✓ 100 XP

Exercise - Extend a class

5 minutes

In this exercise, you'll extend the `Car` class to create a new class called `ElectricCar` and override a method.

1. Continue working in the Playground.
2. Below the `Car` class, create a new class called `ElectricCar` that `extends Car`.

TypeScript

```
class ElectricCar extends Car {
    // Properties unique to ElectricCar

    // Constructor

    // Accessors

    // Methods

}
```

3. Declare the property that is unique to the `ElectricCar` class, `_range`, as a `private` property of type `number`.

TypeScript

```
// Properties
private _range: number;
```

4. The `constructor` for the subclass is different from the `constructor` for the base class in a few ways.

   - The parameter list can include any of the properties of both the base class and the subclass. (As with all parameter lists in TypeScript, remember that required parameters must appear before optional parameters.)
   - In the body of the `constructor`, you must add the `super()` keyword to include the

parameters from the base class. The `super` keyword executes the `constructor` of the base class when it runs.
- The `super` keyword must appear before any references to `this.` when referring to properties in the subclass.

5. Define the class `constructor` for `ElectricCar`, including the `_make`, `_color`, and `_doors` properties of the base class and the `_range` property of the subclass. In this `constructor`, set the default value of the `doors` parameter to `2`.

TypeScript

```typescript
// Constructor
constructor(make: string, color: string, range: number, doors = 2) {
    super(make, color, doors);
    this._range = range;
}
```

6. Define the `get` and `set` accessors for the `range` parameter.

TypeScript

```typescript
// Accessors
get range() {
    return this._range;
}
set range(range) {
    this._range = range;
}
```

7. Enter the following `charge` method that returns a message to the console. This method includes a call to the `worker` function that you defined in the `Car` class. But it raises the error **Property 'worker' is private and only accessible within class 'Car'**. Do you know how to correct this problem?

TypeScript

```typescript
// Methods
charge() {
    console.log(this.worker() + " is charging.")
}
```

8. In the `Car` class, change the access modifier of the `worker` function from `private` to `protected`. This allows subclasses of the `Car` class to use the function, while keeping it hidden from the members available to objects instantiated from the class. The error in the `charge` method should now resolve.

9. Test the new `ElectricCar` class to verify that it's working as expected.

TypeScript

```
let spark = new ElectricCar('Spark Motors','silver', 124, 2);
let eCar = new ElectricCar('Electric Car Co.', 'black', 263);
console.log(eCar.doors);         // returns the default, 2
spark.charge();                  // returns "Spark Motors is charging"
```

10. Define a new `brake` method in the `ElectricCar` class that has different implementation details. Note that the parameter signature and return type for the `brake` method must be the same as the `brake` method in the `Car` class.

TypeScript

```
// Overrides the brake method of the Car class
brake(): string {
    return `${this.worker()}  is braking with the regenerative braking sys-
tem.`
}
```

11. Test the new method and verify that it works as expected.

TypeScript

```
console.log(spark.brake());  // returns "Spark Motors is braking with the re-
generative braking system"
```

Next unit: Exercise - Declare an interface to ensure class shape

Continue >