Unit 6: OOP & Typescript: Lab 3 – Race Cars

Assigned 12/15/22

Due 12/15/22

Completed 01/07/23

**LMS:** <https://lms.grandcircus.co/mod/assign/view.php?id=22794>

**Google Doc:** <https://docs.google.com/document/d/1_XWB03Mtxm6lDsPuoDCEIDj7pFS5RHbjlTI3l9V_d74/preview>

**GitHub:**

**Skills:** Interfaces with methods, classes implementing interfaces, polymorphism, testing

**Overview:** Using TypeScript, create and test interfaces and classes that represent different kinds of race cars. This challenge does not represent a complete program; these pieces could be used as part of a larger TypeScript program.

**Build Specifications**

* You will create a Racer interface and two classes that implement that interface: SolarCar and GasCar.
* Create each interface and class in its own file.
* Each class has required Jest test cases that you must write.
* You will also create three functions that work with these interfaces and classes.
* All three functions are written in one file and also have required Jest test cases.

**Racer Interface**

In its own file, create an interface named Racer and export it.

* Properties:
  + team (a string)
  + speed (a number)
* Methods:
  + accelerate: Has no parameters. It returns void.
  + isFuelEmpty: Has no parameters. It returns Boolean.
* Jest Tests: N/A - We won't be writing tests directly for our interfaces.

**SolarCar Class**

In its own file, create a class named SolarCar and export it.

* Implements the Racer interface.
* Properties:
  + team (a string)
  + speed (a number) - This property **ALWAYS STARTS at 0**.
* Constructor Parameters:
  + team (a string) sets the team property
* Methods:
  + accelerate: Has no parameters. Adds **1** to the speed property.
  + isFuelEmpty: Has no parameters. Always returns **false**.
* Jest Tests:
  + The team property is set from the constructor parameter.
  + The speed property starts at **0**.
  + Calling accelerate once brings speed to **1**.
  + Calling accelerate twice brings speed to **2**.
  + isFuelEmpty returns **false**.

**GasCar Class**

In its own file, create a class named GasCar and export it.

* Implements the Racer interface.
* Properties:
  + team (a string)
  + speed (a number) - This property **ALWAYS STARTS at 0**.
  + fuel (a number)
* Constructor Parameters:
  + team (a string) sets the team property
  + fuel (a number) sets the fuel property. This parameter is optional and has a default value of **10**.
* Methods:
  + accelerate: Has no parameters. Adds **2** to the speed property and subtracts **1** from the fuel property.
  + isFuelEmpty: Has no parameters. Returns true if fuel is less than or equal to **0**. Otherwise returns false.
* Jest Tests:
  + The team and fuel properties are set from the constructor parameters.
  + fuel defaults to **10**, when the second constructor parameter is omitted.
  + The speed property starts at **0**.
  + Calling accelerate once brings speed to **2**.
  + Calling accelerate twice brings speed to **4**.
  + Calling accelerate once reduces fuel by **1**.
  + Calling accelerate twice reduces fuel by **2**.
  + isFuelEmpty returns true when fuel is **0**.
  + isFuelEmpty returns false when fuel is **greater than 0**.

**findRacersWithEmptyFuel Function**

In the racer-functions.ts file, create a function named findRacersWithEmptyFuel and export it.

* Parameters:
  + racers (an array of Racer)
* Returns: an array of Racer
* Functionality: Filter the racers array to find only the racers for which isFuelEmpty returns true. Return a new array of those matching racers.
* Jest Tests: For each test case, create an array of Racers. Call findRacersWithEmptyFuel with this array and confirm the correct result.
  + Do a test case with an array of GasCar where some have no fuel.
  + Do a test case with an array of GasCar where all have no fuel.
  + Do a test case with an array of GasCar where none have no fuel. (Expect an empty array as the result.)
  + Do a test case with an array of SolarCar. (Expect an empty array as the result.)
  + Do a test case with an array that has a mix of SolarCar and GasCar.
  + Do a test case with an empty array. (Expect an empty array as the result.)

**findAverageSpeed Function**

In the racer-functions.ts file, create a function named findAverageSpeed and export it.

* Parameters:
  + racers (an array of Racer)
* Returns: a number
* Functionality: Add up the speeds of all the racers in the array and divide by the length of the array. That will give you the average speed. Return the result.
  + In case of an empty array, return 0.
* Jest Tests: For each test case, create an array of Racers. Call findAverageSpeed with this array and confirm the correct result.
  + Do a test case with an array of GasCar with different speeds.
  + Do a test case with an array that has a mix of SolarCar and GasCar with different speeds.
  + Do a test case with an array of cars that all have **0** speed. (Expect **0**.)
  + Do a test case with an empty array. (Expect **0**.)

**getFasterRacer Function**

In the racer-functions.ts file, create a function named getFasterRacer and export it.

* Parameters:
  + racerA (a Racer)
  + racerB (a Racer)
* Returns: a Racer or null
* Functionality: Determines which of the two racer arguments has the faster speed. Return the faster Racer. If both have the same speed, return null.
* Jest Tests:
  + Test with racerA faster.
  + Test with racerB faster.
  + Test with both racers the same speed.
  + Repeat one of the above with a different mix GasCar and SolarCar in the parameters