Drag Race Revengance: BenchRacer Simulator 2: Electric Boogaloo

By Nick Rodgers, TPMSKR
Programming 3

This project took quite a while to finish, longer than I anticipated in fact, because I made lots of small mistakes and most of the errors stemmed from a line of code here or there that completely ruined the functionality of the program. As frustrating as it was, I believe it was a good exercise in not only swing, but also in weeding out bugs. Eventually though, it all worked and this is how it works:

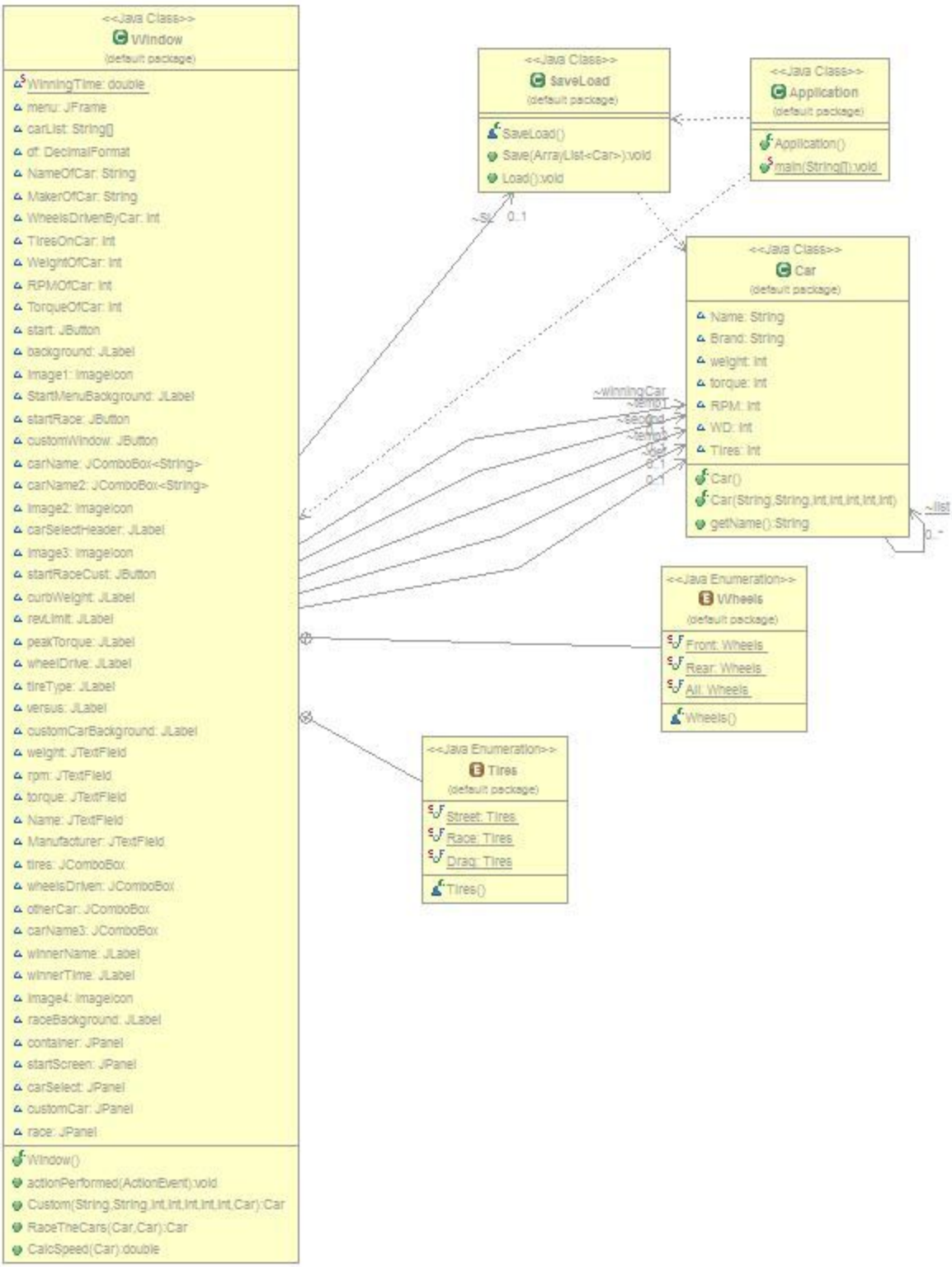
The application starts, and loads from the serialized file of cars that can be expanded by the user adding custom cars. It creates a new window, where most of the operations happen. In the window, 90% of the code is just setting up the panels so that they work nicely, the actual calculation is rather short. At the bottom, past the constructor, is the calculation function and the comparing function, which work rather simply. The calculator takes a car objects torque, RPM and weight to find the horsepower and acceleration, and uses all of these to find the theoretical quarter mile time (assuming no air resistance, perfect launch, good grip, etc), taking into account the type of tires a car has (street, racing, or drag), and drivetrain (front wheel drive, rear wheel drive, all wheel drive). These drivetrains have advantages and disadvantages, mostly having to do with power loss because of increasing complexity. These are simple modifiers, and are used as such.

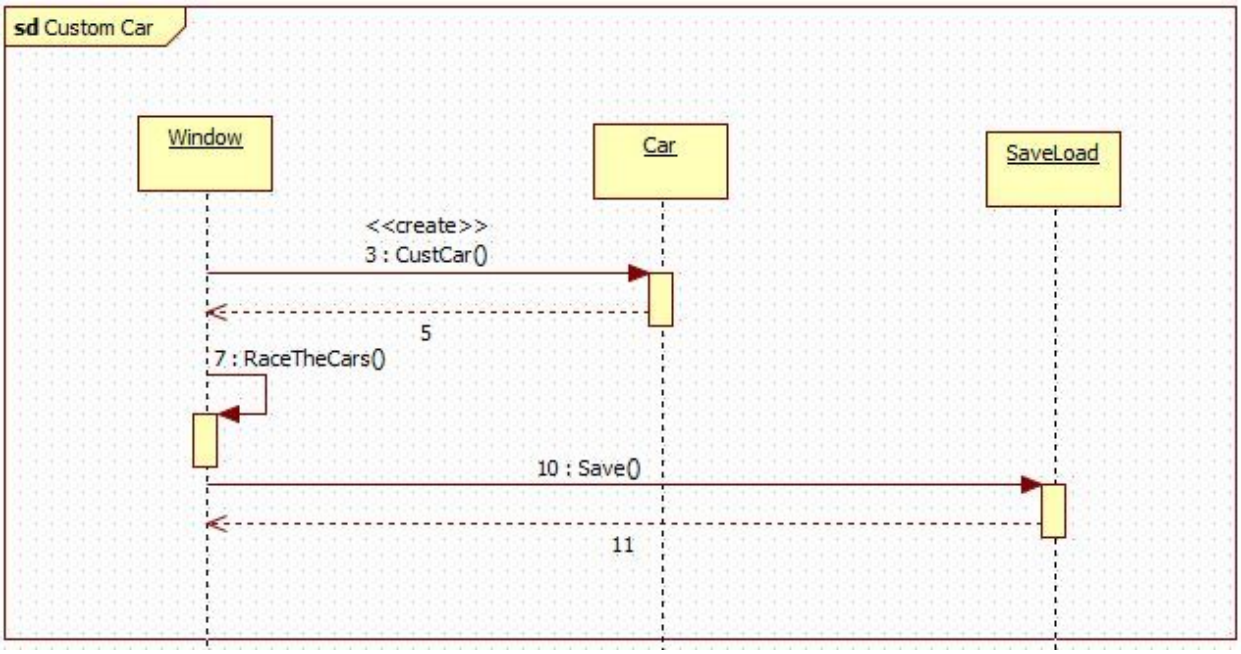
The compare cars function takes two cars, calculates both of their quarter mile times, and prints the fastest time and the car that did it to the race panel, for easy viewing. It is very simple, it receives two cars either from the car selector or a custom car and a car selector and then it compares them.

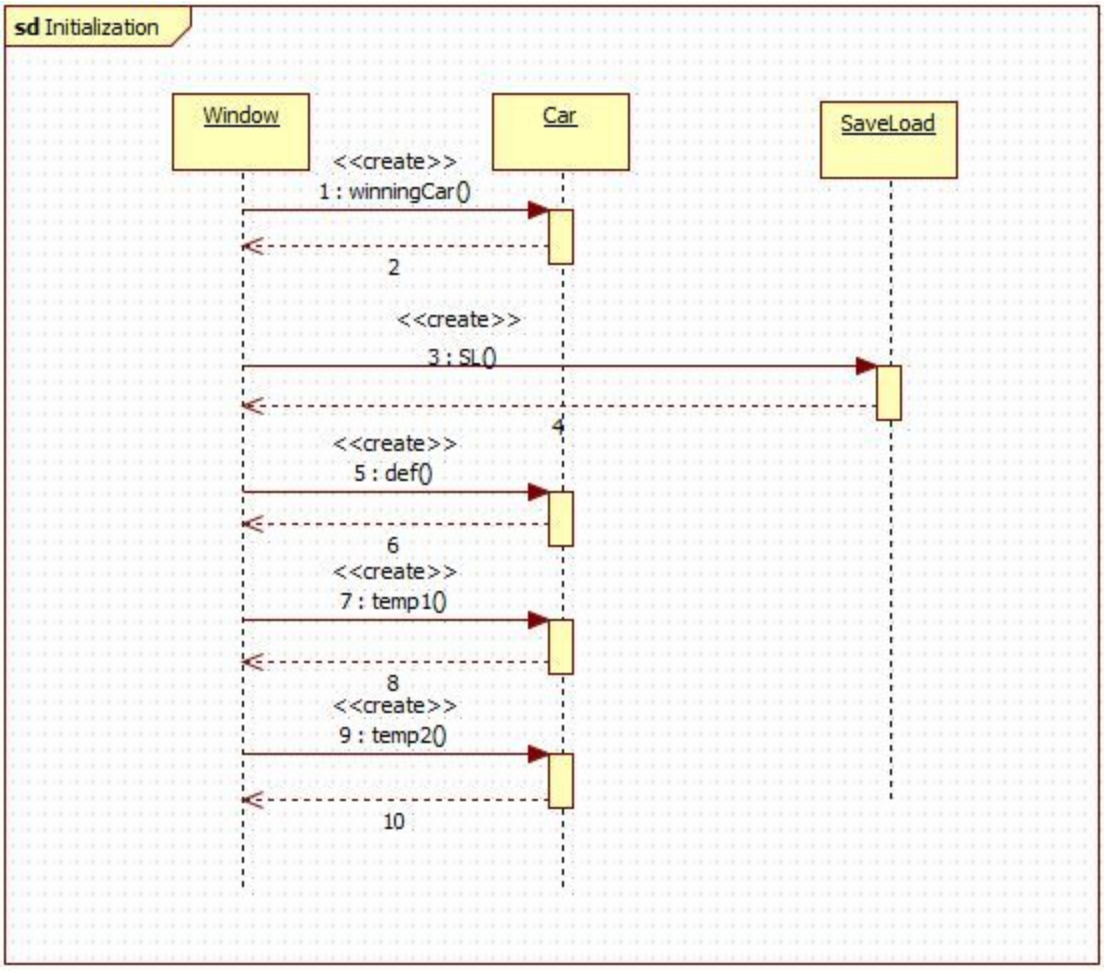
The window was fairly difficult to set up, but is very user friendly. There is a start screen, styled after the Gulf livery, on which a single button resides, purely for effect because I wanted to make one. Once clicked, the frame switches to the next card, the compare cars screen. On this screen, styled after the Red Bull Racing livery, the user can pick two cars from a list of cars and click race, whereby the actionlistener throws the two picked cars into the comparator and so on.

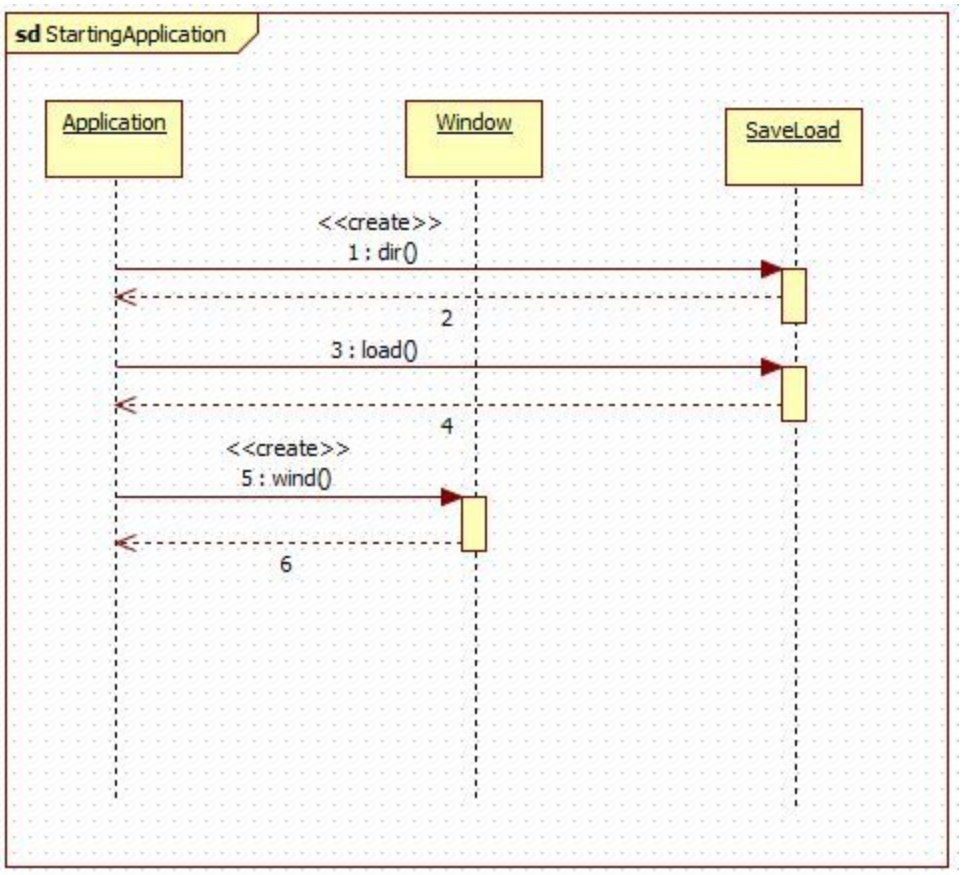
If, however, the user clicks on the custom button, they are taken to the custom car card, styled after the BMW ///M livery. The user can here enter the stats of the car they would like to compare and add, and to make one they must put an RPM, Torque, Weight, Name and Manufacturer. The user can then choose a car to race against their custom car, from a list of cars. The list is then saved, so that the user can recall the car they just put in whenever they feel like comparing cars again.

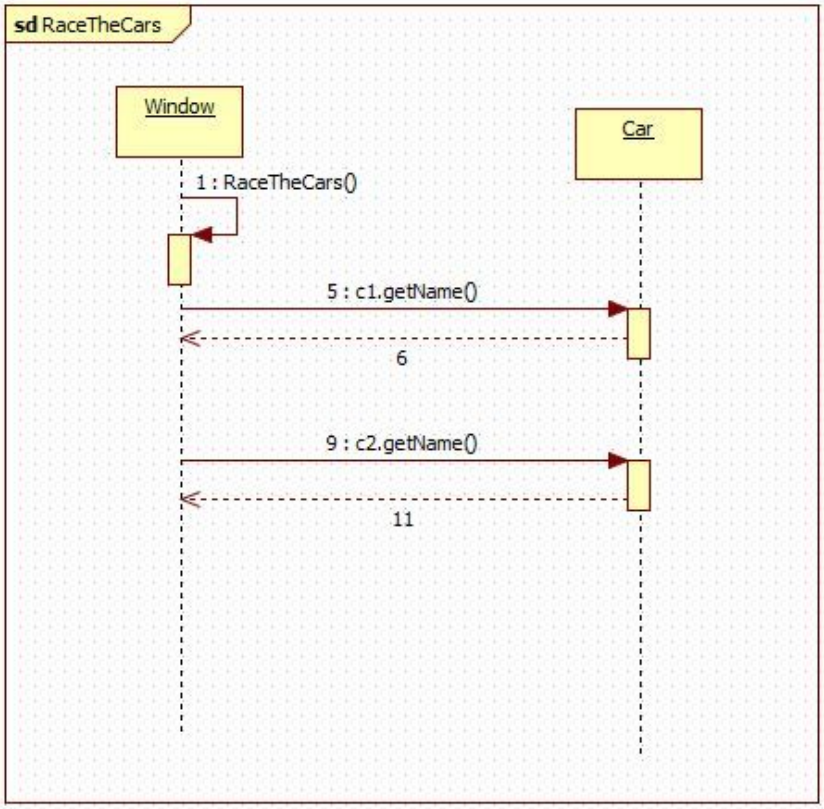
Regardless of whether the user choose to make a car or pick two cars, they are finally taken to the race screen, which prints out the name and quarter mile time of the winning car. The end.


<<Java Class>>
Window
(default package)
WinningTime: double
menu: JFrame
carList: String[]
df: DecimalFormat
NameOfCar: String
MakerOfCar: String
WheelsDrivenByCar: int
TiresOnCar: int
WeightOfCar: int
RPMOfCar: int
TorqueOfCar: int
start: JButton
background: JLabel
Image1: ImageIcon
StartMenuBackground: JLabel
startRace: JButton
customWindow: JButton
carName: JComboBox<String>
carName2: JComboBox<String>
Image2: ImageIcon
carSelectHeader: JLabel
Image3: ImageIcon
startRaceCust: JButton
curbWeight: JLabel
revLimit: JLabel
peakTorque: JLabel
wheelDrive: JLabel
tireType: JLabel
versus: JLabel
customCarBackground: JLabel
weight: JTextField
rpm: JTextField
torque: JTextField
Name: JTextField
Manufacturer: JTextField
tires: JComboBox
wheelsDriven: JComboBox
otherCar: JComboBox
carName3: JComboBox
winnerName: JLabel
winnerTime: JLabel
Image4: ImageIcon
raceBackground: JLabel
container: JPanel
startScreen: JPanel
carSelect: JPanel
customCar: JPanel
race: JPanel
Window()
actionPerformed(ActionEvent):void
Custom(String,String,int,int,int,int,int,Car):Car
RaceTheCars(Car,Car):Car
CalcSpeed(Car):double
<<Java Class>>
SaveLoad
(default package)
SaveLoad()
Save(ArrayList<Car>):void
Load():void
<<Java Class>>
Application
(default package)
Application()
main(String[]):void
~SL
0..1
<<Java Class>>
Car
(default package)
Name: String
Brand: String
weight: int
torque: int
RPM: int
WD: int
Tires: int
Car()
Car(String,String,int,int,int,int,int)
getName():String
~winningCar
~list
0..*
<<Java Enumeration>>
Wheels
(default package)
Front: Wheels
Rear: Wheels
All: Wheels
Wheels()
<<Java Enumeration>>
Tires
(default package)
Street: Tires
Race: Tires
Drag: Tires
Tires()


sd Custom Car
Window
Car
SaveLoad
<<create>>
3 : CustCar()
5
7 : RaceTheCars()
10 : Save()
11


sd Initialization
Window
Car
SaveLoad
<<create>>
1 : winningCar()
2
<<create>>
3 : SL()
4
<<create>>
5 : def()
6
<<create>>
7 : temp1()
8
<<create>>
9 : temp2()
10


sd StartingApplication
Application
Window
SaveLoad
<<create>>
1 : dir()
2
3 : load()
4
<<create>>
5 : wind()
6


sd RaceTheCars
Window
Car
1 : RaceTheCars()
5 : c1.getName()
6
9 : c2.getName()
11