Tire Compound object.

Returns

string

4.49.3.5 getRate()

```
double Tire::getRate ( )
```

Get the Rate object of the tire compounds.

Returns

double

4.49.3.6 getState()

```
TireState * Tire::getState ( )
```

Get the State object.

Returns

TireState*

4.49.3.7 getWear()

```
int Tire::getWear ( )
```

Get the Wear object.

Returns

int

4.49.3.8 lap()

```
void Tire::lap ( )  [virtual]
```

let the car lap

Reimplemented from RaceCar.

4.49.3.9 setGrip()

```
void Tire::setGrip (
            int grip )
```

Set the Grip object.

Parameters

grip	

4.49.3.10 setState()

```
void Tire::setState (
            TireState * tState )
```

Set the State object.

Parameters

tState	

4.49.3.11 setType() [1/2]

```
void Tire::setType (
            string type )
```

Set the Type object.

Parameters

type	

4.49.3.12 setType() [2/2]

```
void Tire::setType (
            TireCompound * type )
```

Set the Type object.

Parameters

type	

4.49.3.13 setWear()

```
void Tire::setWear (
            int wear )
```

Set the Wear object.

Parameters

wear	

The documentation for this class was generated from the following files:

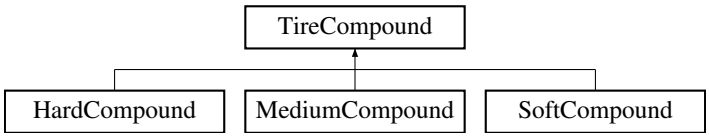
- CarComposite/Tire.h
- CarComposite/Tire.cpp

4.50 TireCompound Class Reference

strategy participant of the strategy design pattern

`#include <TireCompound.h>`

Inheritance diagram for TireCompound:



Public Member Functions

- TireCompound ()
- ~TireCompound ()

  constructor
- virtual int getGrip ()=0

  destructor
- virtual void setGrip (int grip)=0
- virtual int getWear ()=0
- virtual void setWear (int wear)=0
- virtual double getRate ()=0
- virtual TireCompound * clone ()=0

Protected Attributes

- int grip
- int wear
- int rate

4.50.1 Detailed Description

strategy participant of the strategy design pattern

4.50.2 Constructor & Destructor Documentation

4.50.2.1 TireCompound()

```
TireCompound::TireCompound ( )
```

4.50.2.2 ~TireCompound()

```
TireCompound::~TireCompound ( )
```

constructor

4.50.3 Member Function Documentation

4.50.3.1 clone()

```
virtual TireCompound* TireCompound::clone ( )  [pure virtual]
```

abstract interface to clone the TireCompound

Returns

TireCompound clone

Implemented in SoftCompound, MediumCompound, and HardCompound.

4.50.3.2 getGrip()

```
virtual int TireCompound::getGrip ( )  [pure virtual]
```

destructor

abstract interface to get the grip

Returns

int grip of the car default value of 100

Implemented in SoftCompound, MediumCompound, and HardCompound.

4.50.3.3 getRate()

```
virtual double TireCompound::getRate ( )  [pure virtual]
```

abstract interface to get the rate at which tires wear

Returns

double rate of the car

Implemented in SoftCompound, MediumCompound, and HardCompound.

4.50.3.4 getWear()

```
virtual int TireCompound::getWear ( )  [pure virtual]
```

abstract interface to get the wear

Returns

int wear of the car default value of 0

Implemented in SoftCompound, MediumCompound, and HardCompound.

4.50.3.5 setGrip()

```
virtual void TireCompound::setGrip (
            int grip )  [pure virtual]
```

abstract interface to set the grip of the tires

Parameters

grip	int

Implemented in SoftCompound, MediumCompound, and HardCompound.

4.50.3.6 setWear()

```
virtual void TireCompound::setWear (
            int wear )  [pure virtual]
```

abstract interface to set the wear of the tires

Parameters

wear	int

Implemented in SoftCompound, MediumCompound, and HardCompound.

4.50.4 Member Data Documentation

4.50.4.1 grip

```
int TireCompound::grip  [protected]
```

the grip of the tire

4.50.4.2 rate

```
int TireCompound::rate  [protected]
```

the rate of the tire

4.50.4.3 wear

```
int TireCompound::wear  [protected]
```

the wear of the tire

The documentation for this class was generated from the following files:

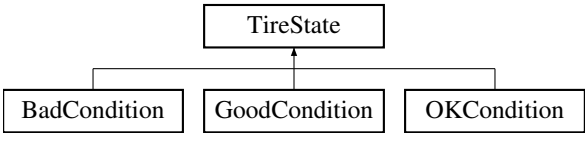
- TireCompoundStrategy/TireCompound.h
- TireCompoundStrategy/TireCompound.cpp

4.51 TireState Class Reference

The state participant of the State design pattern.

```
#include <TireState.h>
```

Inheritance diagram for TireState:



Public Member Functions

- TireState ()
- virtual ~TireState ()

  constructor
- virtual bool handle (Tire *tire)=0

  destructor
- virtual void changeTireState (Tire *tire)=0

  abstract interface to check the state of the tires
- virtual TireState * clone ()=0

  abstract interface to change the tire state

4.51.1 Detailed Description

The state participant of the State design pattern.

4.51.2 Constructor & Destructor Documentation

4.51.2.1 TireState()

```
TireState::TireState ( )
```

4.51.2.2 ~TireState()

```
TireState::~TireState ( )  [virtual]
```

constructor

4.51.3 Member Function Documentation

4.51.3.1 changeTireState()

```
virtual void TireState::changeTireState (
            Tire * tire )  [pure virtual]
```

abstract interface to check the state of the tires

abstract interface to change the tire state

Returns

true if the race car should pit and false otherwise

Implemented in OKCondition, GoodCondition, and BadCondition.

4.51.3.2 clone()

```
virtual TireState* TireState::clone ( )  [pure virtual]
```

abstract interface to change the tire state

abstract interface to clone the TireState.

Returns

a clone of the the tireState

Implemented in OKCondition, GoodCondition, and BadCondition.

4.51.3.3 handle()

```
virtual bool TireState::handle (
            Tire * tire )  [pure virtual]
```

destructor

Checks if the race car should pit

Parameters

Tire	this checks the state of the tire and if it is needded to be changed.

Returns

true if the race car should pit and false otherwise

Implemented in OKCondition, GoodCondition, and BadCondition.

The documentation for this class was generated from the following files:

- TireState/TireState.h
- TireState/TireState.cpp

4.52 Track Class Reference

```
#include <Track.h>
```

Public Member Functions

- Track ()
- Track (string n)
- string getTrackName ()

  Get the Track Name object.
- int getTrackDistance ()

  Get the Track Distance object.
- int getTrackRisk ()

  Get the Track Risk object.
- int getSectionCount ()

  Get the Section Count object.
- void addSection (TrackSection ∗)

  function to add a Track Section to the track
- void showTrack ()

  display the track
- vector< TrackSection > getTrack ()

  Get the Track object.

4.52.1 Constructor & Destructor Documentation

4.52.1.1 Track() [1/2]

```
Track::Track ( )
```

4.52.1.2 Track() [2/2]

```
Track::Track (
            string n )
```

Constructor taking in a string name

Parameters

name	

4.52.2 Member Function Documentation

4.52.2.1 addSection()

```
void Track::addSection (
            TrackSection * tempSection )
```

function to add a Track Section to the track

4.52.2.2 getSectionCount()

```
int Track::getSectionCount ( )
```

Get the Section Count object.

Returns

int

4.52.2.3 getTrack()

```
vector< TrackSection > Track::getTrack ( )
```

Get the Track object.

Returns

vector<TrackSection>

4.52.2.4 getTrackDistance()

```
int Track::getTrackDistance ( )
```

Get the Track Distance object.

Returns

int

4.52.2.5 getTrackName()

```
string Track::getTrackName ( )
```

Get the Track Name object.

Returns

string

4.52.2.6 getTrackRisk()

```
int Track::getTrackRisk ( )
```

Get the Track Risk object.

Returns

int

4.52.2.7 showTrack()

```
void Track::showTrack ( )
```

display the track

The documentation for this class was generated from the following files:

- Builder/Track.h
- Builder/Track.cpp

4.53 TrackBuilder Class Reference

```
#include <TrackBuilder.h>
```

Public Member Functions

- TrackBuilder ()
- TrackBuilder (string, int)
- ~TrackBuilder ()
- int getLaps ()

  Get the Laps object.
- string getName ()

  Get the Name object.
- string getLocation ()

  Get the Location object.
- void construct ()

  this builds the track or constructs the track consisting of the section we would want
- void construct (string, int)

  this builds the track or constructs the track consisting of the section we would want
- void display ()

  this displays the track
- ConcreteTrack * getTrack ()

  Get the Track object.

4.53.1 Constructor & Destructor Documentation

4.53.1.1 TrackBuilder() [1/2]

```
TrackBuilder::TrackBuilder ( )
```

4.53.1.2 TrackBuilder() [2/2]

```
TrackBuilder::TrackBuilder (
            string n,
            int l )
```

Constructor taking in a string name

Parameters

name	

4.53.1.3 ~TrackBuilder()

```
TrackBuilder::~TrackBuilder ( )
```

4.53.2 Member Function Documentation

4.53.2.1 construct() [1/2]

```
void TrackBuilder::construct ( )
```

this builds the track or constructs the track consisting of the section we would want

4.53.2.2 construct() [2/2]

```
void TrackBuilder::construct (
            string s,
            int l )
```

this builds the track or constructs the track consisting of the section we would want

4.53.2.3 display()

```
void TrackBuilder::display ( )
```

this displays the track

4.53.2.4 getLaps()

```
int TrackBuilder::getLaps ( )
```

Get the Laps object.

Returns

int

4.53.2.5 getLocation()

```
string TrackBuilder::getLocation ( )
```

Get the Location object.

Returns

string

4.53.2.6 getName()

```
string TrackBuilder::getName ( )
```

Get the Name object.

Returns

string

4.53.2.7 getTrack()

```
ConcreteTrack * TrackBuilder::getTrack ( )
```

Get the Track object.

Returns

ConcreteTrack*

The documentation for this class was generated from the following files:

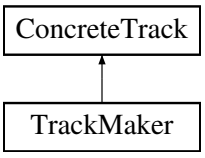
- Builder/TrackBuilder.h
- Builder/TrackBuilder.cpp

4.54 TrackMaker Class Reference

This is the client of the Composite Design Pattern.

```
#include <TrackMaker.h>
```

Inheritance diagram for TrackMaker:



Public Member Functions

- TrackMaker ()
- TrackMaker (string)

  a default constructor
- ~TrackMaker ()
- void addNinetyDegree (int)

  Destructor.
- void addStraight (int)

  adds a Straight section to the track
- void addHairpin (int)

  adds a Hairpin section to the track object
- void addS_section (int)

  adds a S section to the track object
- void addSlightTurn (int)

  adds a slight turn to the track object
- vector< TrackSection > getTrack ()

  Get the Track object.
- int getNumSections ()

  Get the Num Sections object.
- void showTrack ()

  returns the number of track sections.

4.54.1 Detailed Description

This is the client of the Composite Design Pattern.

This is the concrete builder class of The Builder design pattern

4.54.2 Constructor & Destructor Documentation

4.54.2.1 TrackMaker() [1/2]

```
TrackMaker::TrackMaker ( )
```

4.54.2.2 TrackMaker() [2/2]

```
TrackMaker::TrackMaker (
            string n )
```

a default constructor

Constructor taking in a string name

Parameters

name	

4.54.2.3 ~TrackMaker()

```
TrackMaker::~TrackMaker ( )
```

4.54.3 Member Function Documentation

4.54.3.1 addHairpin()

```
void TrackMaker::addHairpin (
            int d )  [virtual]
```

adds a Hairpin section to the track object

Parameters

distance	

Implements ConcreteTrack.

4.54.3.2 addNinetyDegree()

```
void TrackMaker::addNinetyDegree (
            int d )  [virtual]
```

Destructor.

adds a ninety degree turn section to the track

Parameters

distance	

Implements ConcreteTrack.

4.54.3.3 addS_section()

```
void TrackMaker::addS_section (
            int d )  [virtual]
```

adds a S section to the track object

Parameters

distance	

Implements ConcreteTrack.

4.54.3.4 addSlightTurn()

```
void TrackMaker::addSlightTurn (
            int d )  [virtual]
```

adds a slight turn to the track object

Parameters

distance	

Implements ConcreteTrack.

4.54.3.5 addStraight()

```
void TrackMaker::addStraight (
            int d )  [virtual]
```

adds a Straight section to the track

Parameters

distance	

Implements ConcreteTrack.

4.54.3.6 getNumSections()

```
int TrackMaker::getNumSections ( )  [virtual]
```

Get the Num Sections object.

Returns

int

Implements ConcreteTrack.

4.54.3.7 getTrack()

```
vector< TrackSection > TrackMaker::getTrack ( )  [virtual]
```

Get the Track object.

Returns

vector<TrackSection>

Implements ConcreteTrack.

4.54.3.8 showTrack()

```
void TrackMaker::showTrack ( )  [virtual]
```

returns the number of track sections.

prints out how the track currently looks like. eg the amount of straights or hairpins

Implements ConcreteTrack.

The documentation for this class was generated from the following files:

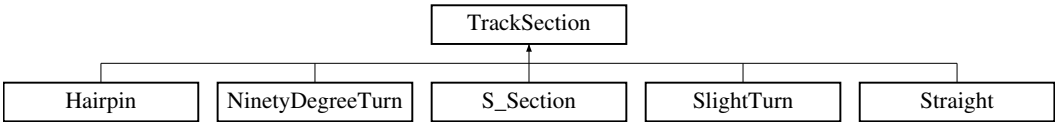
- Builder/TrackMaker.h
- Builder/TrackMaker.cpp

4.55 TrackSection Class Reference

This is the Composite participant of the Composite Design pattern.

```
#include <TrackSection.h>
```

Inheritance diagram for TrackSection:



Public Member Functions

- string getName ()
- int getRiskValue ()
- int getDistance ()

Protected Attributes

- string name
- int riskValue
- int distance

4.55.1 Detailed Description

This is the Composite participant of the Composite Design pattern.

4.55.2 Member Function Documentation

4.55.2.1 getDistance()

```
int TrackSection::getDistance ( )
```

a get method to get the distance of the piece of track

Returns

int distance of track

4.55.2.2 getName()

```
string TrackSection::getName ( )
```

returns the string of the tracksection

Returns

string name

4.55.2.3 getRiskValue()

```
int TrackSection::getRiskValue ( )
```

a get method to get the risk value of the piece of track

Returns

int risk value

4.55.3 Member Data Documentation

4.55.3.1 distance

```
int TrackSection::distance  [protected]
```

the distance of the piece of track

4.55.3.2 name

```
string TrackSection::name  [protected]
```

name of the track section

4.55.3.3 riskValue

```
int TrackSection::riskValue  [protected]
```

The risk value of driving on the piece of track eg you have a higher risk of crashing on a hairpin than on a straight

The documentation for this class was generated from the following files:

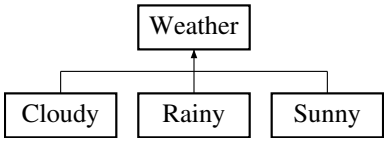
- Builder/TrackSection.h
- Builder/TrackSection.cpp

4.56 Weather Class Reference

state partipant of the state design pattern

```
#include <Weather.h>
```

Inheritance diagram for Weather:



Public Member Functions

- Weather ()
- virtual Weather * changeWeather ()=0

  constructor
- string getWeather ()
- void setWeather (string)

4.56.1 Detailed Description

state partipant of the state design pattern

4.56.2 Constructor & Destructor Documentation

4.56.2.1 Weather()

```
Weather::Weather ( )
```

4.56.3 Member Function Documentation

4.56.3.1 changeWeather()

```
virtual Weather* Weather::changeWeather ( )  [pure virtual]
```

constructor

abstract interface to change the weather state

Returns

the weather state as it has changed.

Implemented in Sunny, Rainy, and Cloudy.

4.56.3.2 getWeather()

```
string Weather::getWeather ( )
```

Checks if the race car should pit

Returns

the string of the weather

4.56.3.3 setWeather()

```
void Weather::setWeather (
            string w )
```

sets the weather

Parameters

weather	string that takes in the whether of the day default weather of sunny

The documentation for this class was generated from the following files:

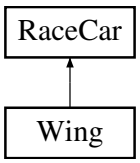
- StateWeather/Weather.h
- StateWeather/Weather.cpp

4.57 Wing Class Reference

This is the leaf participant of the Composite design Pattern.

`#include <Wing.h>`

Inheritance diagram for Wing:



Public Member Functions

- Wing ()
- ∼Wing ()

  constructor
- void degrade ()

  destructor
- RaceCar ∗ clone ()

  Clone function that returns a clone of the current Race Car.

Additional Inherited Members

4.57.1 Detailed Description

This is the leaf participant of the Composite design Pattern.

This is the leaf participant of the Composite design Pattern

4.57.2 Constructor & Destructor Documentation

4.57.2.1 Wing()

```
Wing::Wing ( )
```

4.57.2.2 ~Wing()

```
Wing::~Wing ( )
```

constructor

4.57.3 Member Function Documentation

4.57.3.1 clone()

```
RaceCar * Wing::clone ( )  [virtual]
```

Clone function that returns a clone of the current Race Car.

Returns

a clone of the the Race Car

Implements RaceCar.

4.57.3.2 degrade()

```
void Wing::degrade ( )  [virtual]
```

destructor

function that allows the wings to degrade during the race.

Reimplemented from RaceCar.

The documentation for this class was generated from the following files:

- CarComposite/Wing.h
- CarComposite/Wing.cpp

Chapter 5

File Documentation

5.1 Builder/ConcreteTrack.cpp File Reference

```
#include "ConcreteTrack.h"
```

5.2 Builder/ConcreteTrack.h File Reference

```
#include <iostream>
#include <list>
#include <vector>
#include "Track.h"
```

Classes

- class ConcreteTrack

  The builder participant in the Builder design Pattern.

5.3 Builder/Hairpin.cpp File Reference

```
#include "Hairpin.h"
```

5.4 Builder/Hairpin.h File Reference

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

Classes

- class Hairpin

5.5 Builder/NinetyDegreeTurn.cpp File Reference

```
#include "NinetyDegreeTurn.h"
```

5.6 Builder/NinetyDegreeTurn.h File Reference

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

Classes

- class NinetyDegreeTurn

5.7 Builder/S_Section.cpp File Reference

```
#include "S_Section.h"
```

5.8 Builder/S_Section.h File Reference

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

Classes

- class S_Section

5.9 Builder/SlightTurn.cpp File Reference

```
#include "SlightTurn.h"
```

5.10 Builder/SlightTurn.h File Reference

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

Classes

- class SlightTurn

  This is the leaf participant of the Composite Design Pattern.

5.11 Builder/Straight.cpp File Reference

```
#include "Straight.h"
```

5.12 Builder/Straight.h File Reference

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

Classes

- class Straight

  This is the leaf participant of the Composite Design Pattern.

5.13 Builder/Track.cpp File Reference

```
#include "Track.h"
```

5.14 Builder/Track.h File Reference

```
#include <iostream>
#include <map>
#include <cstring>
#include <list>
#include <iterator>
#include <vector>
#include "TrackSection.h"
#include "SlightTurn.h"
#include "Straight.h"
#include "S_Section.h"
#include "NinetyDegreeTurn.h"
#include "Hairpin.h"
```

Classes

- class Track

5.15 Builder/TrackBuilder.cpp File Reference

```
#include "TrackBuilder.h"
```

5.16 Builder/TrackBuilder.h File Reference

```
#include "TrackMaker.h"
```

Classes

- class TrackBuilder

5.17 Builder/TrackMaker.cpp File Reference

```
#include "TrackMaker.h"
```

5.18 Builder/TrackMaker.h File Reference

```
#include <iostream>
#include <map>
#include <cstring>
#include <list>
#include <iterator>
#include "Track.h"
#include "ConcreteTrack.h"
```

Classes

- class TrackMaker

  This is the client of the Composite Design Pattern.

5.19 Builder/TrackSection.cpp File Reference

```
#include "TrackSection.h"
```

5.20 Builder/TrackSection.h File Reference

```
#include <iostream>
```

Classes

- class TrackSection

  This is the Composite participant of the Composite Design pattern.

5.21 CarComposite/CarBuilder.cpp File Reference

```
#include "CarBuilder.h"
```

5.22 CarComposite/CarBuilder.h File Reference

```
#include <iostream>
#include "RaceCarBuilder.h"
#include "Tire.h"
#include "Chassie.h"
#include "Engine.h"
#include "Hub.h"
#include "Suspension.h"
#include "Wing.h"
#include "CarPart.h"
```

Classes

- class CarBuilder

  this class is the Concrete Builder in the Builder design Pattern

5.23 CarComposite/CarPart.cpp File Reference

```
#include "CarPart.h"
```

5.24 CarComposite/CarPart.h File Reference

```
#include <iostream>
#include <list>
#include "RaceCar.h"
#include "Tire.h"
```

Classes

- class CarPart

  the composite participant of the Composite Design Pattern

5.25 CarComposite/Chassie.cpp File Reference

```
#include "Chassie.h"
```

5.26 CarComposite/Chassie.h File Reference

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

Classes

- class Chassie

  leaf participant of the Composite Design Pattern

5.27 CarComposite/Engine.cpp File Reference

```
#include "Engine.h"
```

5.28 CarComposite/Engine.h File Reference

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

Classes

- class Engine

  leaf participant of the Composite design Pattern

5.29 CarComposite/Hub.cpp File Reference

```
#include "Hub.h"
```

5.30 CarComposite/Hub.h File Reference

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

Classes

- class Hub

  leaf participant of the Composite Design pattern

5.31 CarComposite/RaceCar.cpp File Reference

```
#include "RaceCar.h"
```

5.32 CarComposite/RaceCar.h File Reference

```
#include <list>
#include <iostream>
#include "../Observer/PitStop.h"
#include "../Strategy/Strategy.h"
#include "../Strategy/Sensible.h"
#include "../Strategy/Cautious.h"
#include "../Strategy/Aggressive.h"
#include "../TireCompoundStrategy/SoftCompound.h"
#include "../TireCompoundStrategy/MediumCompound.h"
#include "../TireCompoundStrategy/HardCompound.h"
```

Classes

- class RaceCar

  This class is the product participant of the Builder Design pattern. Subject participant of the observer design Pattern.

5.33 CarComposite/RaceCarBuilder.cpp File Reference

```
#include "RaceCarBuilder.h"
```

5.34 CarComposite/RaceCarBuilder.h File Reference

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

Classes

- class RaceCarBuilder

  The Builder participant of The BUilder Design Pattern.

5.35 CarComposite/Suspension.cpp File Reference

```
#include "Suspension.h"
```

5.36 CarComposite/Suspension.h File Reference

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

Classes

- class Suspension

  The leaf participant of the composite Design Pattern.

5.37 CarComposite/Tire.cpp File Reference

```
#include "Tire.h"
```

5.38 CarComposite/Tire.h File Reference

```
#include <iostream>
#include <stdio.h>
#include <ctime>
#include "RaceCar.h"
#include "../TireCompoundStrategy/SoftCompound.h"
#include "../TireCompoundStrategy/MediumCompound.h"
#include "../TireCompoundStrategy/HardCompound.h"
#include "../TireState/GoodCondition.h"
#include "../TireState/TireState.h"
#include "../Observer/ChangeTires.h"
```

Classes

- class Tire

  concrete subject participant of the observer design pattern. The context participant of the State design Pattern

5.39 CarComposite/Wing.cpp File Reference

```
#include "Wing.h"
```

5.40 CarComposite/Wing.h File Reference

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

Classes

- class Wing

  This is the leaf participant of the Composite design Pattern.

5.41 Command/BuildTrackCommand.cpp File Reference

```
#include "BuildTrackCommand.h"
```

5.42 Command/BuildTrackCommand.h File Reference

```
#include <iostream>
#include <list>
#include <iterator>
#include "Command.h"
#include "../Builder/TrackBuilder.h"
#include "../Builder/ConcreteTrack.h"
```

Classes

- class BuildTrackCommand

5.43 Command/Command.h File Reference

```
#include <iostream>
```

Classes

- class Command

5.44 Command/CreateTeamCommand.cpp File Reference

```
#include "CreateTeamCommand.h"
```

5.45 Command/CreateTeamCommand.h File Reference

```
#include "Command.h"
#include "../Prototype/Team.h"
#include "../Prototype/RacingTeam.h"
#include "../Memento/TeamStateCaretaker.h"
```

Classes

- class CreateTeamCommand

5.46 Command/RaceConditionCommand.cpp File Reference

```
#include "RaceConditionCommand.h"
```

5.47 Command/RaceConditionCommand.h File Reference

```
#include <iostream>
#include <cstdio>
#include "Command.h"
#include "../StateWeather/Weather.h"
#include "../StateWeather/Race.h"
```

Classes

- class RaceConditionCommand

5.48 Command/SeasonalResultsCommand.cpp File Reference

```
#include "SeasonalResultsCommand.h"
```

5.49 Command/SeasonalResultsCommand.h File Reference

```
#include <iostream>
#include <cstdio>
#include "Command.h"
#include "../Prototype/Team.h"
```

Classes

- struct TeamResult

  the team results
- class SeasonalResultsCommand

  concrete Command participant of the Command Design Pattern

5.50 Command/StartRaceCommand.cpp File Reference

```
#include "StartRaceCommand.h"
```

5.51 Command/StartRaceCommand.h File Reference

```
#include "Command.h"
#include "../Memento/Memento.h"
#include "../Memento/TeamState.h"
#include "../Memento/TeamStateCaretaker.h"
#include "../Prototype/Team.h"
#include "../Prototype/RacingTeam.h"
#include "../CarComposite/RaceCar.h"
#include "BuildTrackCommand.h"
#include "RaceConditionCommand.h"
#include "CreateTeamCommand.h"
#include "../Template/Championship.h"
#include "../Template/ConstructorsChampionship.h"
#include "../Template/DriversChampionship.h"
```

Classes

- class StartRaceCommand

  Concrete Command Participant of the Command Design Pattern.

5.52 main.cpp File Reference

```
#include <string>
#include <iostream>
#include <new>
#include "Memento/TeamStateCaretaker.h"
#include "Memento/Memento.h"
#include "Memento/TeamState.h"
#include "Memento/TeamStateCaretaker.cpp"
#include "Memento/Memento.cpp"
#include "Memento/TeamState.cpp"
#include "Observer/PitStop.h"
#include "Observer/ChangeTires.h"
#include "Observer/ChangeTires.cpp"
#include "Builder/Track.h"
#include "Builder/TrackSection.h"
#include "Builder/ConcreteTrack.h"
#include "Builder/TrackBuilder.h"
#include "Builder/TrackMaker.h"
#include "Builder/NinetyDegreeTurn.h"
#include "Builder/Hairpin.h"
#include "Builder/S_Section.h"
#include "Builder/SlightTurn.h"
#include "Builder/Straight.h"
#include "Builder/Track.cpp"
#include "Builder/TrackSection.cpp"
#include "Builder/ConcreteTrack.cpp"
#include "Builder/TrackBuilder.cpp"
#include "Builder/TrackMaker.cpp"
#include "Builder/NinetyDegreeTurn.cpp"
#include "Builder/Hairpin.cpp"
#include "Builder/S_Section.cpp"
#include "Builder/SlightTurn.cpp"
#include "Builder/Straight.cpp"
#include "Command/Command.h"
#include "Command/BuildTrackCommand.h"
#include "Command/BuildTrackCommand.cpp"
#include "CarComposite/CarBuilder.h"
#include "CarComposite/CarPart.h"
#include "CarComposite/Chassie.h"
#include "CarComposite/Engine.h"
#include "CarComposite/Hub.h"
#include "CarComposite/RaceCar.h"
#include "CarComposite/RaceCarBuilder.h"
#include "CarComposite/Suspension.h"
#include "CarComposite/Tire.h"
#include "CarComposite/Wing.h"
#include "CarComposite/CarBuilder.cpp"
#include "CarComposite/CarPart.cpp"
#include "CarComposite/Chassie.cpp"
#include "CarComposite/Engine.cpp"
#include "CarComposite/Hub.cpp"
#include "CarComposite/RaceCar.cpp"
#include "CarComposite/RaceCarBuilder.cpp"
#include "CarComposite/Suspension.cpp"
#include "CarComposite/Tire.cpp"
#include "CarComposite/Wing.cpp"
#include "TireState/GoodCondition.h"
```

```
#include "TireState/OKCondition.h"
#include "TireState/BadCondition.h"
#include "TireState/TireState.h"
#include "TireState/TireState.cpp"
#include "TireState/GoodCondition.cpp"
#include "TireState/OKCondition.cpp"
#include "TireState/BadCondition.cpp"
#include "TireCompoundStrategy/TireCompound.h"
#include "TireCompoundStrategy/SoftCompound.h"
#include "TireCompoundStrategy/MediumCompound.h"
#include "TireCompoundStrategy/HardCompound.h"
#include "TireCompoundStrategy/TireCompound.cpp"
#include "TireCompoundStrategy/SoftCompound.cpp"
#include "TireCompoundStrategy/MediumCompound.cpp"
#include "TireCompoundStrategy/HardCompound.cpp"
#include "Prototype/Team.h"
#include "Prototype/RacingTeam.h"
#include "Prototype/Team.cpp"
#include "Prototype/RacingTeam.cpp"
#include "Strategy/Strategy.h"
#include "Strategy/Sensible.h"
#include "Strategy/Cautious.h"
#include "Strategy/Aggressive.h"
#include "Strategy/Sensible.cpp"
#include "Strategy/Cautious.cpp"
#include "Strategy/Aggressive.cpp"
#include "Command/CreateTeamCommand.h"
#include "Command/CreateTeamCommand.cpp"
#include "Command/StartRaceCommand.h"
#include "Command/StartRaceCommand.cpp"
#include "Singleton/SingletonChampionship.h"
#include "Singleton/SingletonChampionship.cpp"
#include "Command/RaceConditionCommand.h"
#include "Command/RaceConditionCommand.cpp"
#include "StateWeather/Cloudy.h"
#include "StateWeather/Rainy.h"
#include "StateWeather/Sunny.h"
#include "StateWeather/Weather.h"
#include "StateWeather/Race.h"
#include "StateWeather/Cloudy.cpp"
#include "StateWeather/Rainy.cpp"
#include "StateWeather/Sunny.cpp"
#include "StateWeather/Weather.cpp"
#include "StateWeather/Race.cpp"
#include "Template/Championship.h"
#include "Template/ConstructorsChampionship.h"
#include "Template/Championship.cpp"
#include "Template/ConstructorsChampionship.cpp"
#include "Template/DriversChampionship.cpp"
#include "Command/SeasonalResultsCommand.h"
#include "Command/SeasonalResultsCommand.cpp"
```

Functions

- int main ()

5.52.1 Function Documentation

5.52.1.1 main()

```
int main ( )
```

5.53 Memento/Memento.cpp File Reference

```
#include "Memento.h"
```

5.54 Memento/Memento.h File Reference

```
#include "TeamState.h"
#include "../CarComposite/RaceCar.h"
#include "../Prototype/Team.h"
#include "../Prototype/RacingTeam.h"
```

Classes

- class Memento

  the memento participant of the memento design pattern

5.55 Memento/TeamState.cpp File Reference

```
#include "TeamState.h"
```

5.56 Memento/TeamState.h File Reference

```
#include "../CarComposite/RaceCar.h"
#include "../CarComposite/CarBuilder.h"
#include "../CarComposite/CarPart.h"
#include "../Prototype/Team.h"
#include "../Prototype/RacingTeam.h"
```

Classes

- class TeamState

  Originator participant of the Memento Design Pattern.

5.57 Memento/TeamStateCaretaker.cpp File Reference

```
#include "TeamStateCaretaker.h"
```

5.58 Memento/TeamStateCaretaker.h File Reference

```
#include "Memento.h"
```

Classes

- class TeamStateCaretaker

  This is the caretaker participant in the memento design pattern.

5.59 Observer/ChangeTires.cpp File Reference

```
#include "ChangeTires.h"
```

5.60 Observer/ChangeTires.h File Reference

```
#include "PitStop.h"
#include "../CarComposite/Tire.h"
#include "../TireState/TireState.h"
```

Classes

- class ChangeTires

  concrete observer participant of the Observer design pattern.

5.61 Observer/PitStop.h File Reference

Classes

- class PitStop

  The observer participant of the observer design pattern.

5.62 Prototype/RacingTeam.cpp File Reference

```
#include "RacingTeam.h"
```

5.63 Prototype/RacingTeam.h File Reference

```
#include "Team.h"
#include "../CarComposite/CarBuilder.h"
#include "../CarComposite/RaceCar.h"
#include "../CarComposite/CarPart.h"
#include "../Memento/TeamState.h"
#include "../Memento/TeamStateCaretaker.h"
#include "../Memento/Memento.h"
```

Classes

- class RacingTeam

  concrete prototype of the prototype design pattern

5.64 Prototype/Team.cpp File Reference

```
#include "Team.h"
```

5.65 Prototype/Team.h File Reference

```
#include <iostream>
#include "../Memento/TeamState.h"
#include "../Memento/TeamStateCaretaker.h"
#include "../CarComposite/CarBuilder.h"
#include "../CarComposite/RaceCar.h"
#include "../CarComposite/CarPart.h"
#include "../Memento/Memento.h"
```

Classes

- class Team

5.66 Singleton/SingletonChampionship.cpp File Reference

```
#include "SingletonChampionship.h"
```

5.67 Singleton/SingletonChampionship.h File Reference

```
#include <iostream>
#include <new>
#include "../Memento/Memento.h"
#include "../Memento/TeamState.h"
#include "../Memento/TeamStateCaretaker.h"
#include "../Builder/Track.h"
#include "../Builder/TrackSection.h"
#include "../Builder/ConcreteTrack.h"
#include "../Builder/TrackBuilder.h"
#include "../Builder/TrackMaker.h"
#include "../Builder/NinetyDegreeTurn.h"
#include "../Builder/Hairpin.h"
#include "../Builder/S_Section.h"
#include "../Builder/SlightTurn.h"
#include "../Builder/Straight.h"
#include "../Command/Command.h"
#include "../Command/BuildTrackCommand.h"
#include "../CarComposite/CarBuilder.h"
#include "../CarComposite/CarPart.h"
#include "../CarComposite/Chassie.h"
#include "../CarComposite/Engine.h"
#include "../CarComposite/Hub.h"
#include "../CarComposite/RaceCar.h"
#include "../CarComposite/RaceCarBuilder.h"
#include "../CarComposite/Suspension.h"
#include "../CarComposite/Tire.h"
#include "../CarComposite/Wing.h"
#include "../TireState/GoodCondition.h"
#include "../TireState/OKCondition.h"
#include "../TireState/BadCondition.h"
#include "../TireState/TireState.h"
#include "../TireCompoundStrategy/TireCompound.h"
#include "../TireCompoundStrategy/SoftCompound.h"
#include "../TireCompoundStrategy/MediumCompound.h"
#include "../TireCompoundStrategy/HardCompound.h"
#include "../Prototype/Team.h"
#include "../Prototype/RacingTeam.h"
#include "../Strategy/Strategy.h"
#include "../Strategy/Sensible.h"
#include "../Strategy/Cautious.h"
#include "../Strategy/Aggressive.h"
#include "../Command/CreateTeamCommand.h"
#include "../Command/StartRaceCommand.h"
#include "../Command/SeasonalResultsCommand.h"
```

Classes

- class SingletonChampionship

5.68 StateWeather/Cloudy.cpp File Reference

```
#include "Cloudy.h"
```

5.69 StateWeather/Cloudy.h File Reference

```
#include <iostream>
#include "Weather.h"
#include "Race.h"
#include "Sunny.h"
#include "Rainy.h"
```

Classes

- class Cloudy

  concrete state participant of the state participant

5.70 StateWeather/Race.cpp File Reference

```
#include "Race.h"
```

5.71 StateWeather/Race.h File Reference

```
#include <iostream>
#include "Weather.h"
#include "Sunny.h"
#include "Rainy.h"
#include "Cloudy.h"
```

Classes

- class Race

  context participant of the state design pattern

5.72 StateWeather/Rainy.cpp File Reference

```
#include "Rainy.h"
```

5.73 StateWeather/Rainy.h File Reference

```
#include <iostream>
#include "Weather.h"
#include "Race.h"
#include "Sunny.h"
#include "Cloudy.h"
```

Classes

- class Rainy

  concrete state participant of the state participant

5.74 StateWeather/Sunny.cpp File Reference

```
#include "Sunny.h"
```

5.75 StateWeather/Sunny.h File Reference

```
#include <iostream>
#include "Weather.h"
#include "Race.h"
#include "Rainy.h"
#include "Cloudy.h"
```

Classes

- class Sunny

  concrete state participant of the state participant

5.76 StateWeather/Weather.cpp File Reference

```
#include "Weather.h"
```

5.77 StateWeather/Weather.h File Reference

```
#include <iostream>
```

Classes

- class Weather

  state partipant of the state design pattern

5.78 Strategy/Aggressive.cpp File Reference

```
#include "Aggressive.h"
```

5.79 Strategy/Aggressive.h File Reference

```
#include "Strategy.h"
```

Classes

- class Aggressive

  Concrete Strategy Participant of the Strategy design pattern.

5.80 Strategy/Cautious.cpp File Reference

```
#include "Cautious.h"
```

5.81 Strategy/Cautious.h File Reference

```
#include "Strategy.h"
```

Classes

- class Cautious

  Concrete Strategy participant of the strategy design Pattern.

5.82 Strategy/Sensible.cpp File Reference

```
#include "Sensible.h"
```

5.83 Strategy/Sensible.h File Reference

```
#include "Strategy.h"
```

Classes

- class Sensible

  Concrete Strategy Participant of the Strategy design pattern.

5.84 Strategy/Strategy.h File Reference

```
#include <iostream>
#include <stdio.h>
```

Classes

- class Strategy

  strategy participant of the Strategy design pattern

5.85 Template/Championship.cpp File Reference

```
#include "Championship.h"
```

5.86 Template/Championship.h File Reference

```
#include "../Prototype/Team.h"
#include "../Prototype/RacingTeam.h"
#include <iostream>
```

Classes

- struct TeamResults

  the results of the teams
- struct Results

  The results Structure.
- class Championship

5.87 Template/ConstructorsChampionship.cpp File Reference

```
#include "ConstructorsChampionship.h"
```

5.88 Template/ConstructorsChampionship.h File Reference

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

Classes

- class ConstructorsChampionship

5.89 Template/DriversChampionship.cpp File Reference

```
#include "DriversChampionship.h"
```

5.90 Template/DriversChampionship.h File Reference

```
#include "Championship.h"
```

Classes

- class DriversChampionship

5.91 TireCompoundStrategy/HardCompound.cpp File Reference

```
#include "HardCompound.h"
```

5.92 TireCompoundStrategy/HardCompound.h File Reference

```
#include "TireCompound.h"
```

Classes

- class HardCompound

  concrete strategy participant of the strategy design pattern

5.93 TireCompoundStrategy/MediumCompound.cpp File Reference

```
#include "MediumCompound.h"
```

5.94 TireCompoundStrategy/MediumCompound.h File Reference

```
#include "TireCompound.h"
```

Classes

- class MediumCompound

  concrete strategy participant from the strategy design pattern

5.95 TireCompoundStrategy/SoftCompound.cpp File Reference

```
#include "SoftCompound.h"
```

5.96 TireCompoundStrategy/SoftCompound.h File Reference

```
#include "TireCompound.h"
```

Classes

- class SoftCompound

  concrete strategy participant of the strategy design pattern

5.97 TireCompoundStrategy/TireCompound.cpp File Reference

```
#include "TireCompound.h"
```

5.98 TireCompoundStrategy/TireCompound.h File Reference

Classes

- class TireCompound

  strategy participant of the strategy design pattern

5.99 TireState/BadCondition.cpp File Reference

```
#include "BadCondition.h"
```

5.100 TireState/BadCondition.h File Reference

```
#include <iostream>
#include "TireState.h"
#include "GoodCondition.h"
```

Classes

- class BadCondition

  The concrete state participant of the State design Pattern.

5.101 TireState/GoodCondition.cpp File Reference

```
#include "GoodCondition.h"
```

5.102 TireState/GoodCondition.h File Reference

```
#include <iostream>
#include "TireState.h"
#include "OKCondition.h"
```

Classes

- class GoodCondition

  the concrete state participant of the State design Pattern

5.103 TireState/OKCondition.cpp File Reference

```
#include "OKCondition.h"
```

5.104 TireState/OKCondition.h File Reference

```
#include <iostream>
#include "TireState.h"
#include "BadCondition.h"
```

Classes

- class OKCondition

  The concrete state participant of the State design Pattern.

5.105 TireState/TireState.cpp File Reference

```
#include "TireState.h"
```

5.106 TireState/TireState.h File Reference

```
#include <iostream>
#include <string>
#include "../CarComposite/Tire.h"
```

Classes

- class TireState

  The state participant of the State design pattern.

Index