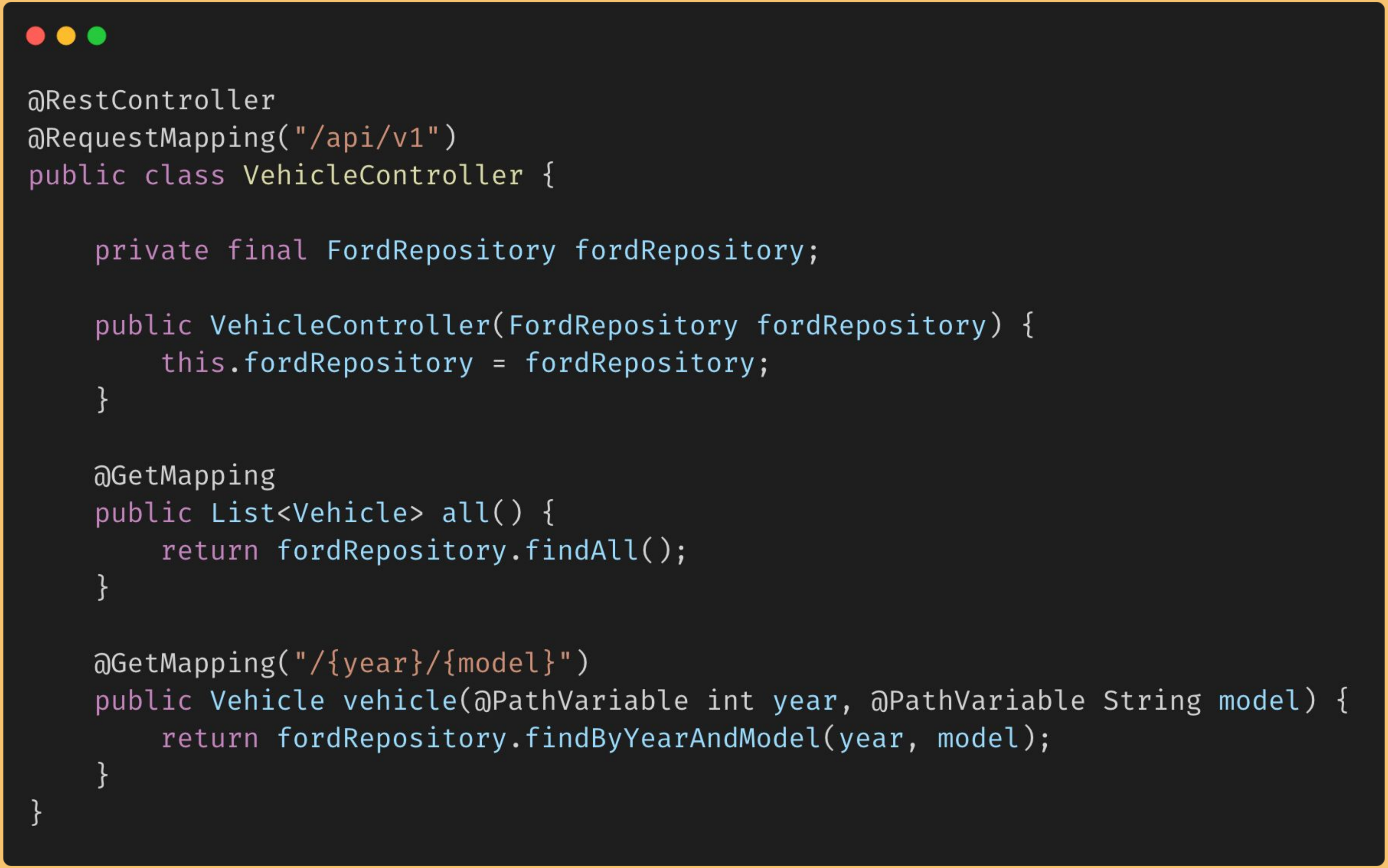
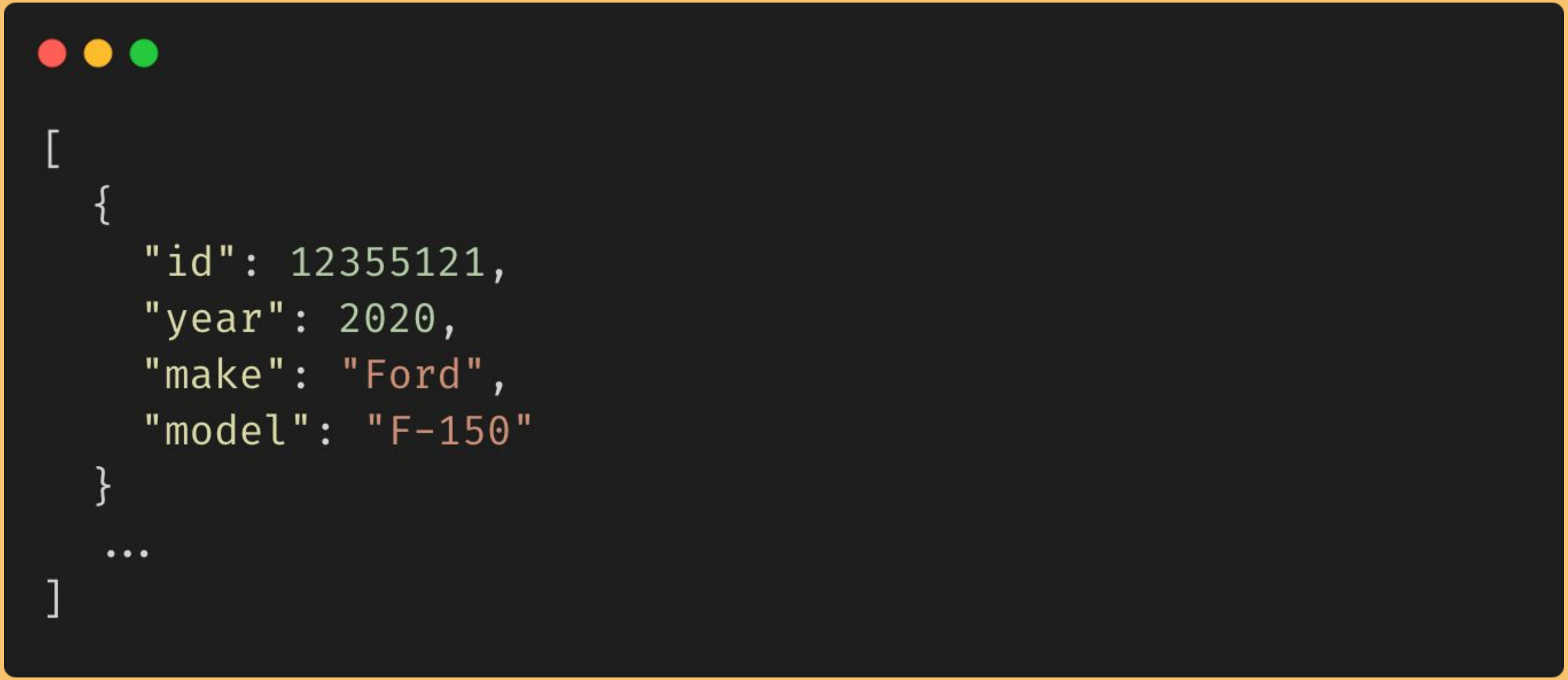
Flexibility Through Interfaces

Task 1:

Create an API that returns information about Ford vehicles



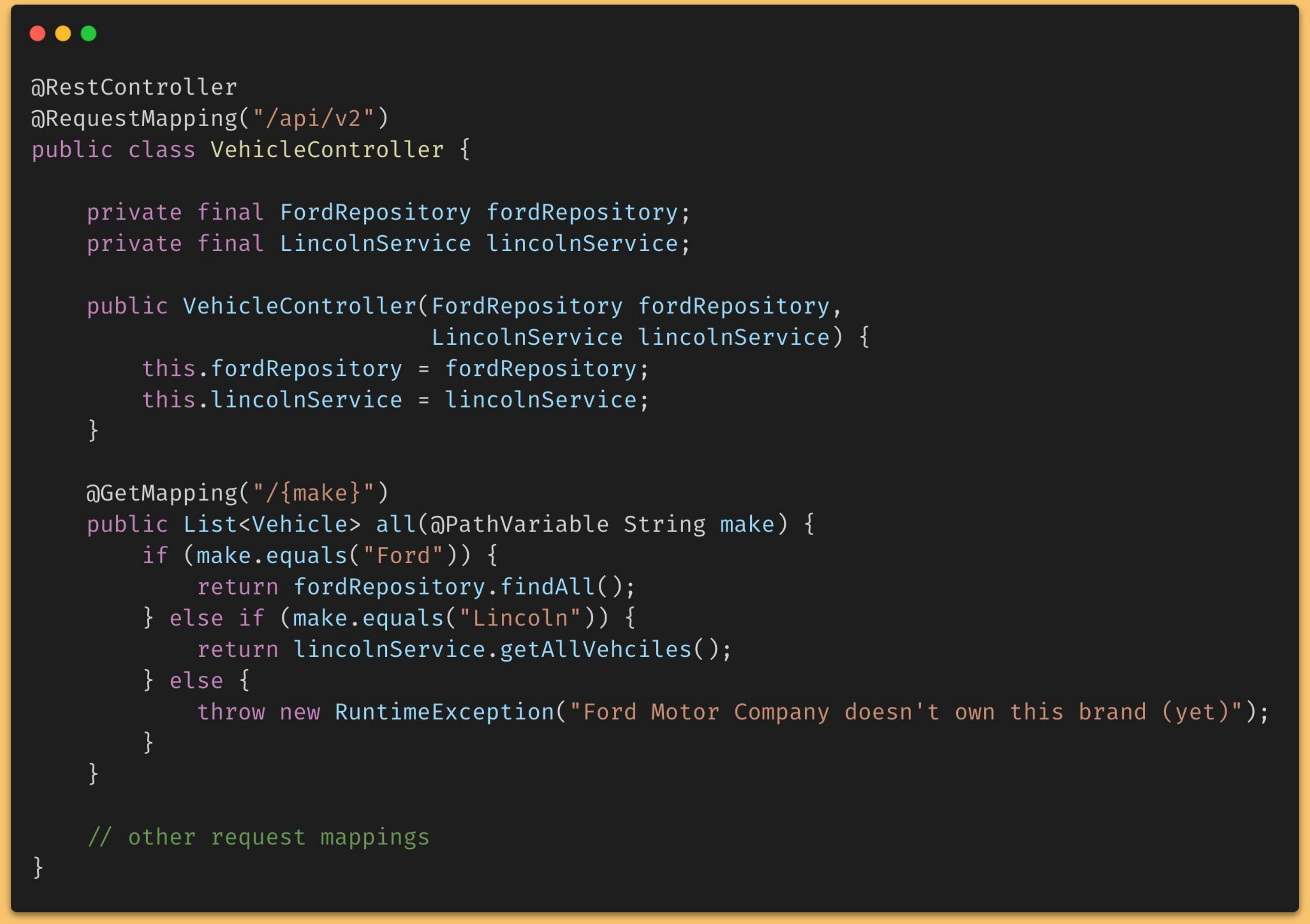
```java
@RestController
@RequestMapping("/api/v1")
public class VehicleController {

    private final FordRepository fordRepository;

    public VehicleController(FordRepository fordRepository) {
        this.fordRepository = fordRepository;
    }

    @GetMapping
    public List<Vehicle> all() {
        return fordRepository.findAll();
    }

    @GetMapping("/{year}/{model}")
    public Vehicle vehicle(@PathVariable int year, @PathVariable String model) {
        return fordRepository.findByYearAndModel(year, model);
    }
}
```



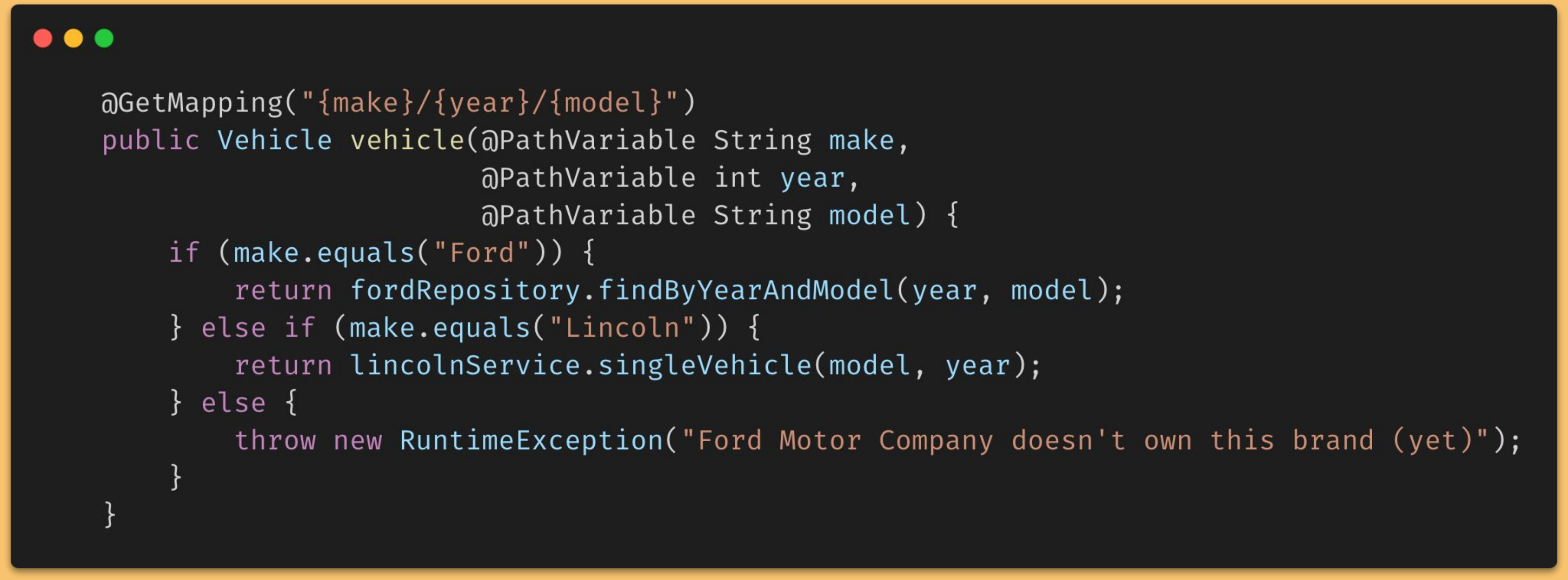
```json
[
  {
    "id": 12355121,
    "year": 2020,
    "make": "Ford",
    "model": "F-150"
  }
  …
]
```

Task 2:

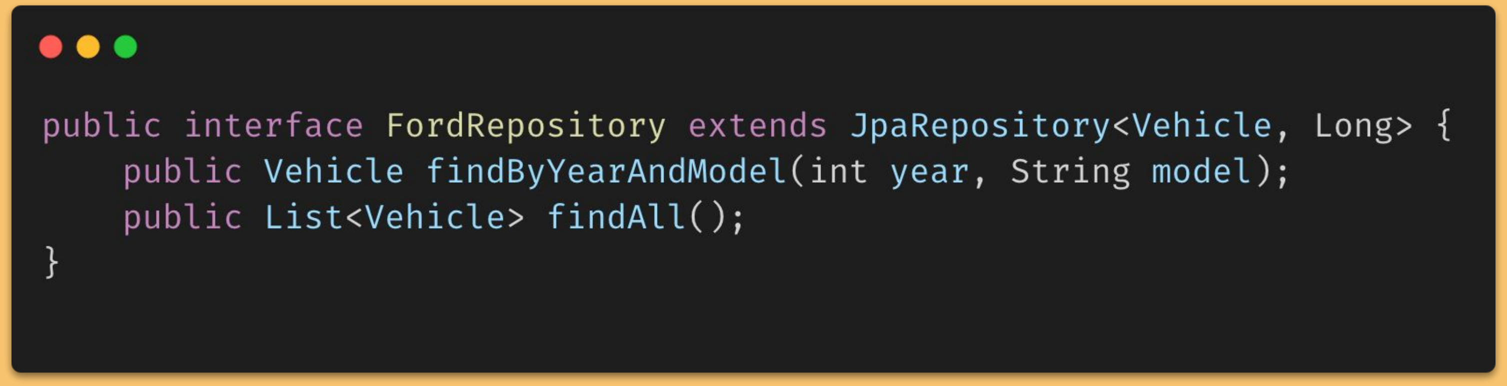
Support Lincoln vehicles too, fam



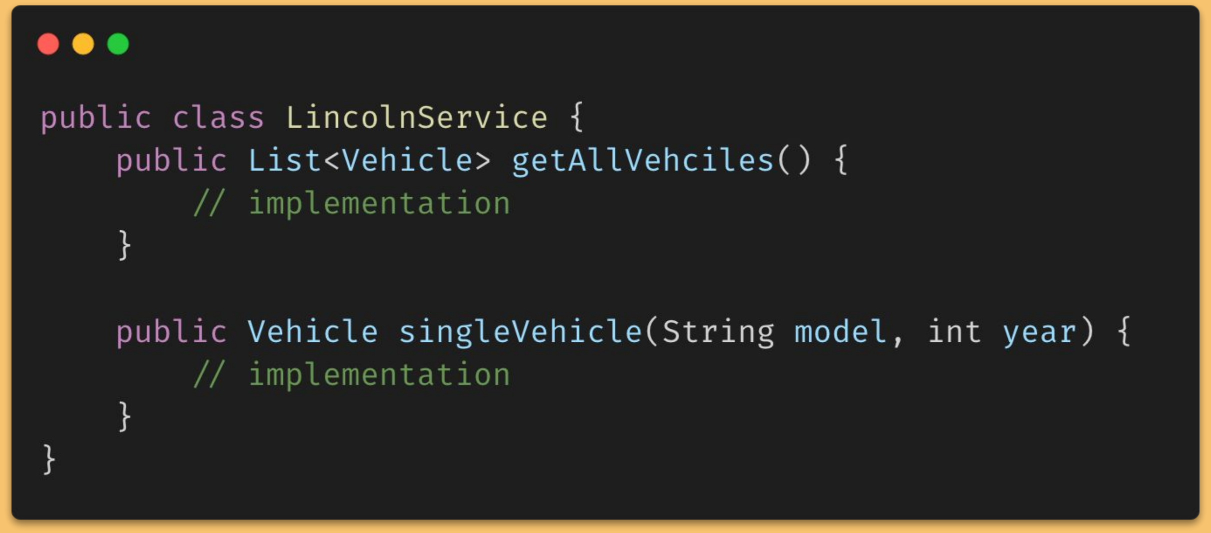
```java
@RestController
@RequestMapping("/api/v2")
public class VehicleController {

    private final FordRepository fordRepository;
    private final LincolnService lincolnService;

    public VehicleController(FordRepository fordRepository,
                             LincolnService lincolnService) {
        this.fordRepository = fordRepository;
        this.lincolnService = lincolnService;
    }

    @GetMapping("/{make}")
    public List<Vehicle> all(@PathVariable String make) {
        if (make.equals("Ford")) {
            return fordRepository.findAll();
        } else if (make.equals("Lincoln")) {
            return lincolnService.getAllVehciles();
        } else {
            throw new RuntimeException("Ford Motor Company doesn't own this brand (yet)");
        }
    }

    // other request mappings
}
```



```
@GetMapping("{make}/{year}/{model}")
public Vehicle vehicle(@PathVariable String make,
                       @PathVariable int year,
                       @PathVariable String model) {
    if (make.equals("Ford")) {
        return fordRepository.findByYearAndModel(year, model);
    } else if (make.equals("Lincoln")) {
        return lincolnService.singleVehicle(model, year);
    } else {
        throw new RuntimeException("Ford Motor Company doesn't own this brand (yet)");
    }
}
```



```java
public interface FordRepository extends JpaRepository<Vehicle, Long> {
    public Vehicle findByYearAndModel(int year, String model);
    public List<Vehicle> findAll();
}
```

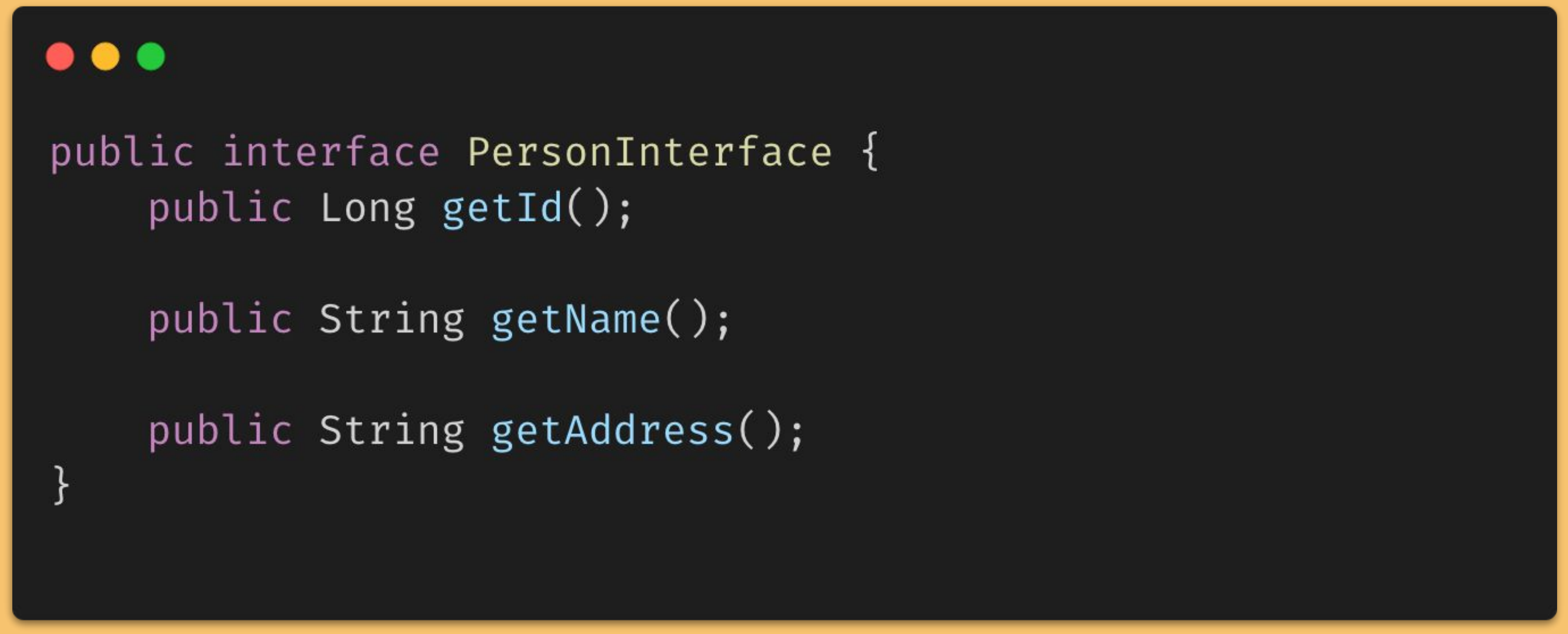


```java
public class LincolnService {
    public List<Vehicle> getAllVehciles() {
        // implementation
    }

    public Vehicle singleVehicle(String model, int year) {
        // implementation
    }
}
```

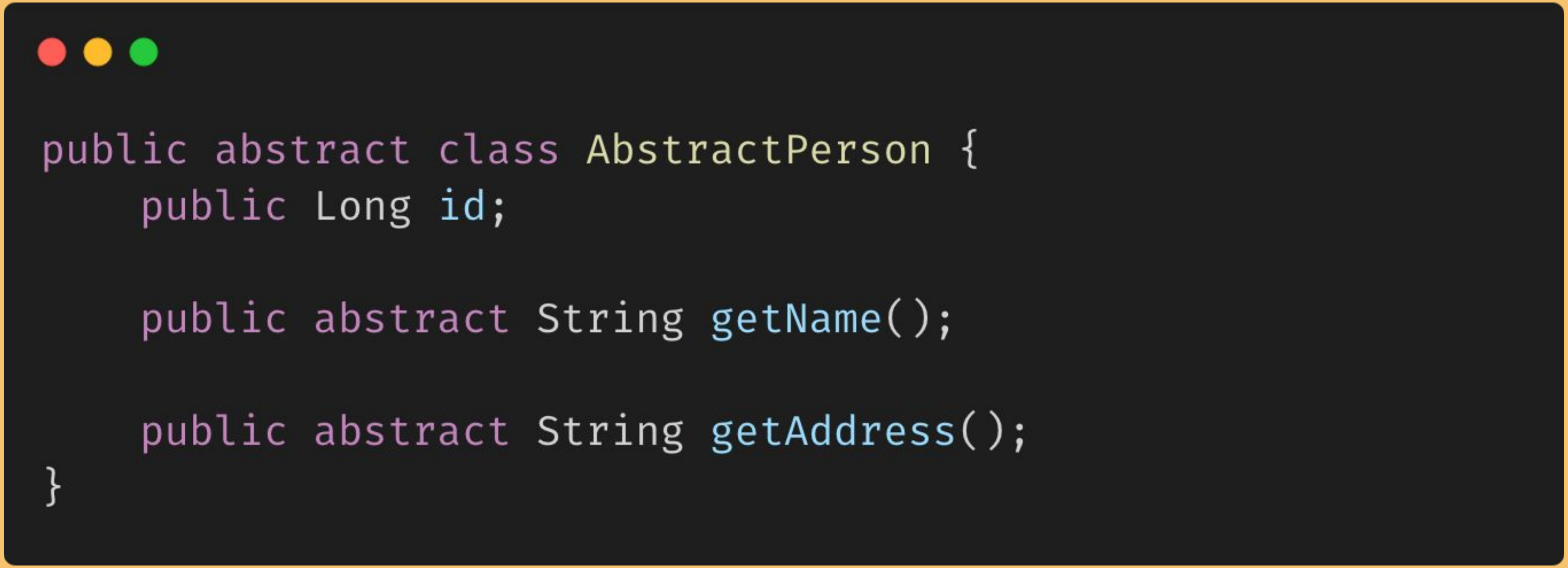
What exactly is an interface?

Interface

A class or module’s publicly accessible methods and attributes.

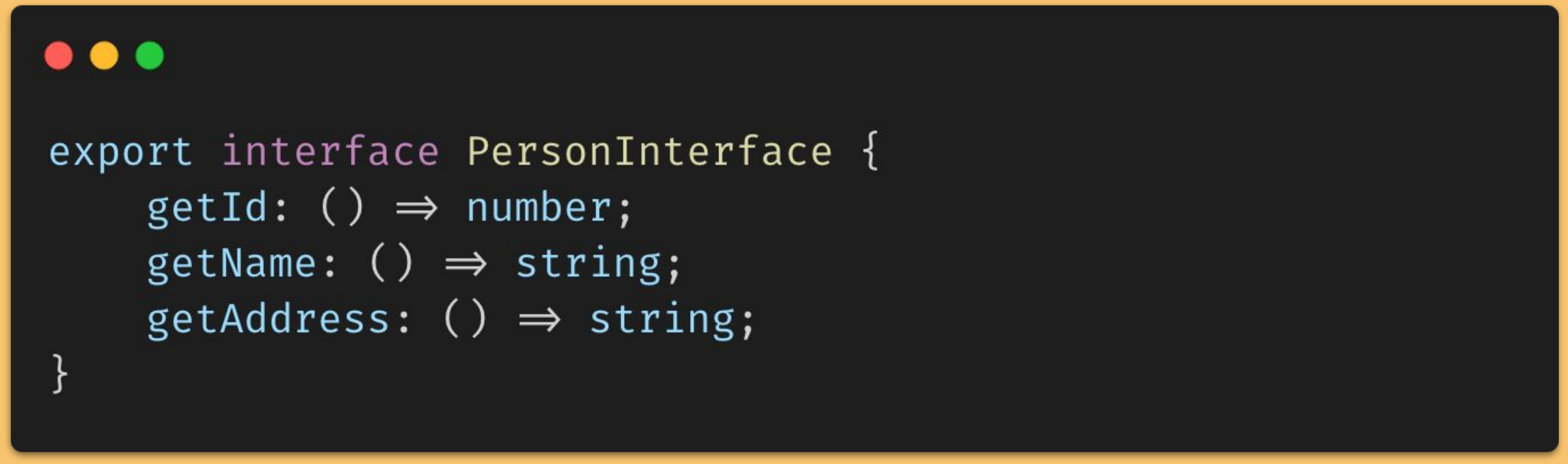


```java
public interface PersonInterface {
    public Long getId();

    public String getName();

    public String getAddress();
}
```

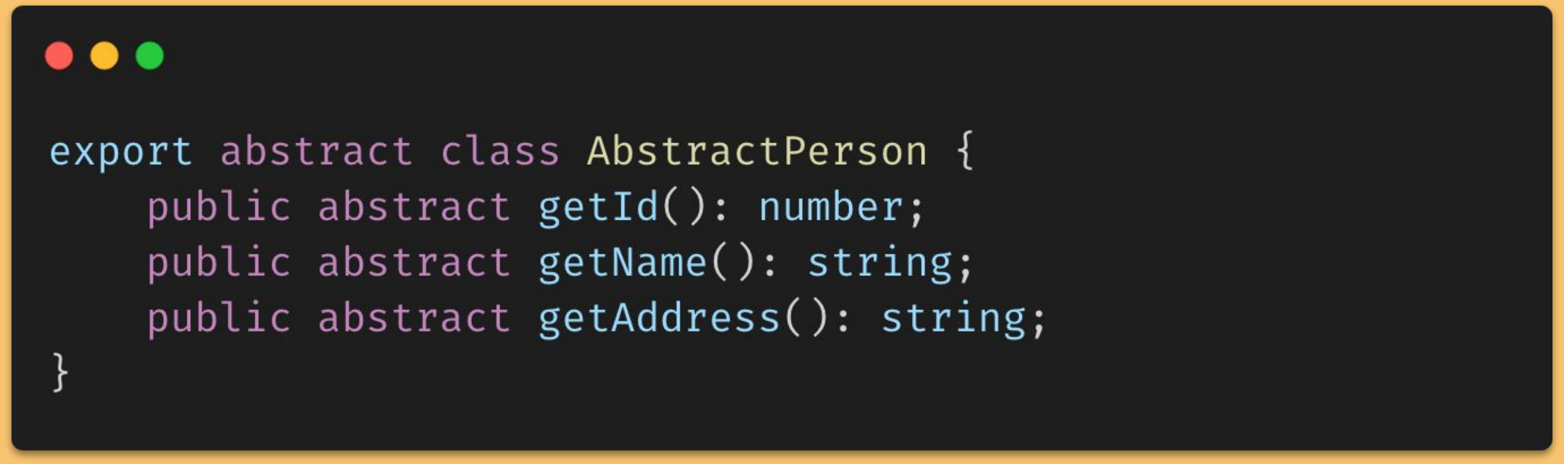


```java
public abstract class AbstractPerson {
    public Long id;

    public abstract String getName();

    public abstract String getAddress();
}
```

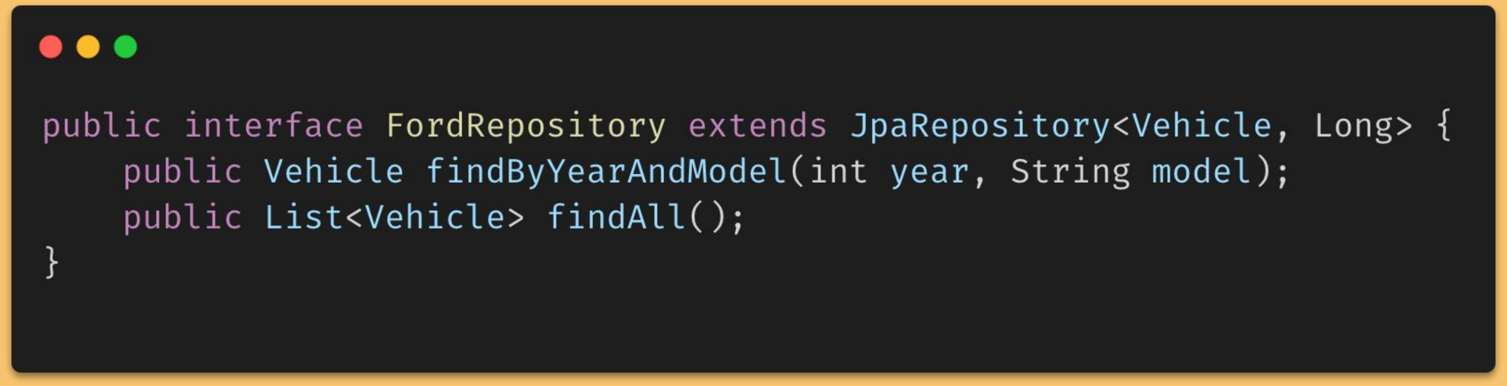
To be an interface you *do not* need to use the interface keyword



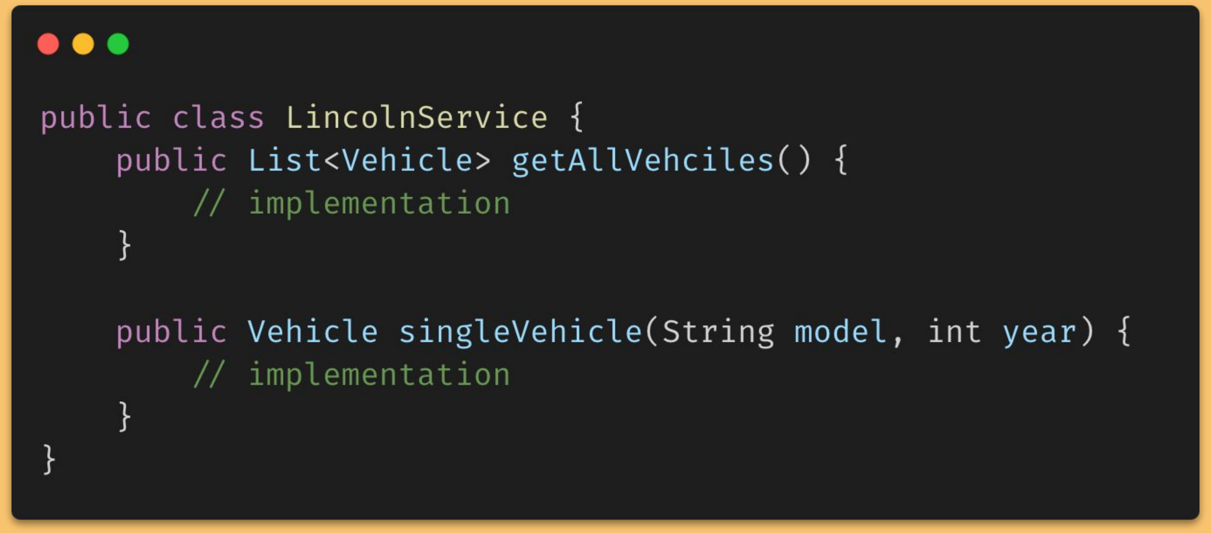
```typescript
export interface PersonInterface {
    getId: () => number;
    getName: () => string;
    getAddress: () => string;
}
```



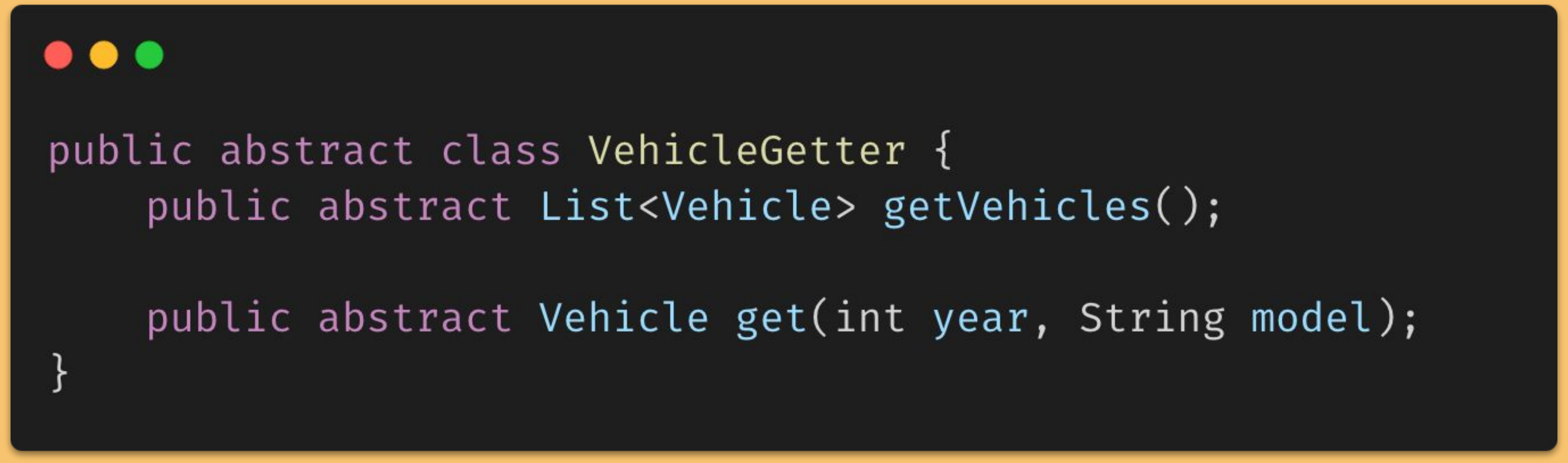
```typescript
export abstract class AbstractPerson {
    public abstract getId(): number;
    public abstract getName(): string;
    public abstract getAddress(): string;
}
```



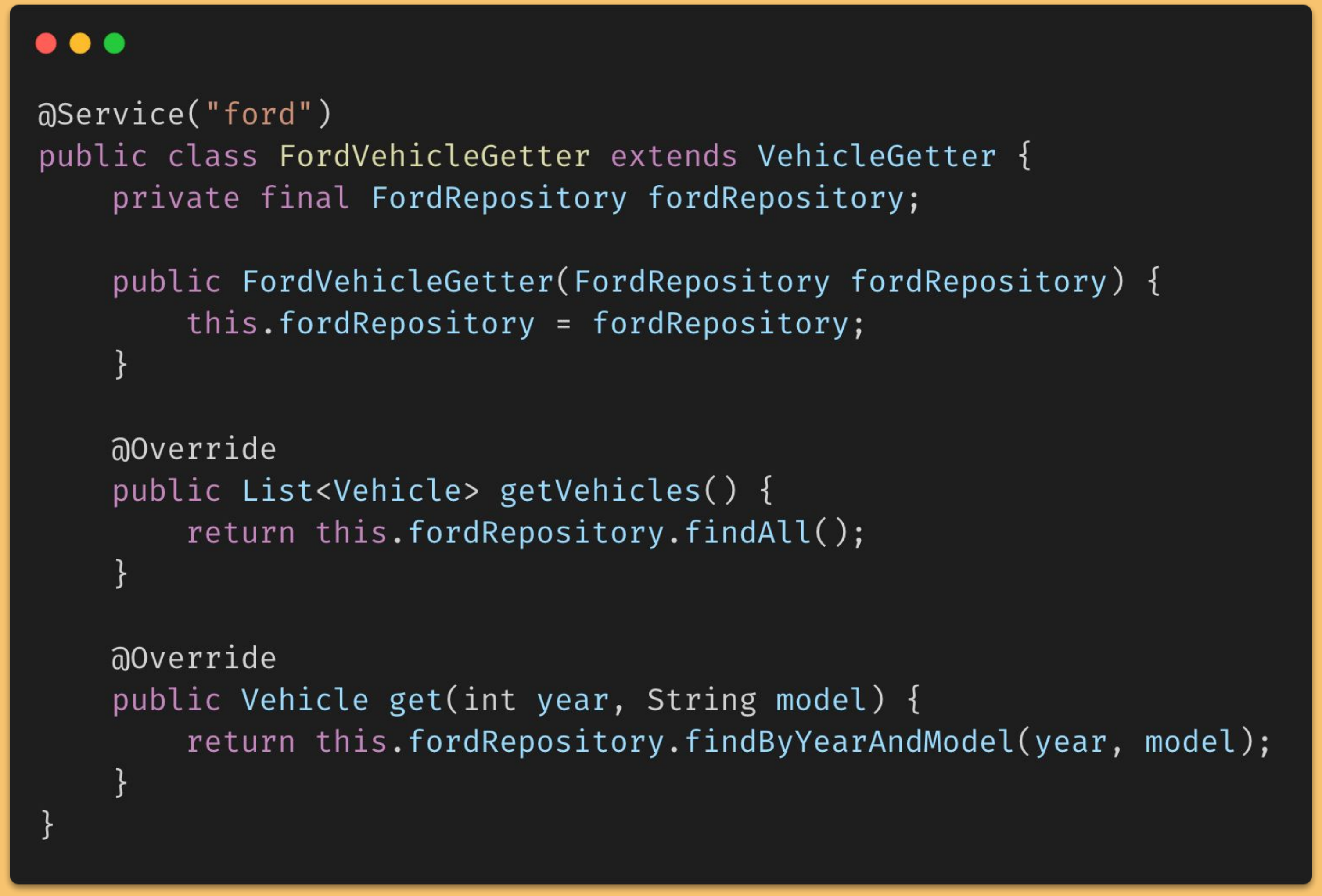
```
public interface FordRepository extends JpaRepository<Vehicle, Long> {
    public Vehicle findByYearAndModel(int year, String model);
    public List<Vehicle> findAll();
}
```



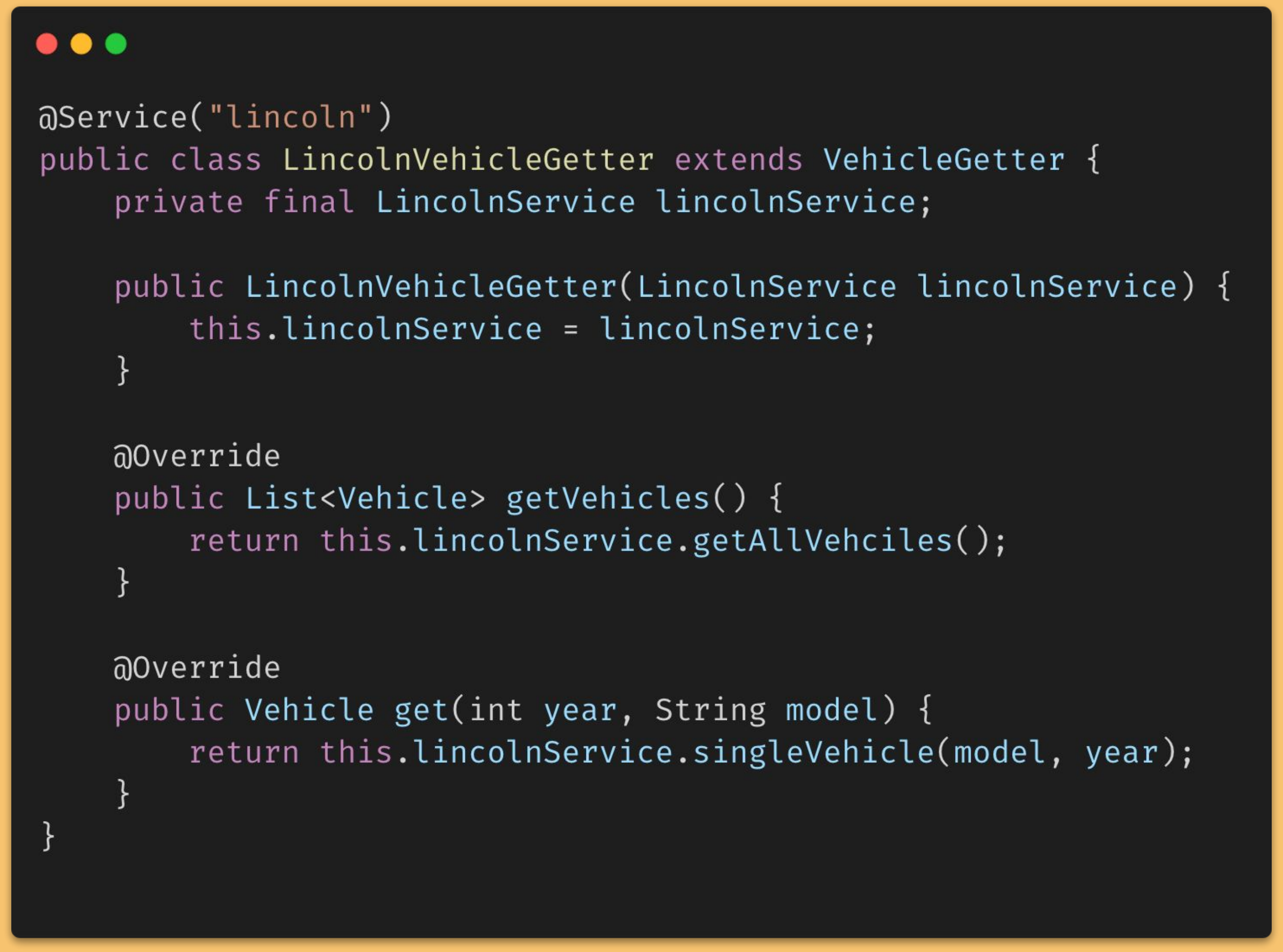
```
public class LincolnService {
    public List<Vehicle> getAllVehciles() {
        // implementation
    }

    public Vehicle singleVehicle(String model, int year) {
        // implementation
    }
}
```



```java
public abstract class VehicleGetter {
    public abstract List<Vehicle> getVehicles();

    public abstract Vehicle get(int year, String model);
}
```



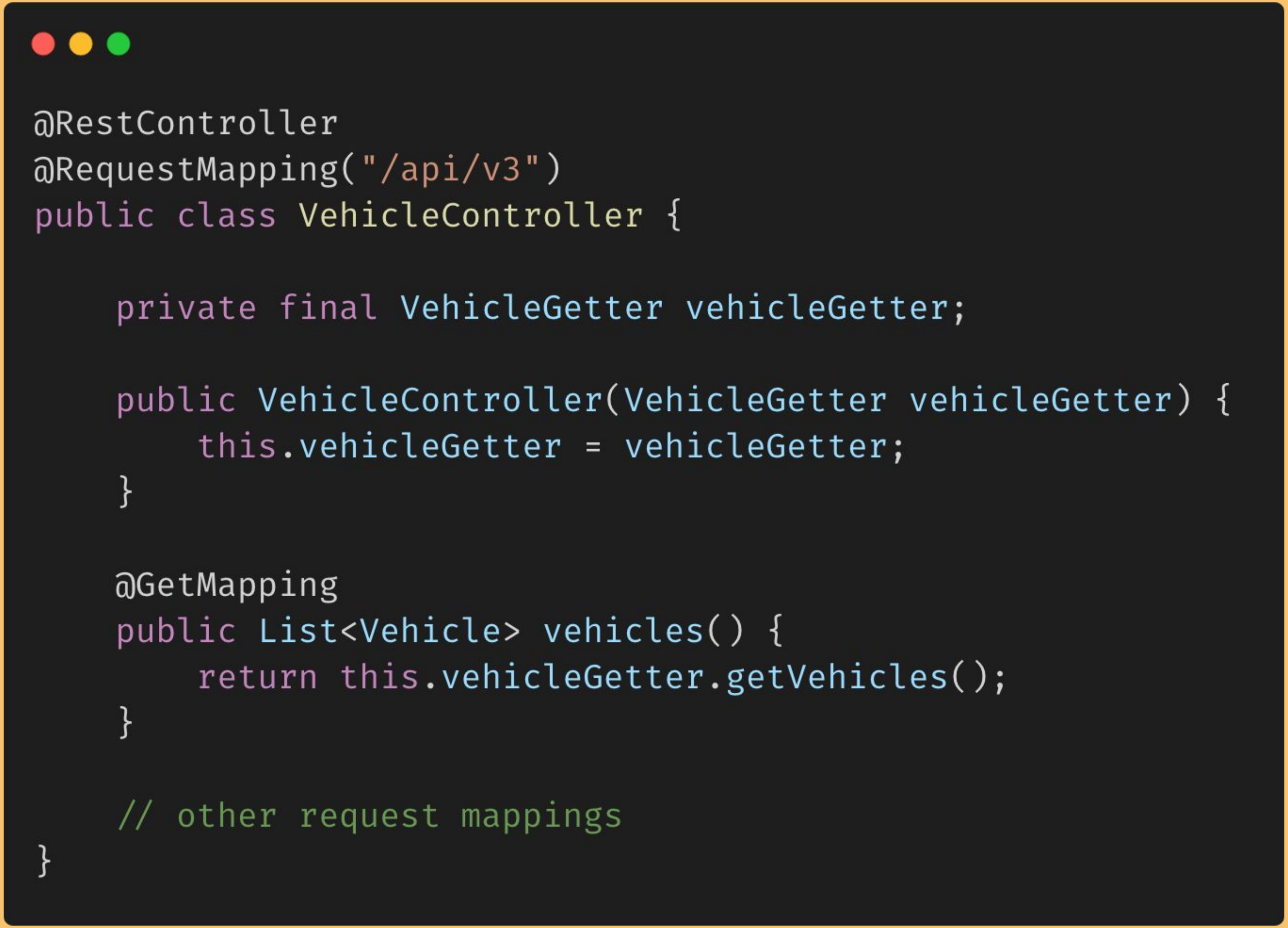
```java
@Service("ford")
public class FordVehicleGetter extends VehicleGetter {
    private final FordRepository fordRepository;

    public FordVehicleGetter(FordRepository fordRepository) {
        this.fordRepository = fordRepository;
    }

    @Override
    public List<Vehicle> getVehicles() {
        return this.fordRepository.findAll();
    }

    @Override
    public Vehicle get(int year, String model) {
        return this.fordRepository.findByYearAndModel(year, model);
    }
}
```



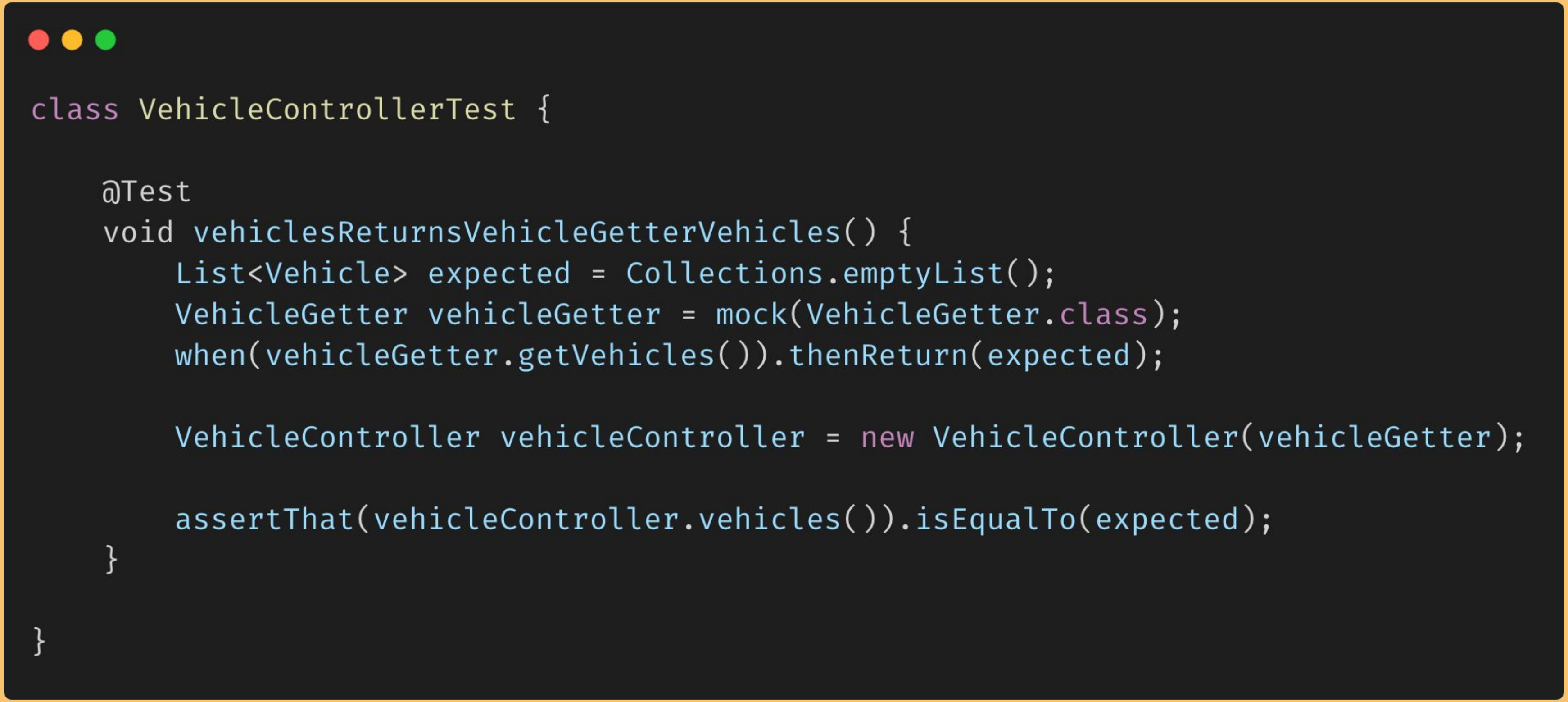
```java
@Service("lincoln")
public class LincolnVehicleGetter extends VehicleGetter {
    private final LincolnService lincolnService;

    public LincolnVehicleGetter(LincolnService lincolnService) {
        this.lincolnService = lincolnService;
    }

    @Override
    public List<Vehicle> getVehicles() {
        return this.lincolnService.getAllVehciles();
    }

    @Override
    public Vehicle get(int year, String model) {
        return this.lincolnService.singleVehicle(model, year);
    }
}
```

Oh yeah, so what?

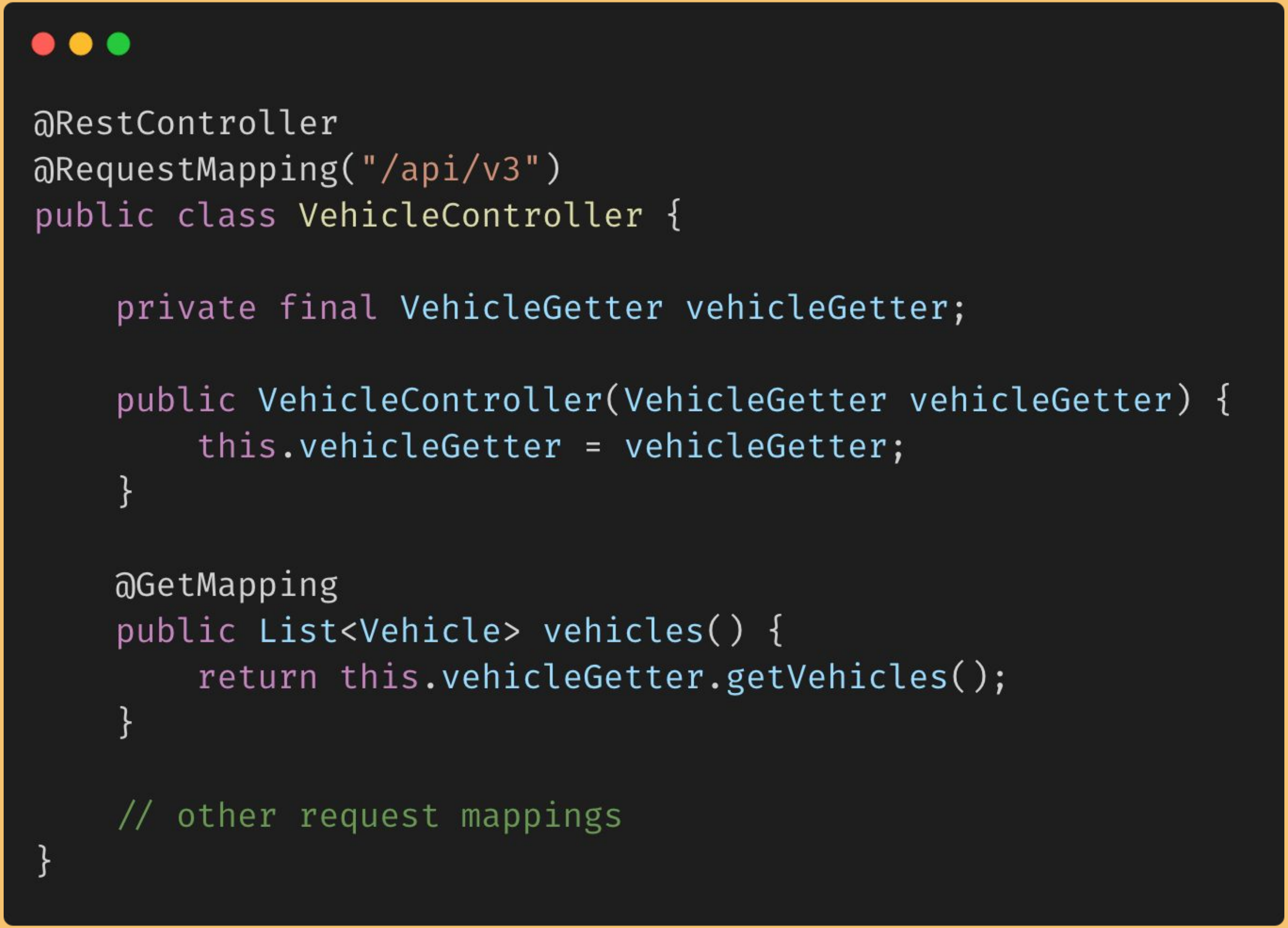
We can remove make specific dependencies from VehicleController



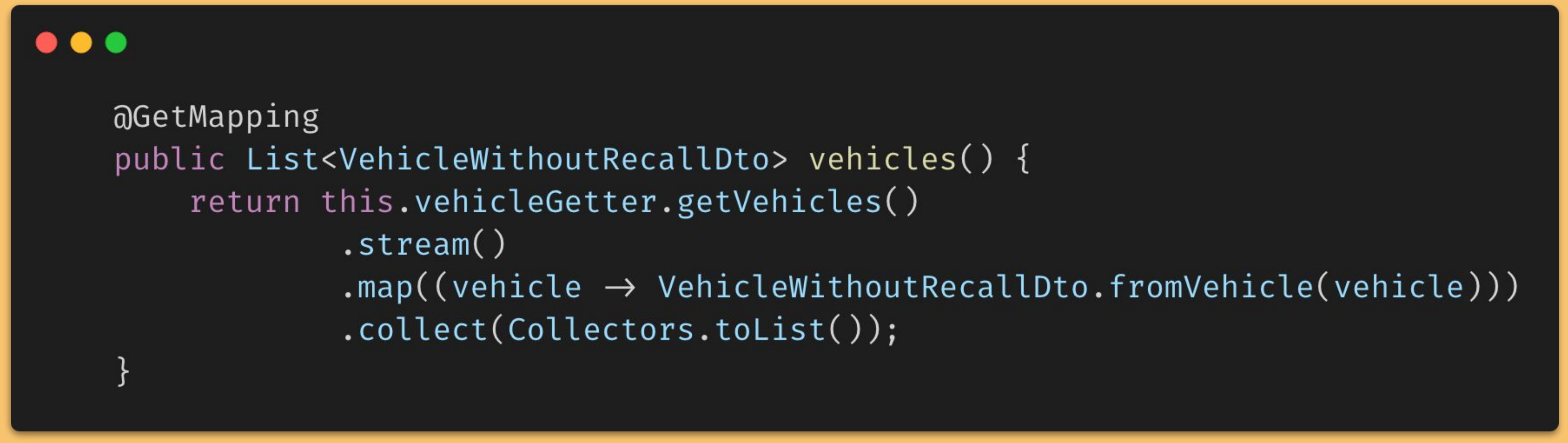
```java
@RestController
@RequestMapping("/api/v3")
public class VehicleController {

    private final VehicleGetter vehicleGetter;

    public VehicleController(VehicleGetter vehicleGetter) {
        this.vehicleGetter = vehicleGetter;
    }

    @GetMapping
    public List<Vehicle> vehicles() {
        return this.vehicleGetter.getVehicles();
    }

    // other request mappings
}
```



```java
class VehicleControllerTest {

    @Test
    void vehiclesReturnsVehicleGetterVehicles() {
        List<Vehicle> expected = Collections.emptyList();
        VehicleGetter vehicleGetter = mock(VehicleGetter.class);
        when(vehicleGetter.getVehicles()).thenReturn(expected);

        VehicleController vehicleController = new VehicleController(vehicleGetter);

        assertThat(vehicleController.vehicles()).isEqualTo(expected);
    }

}
```



```java
@RestController
@RequestMapping("/api/v3")
public class VehicleController {

    private final VehicleGetter vehicleGetter;

    public VehicleController(VehicleGetter vehicleGetter) {
        this.vehicleGetter = vehicleGetter;
    }

    @GetMapping
    public List<Vehicle> vehicles() {
        return this.vehicleGetter.getVehicles();
    }

    // other request mappings
}
```

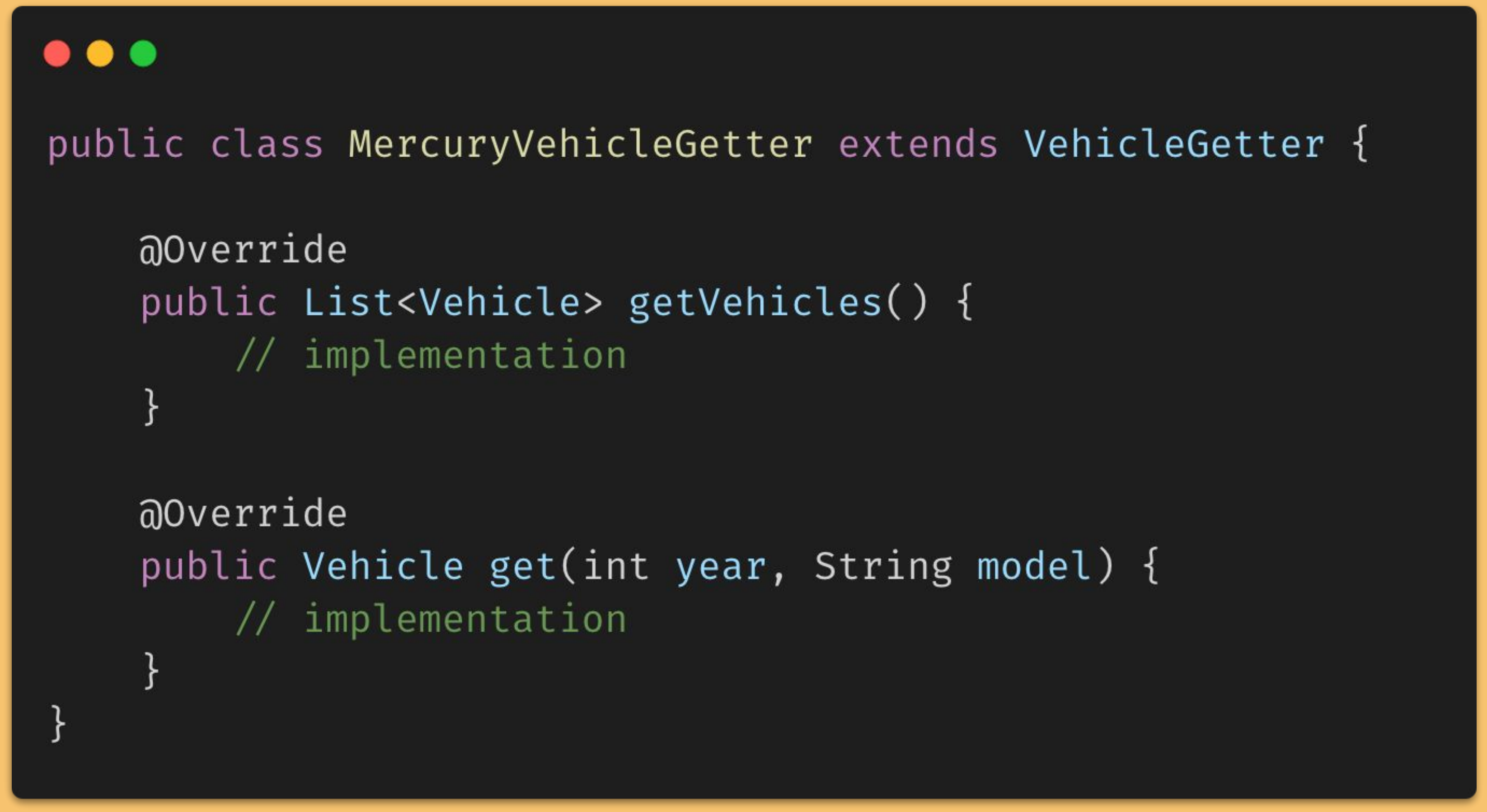


```java
@GetMapping
public List<VehicleWithoutRecallDto> vehicles() {
    return this.vehicleGetter.getVehicles()
            .stream()
            .map((vehicle -> VehicleWithoutRecallDto.fromVehicle(vehicle)))
            .collect(Collectors.toList());
}
```

We can add and remove as many makes as we want without changing VehicleController

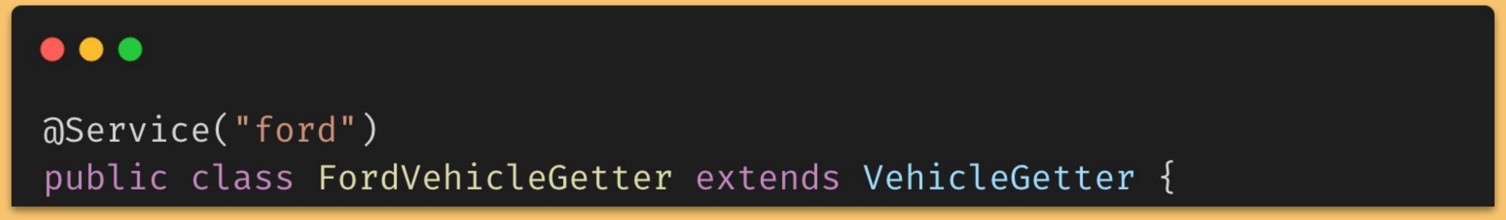
Task 3:

Bring Mercury back!

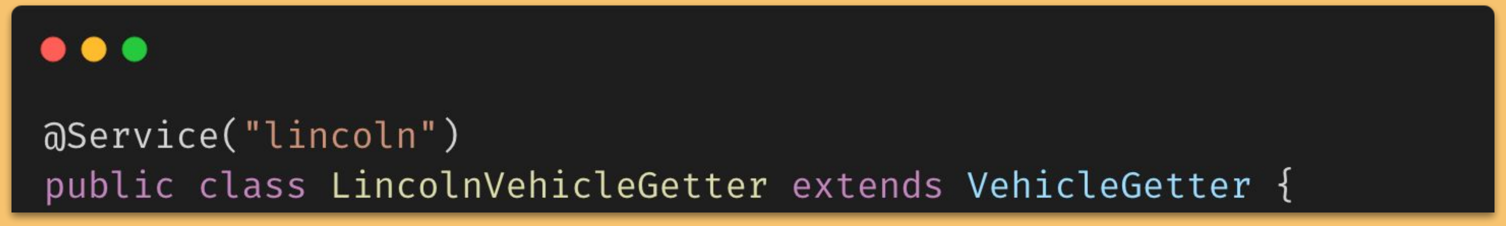


```java
public class MercuryVehicleGetter extends VehicleGetter {

    @Override
    public List<Vehicle> getVehicles() {
        // implementation
    }

    @Override
    public Vehicle get(int year, String model) {
        // implementation
    }
}
```

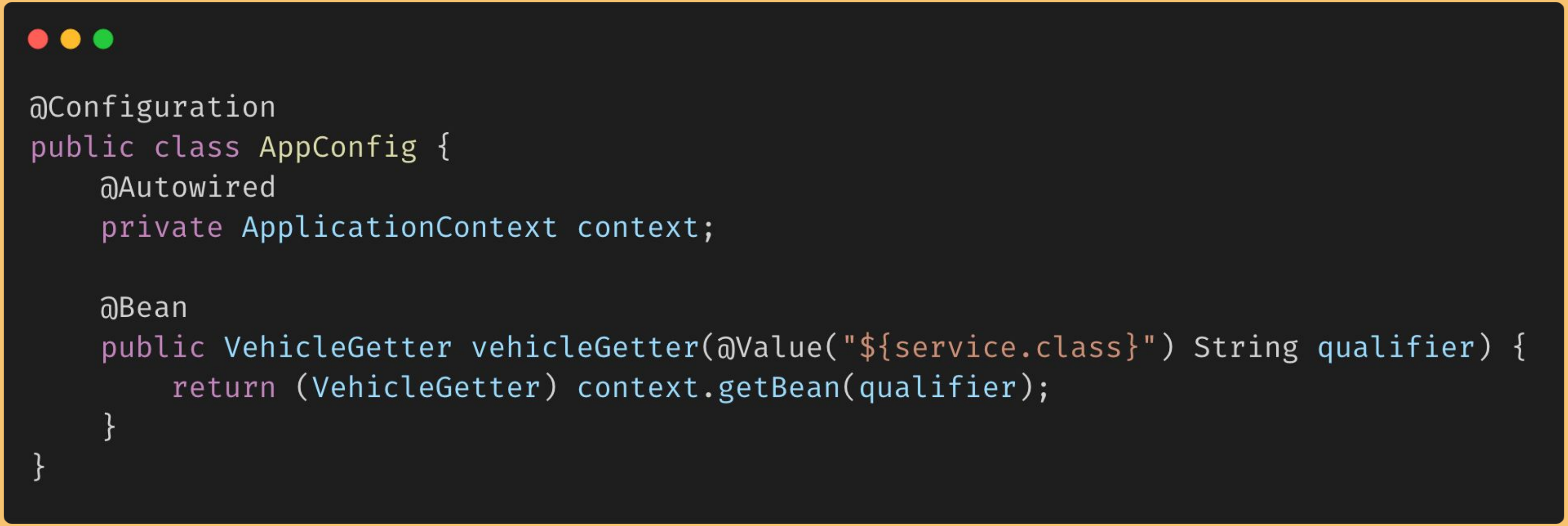
We still have to choose a make somewhere, don’t we?



```java
@Service("ford")
public class FordVehicleGetter extends VehicleGetter {
```

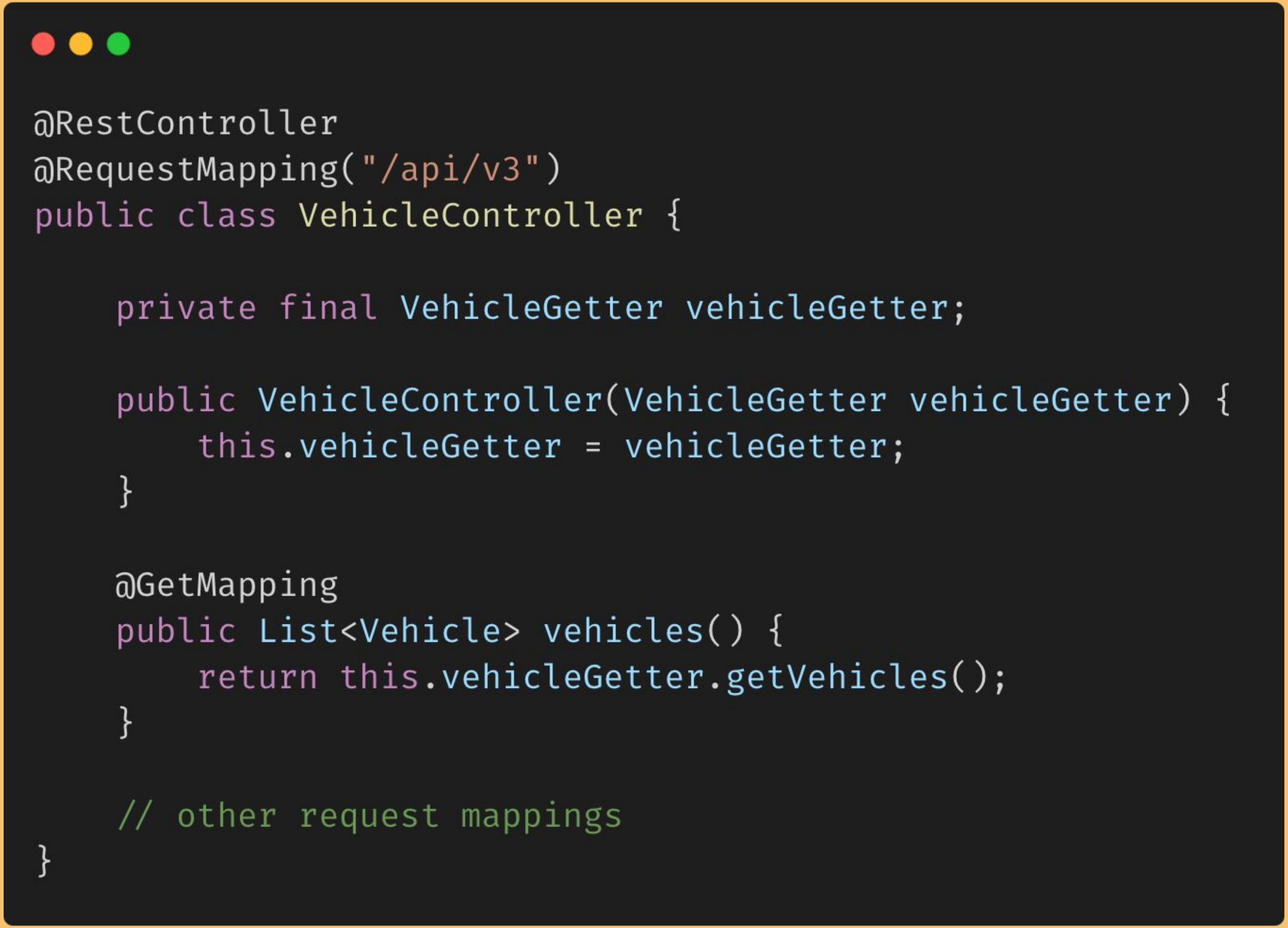


```java
@Service("lincoln")
public class LincolnVehicleGetter extends VehicleGetter {
```



```java
@Configuration
public class AppConfig {
    @Autowired
    private ApplicationContext context;

    @Bean
    public VehicleGetter vehicleGetter(@Value("${service.class}") String qualifier) {
        return (VehicleGetter) context.getBean(qualifier);
    }
}
```

```
service.class=ford
```



```java
@RestController
@RequestMapping("/api/v3")
public class VehicleController {

    private final VehicleGetter vehicleGetter;

    public VehicleController(VehicleGetter vehicleGetter) {
        this.vehicleGetter = vehicleGetter;
    }

    @GetMapping
    public List<Vehicle> vehicles() {
        return this.vehicleGetter.getVehicles();
    }

    // other request mappings
}
```

ford.com/api

```
service.class=ford
```

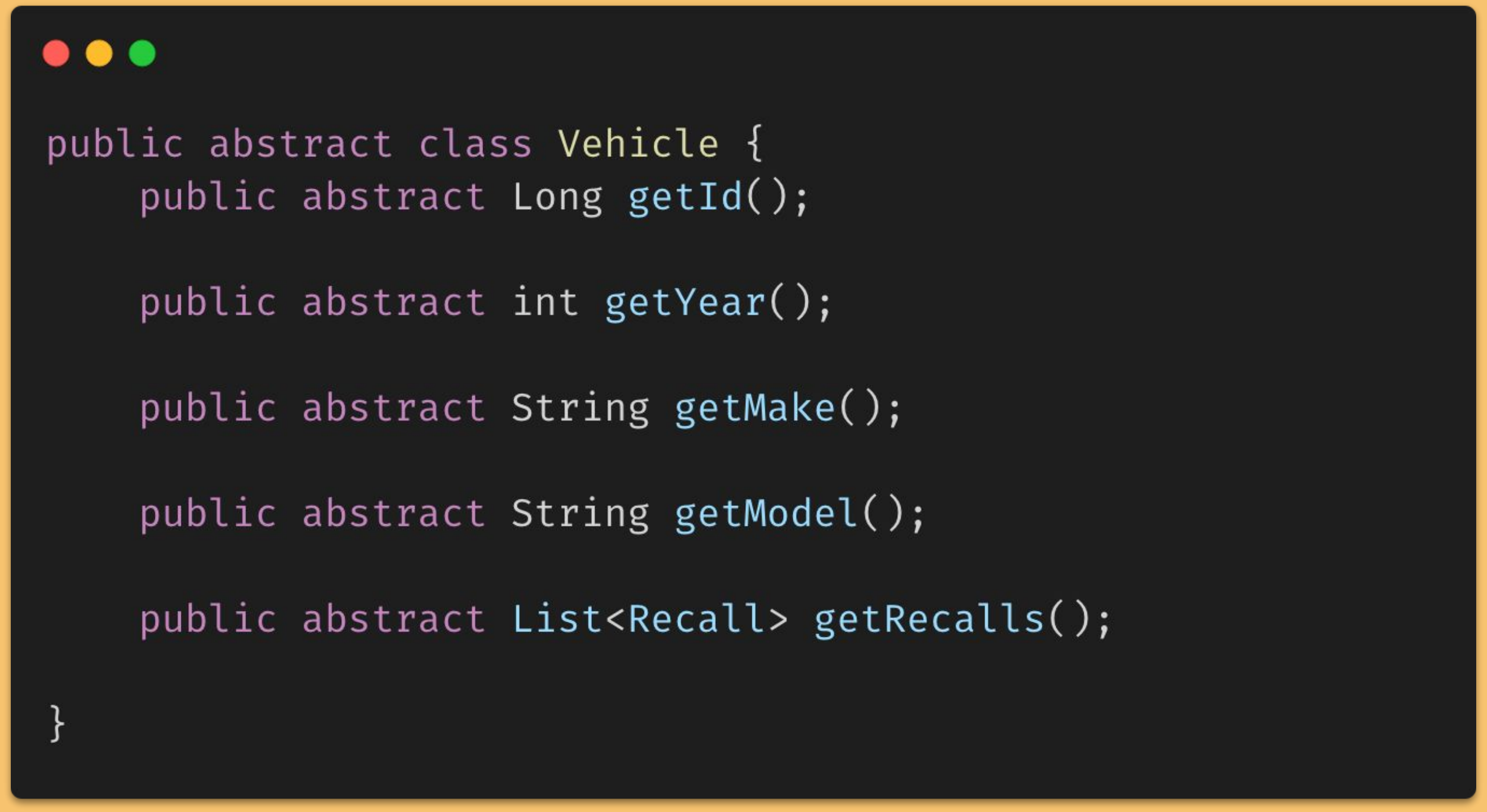
lincoln.com/api

```
service.class=lincoln
```

mercury.com/api

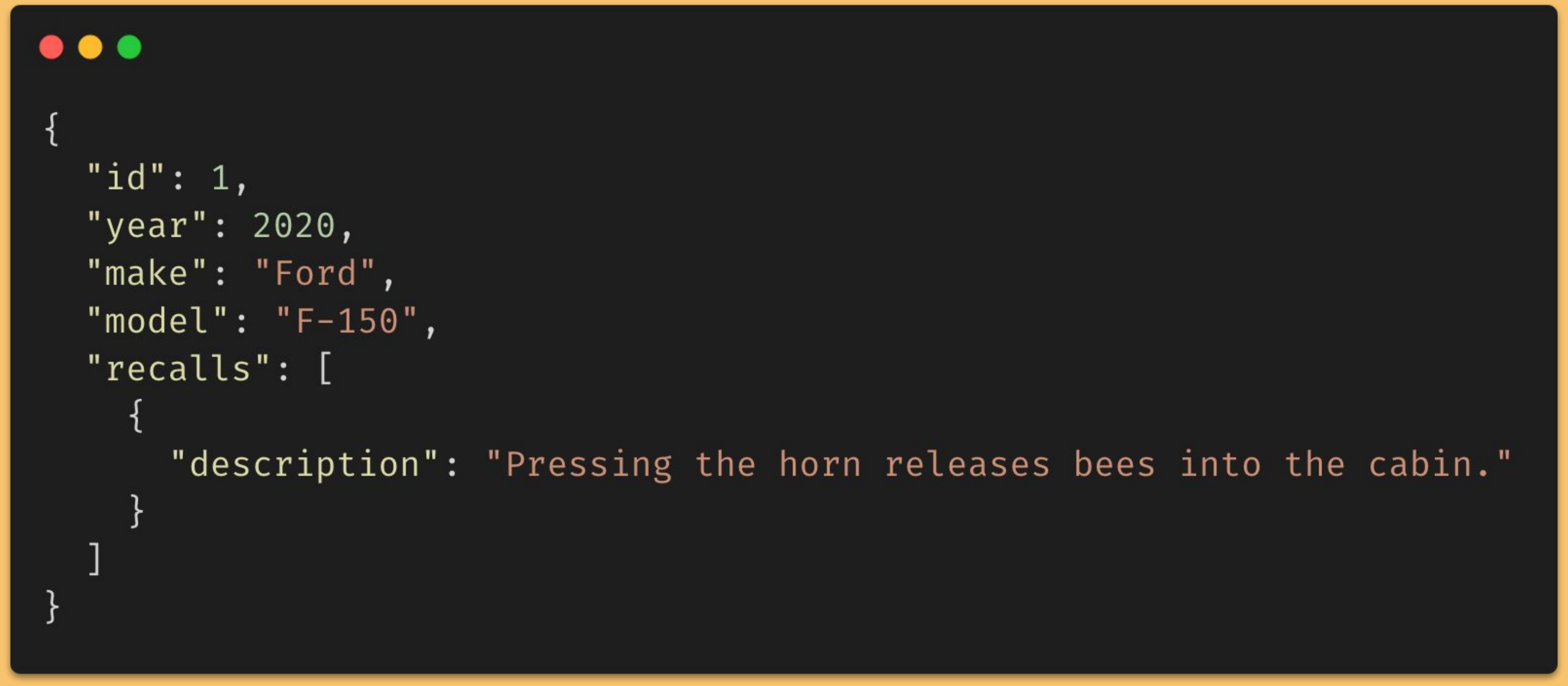
```
service.class=mercury
```

Interfaces build on each other



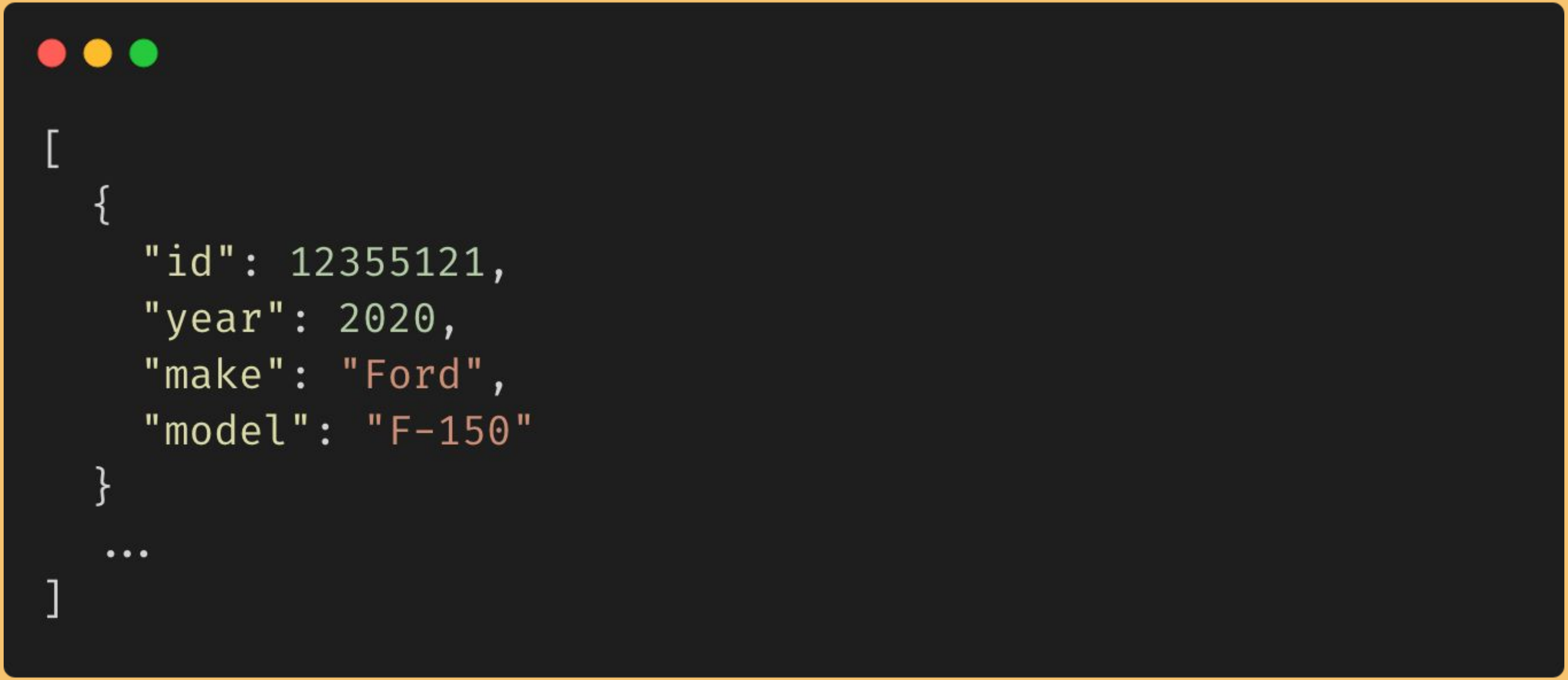
```java
public abstract class Vehicle {
    public abstract Long getId();

    public abstract int getYear();

    public abstract String getMake();

    public abstract String getModel();

    public abstract List<Recall> getRecalls();

}
```

ford.com/api/2020/F-150



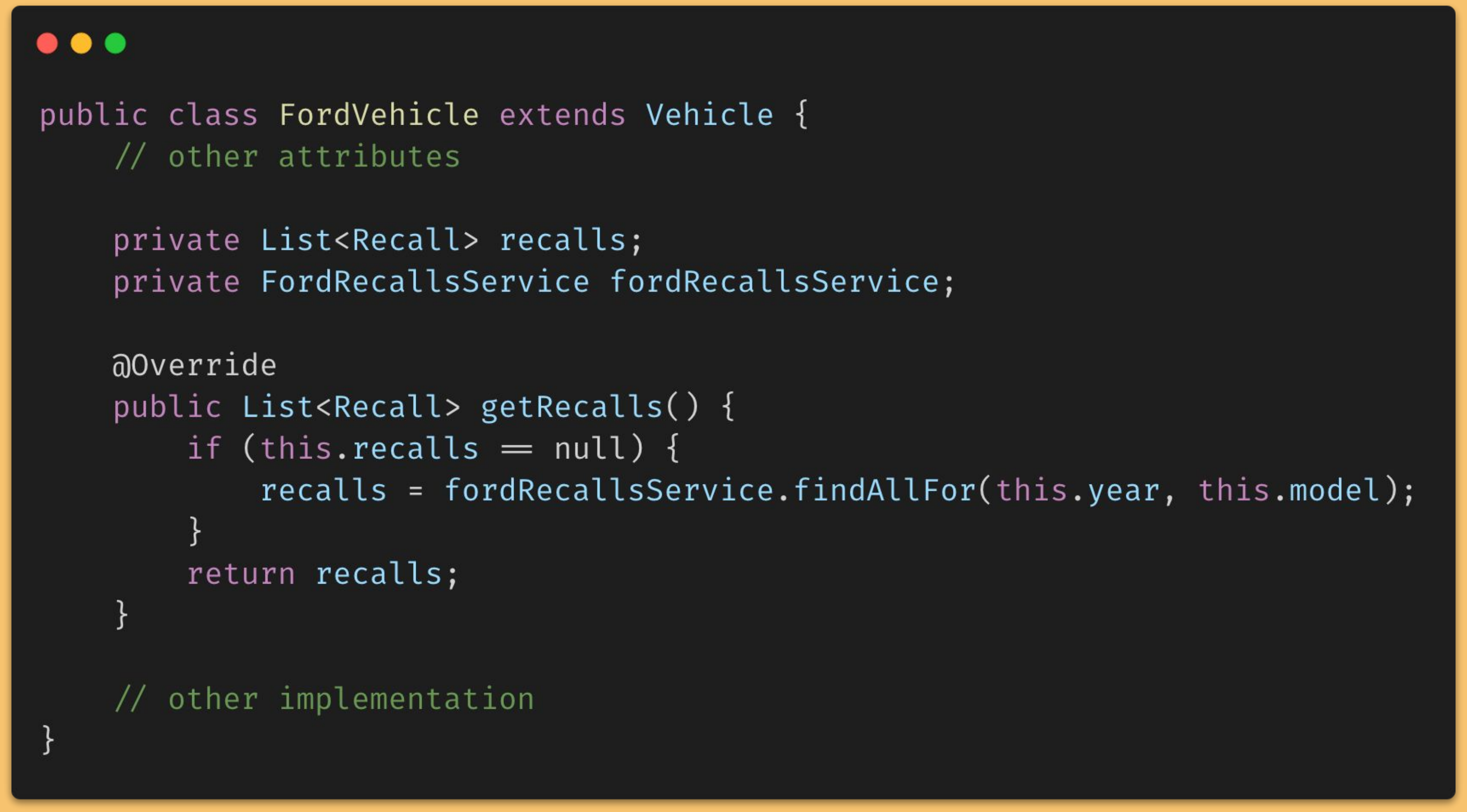
```
{
  "id": 1,
  "year": 2020,
  "make": "Ford",
  "model": "F-150",
  "recalls": [
    {
      "description": "Pressing the horn releases bees into the cabin."
    }
  ]
}
```

ford.com/api



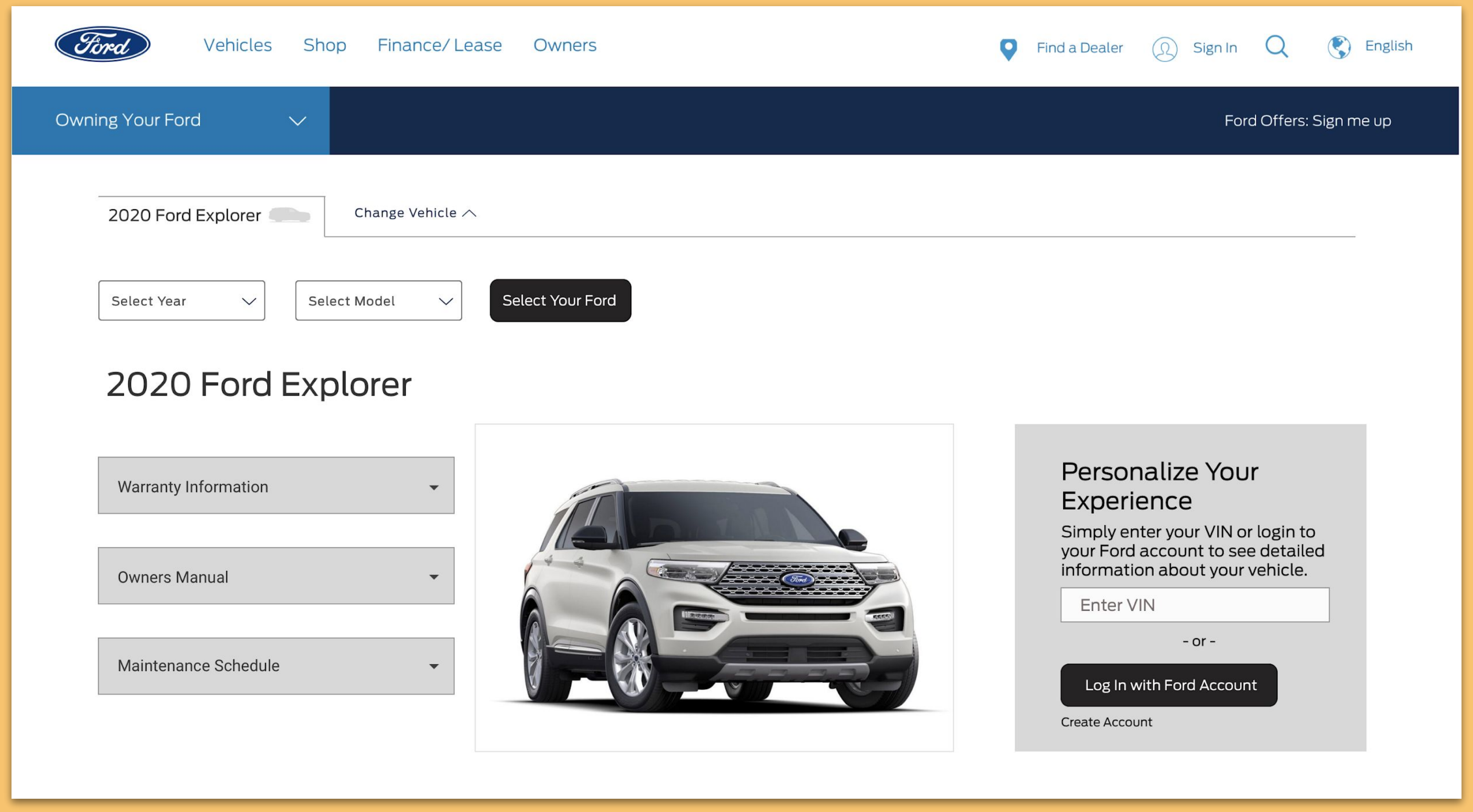
```
[
  {
    "id": 12355121,
    "year": 2020,
    "make": "Ford",
    "model": "F-150"
  }
   ...
]
```

*the recalls take a really long time to load



```java
public class FordVehicle extends Vehicle {
    // other attributes

    private List<Recall> recalls;
    private FordRecallsService fordRecallsService;

    @Override
    public List<Recall> getRecalls() {
        if (this.recalls == null) {
            recalls = fordRecallsService.findAllFor(this.year, this.model);
        }
        return recalls;
    }

    // other implementation
}
```

Programming to an interface instead of an implementation makes more flexible software



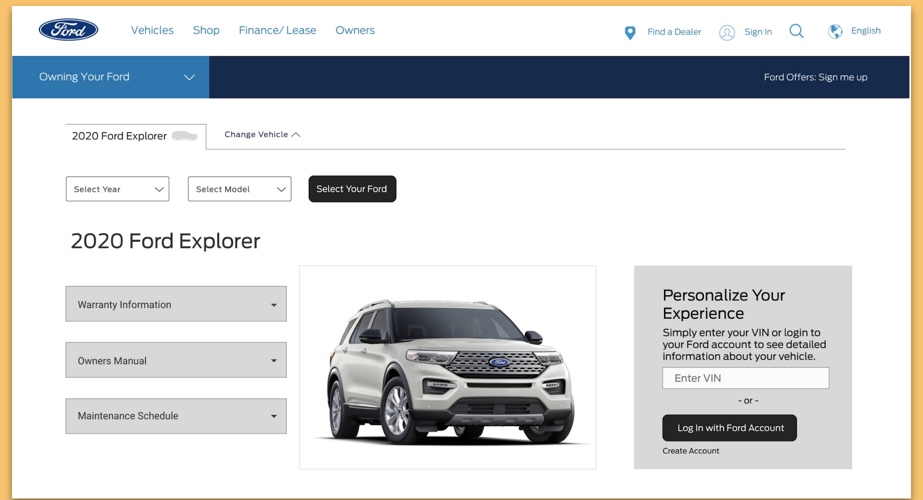
2020 Ford Explorer

Personalize Your Experience

Simply enter your VIN or login to your Ford account to see detailed information about your vehicle.

Enter VIN

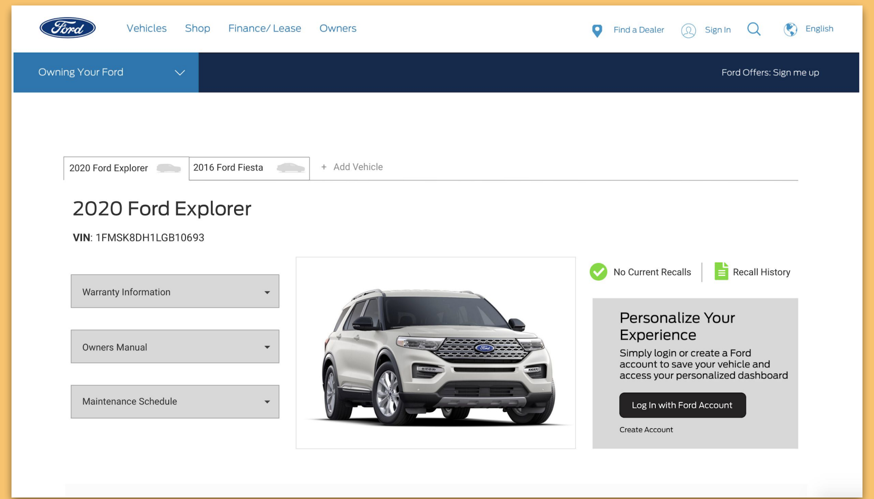
- or -



2020 Ford Explorer

Personalize Your Experience

Simply enter your VIN or login to your Ford account to see detailed information about your vehicle.

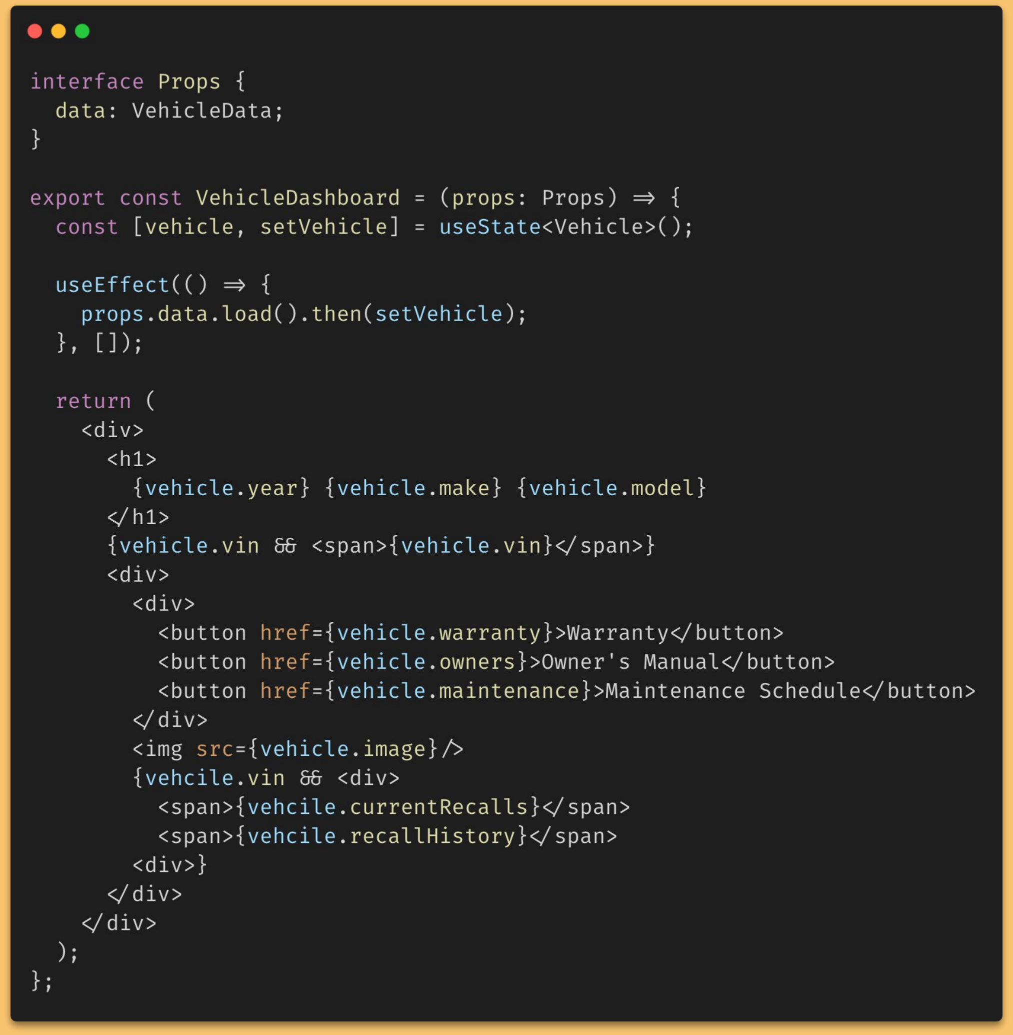


2020 Ford Explorer

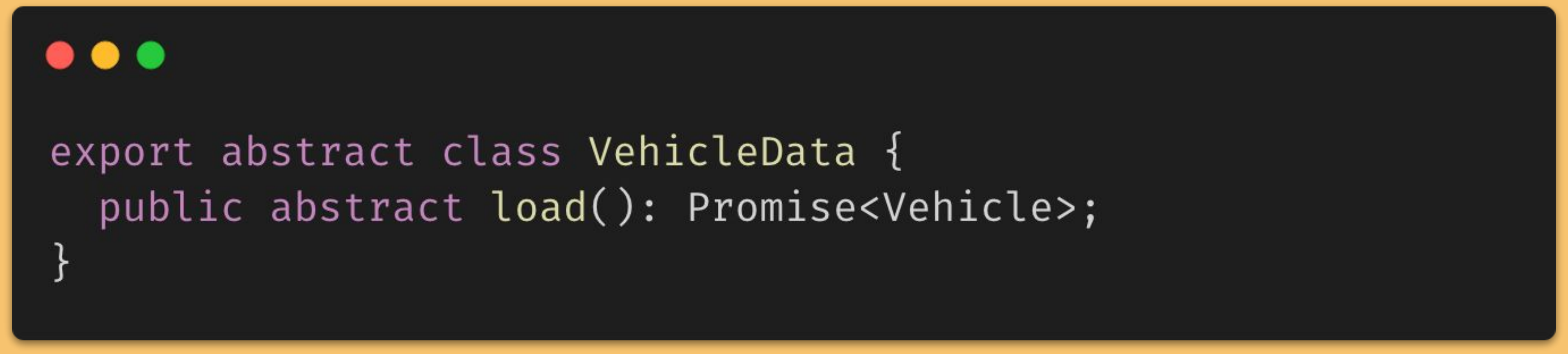
VIN: 1FMSK8DH1LGB10693

Personalize Your Experience

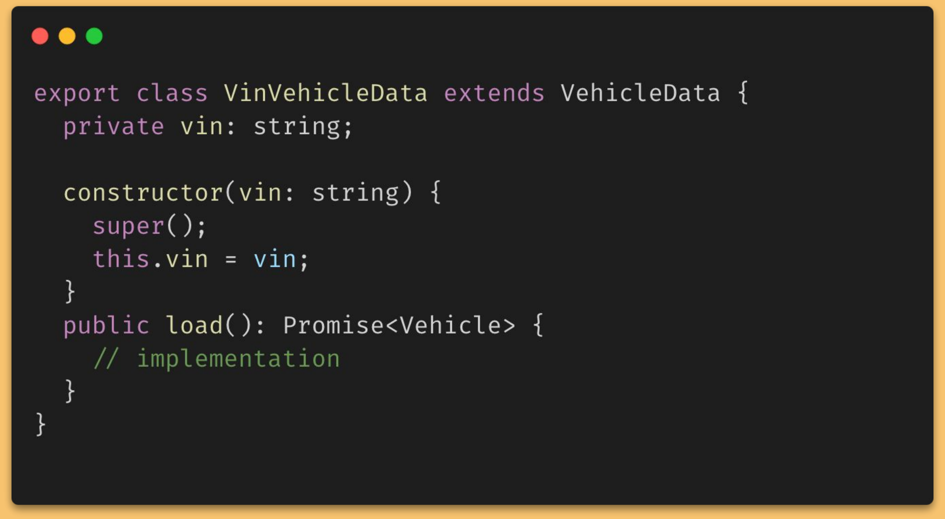
Simply login or create a Ford account to save your vehicle and access your personalized dashboard



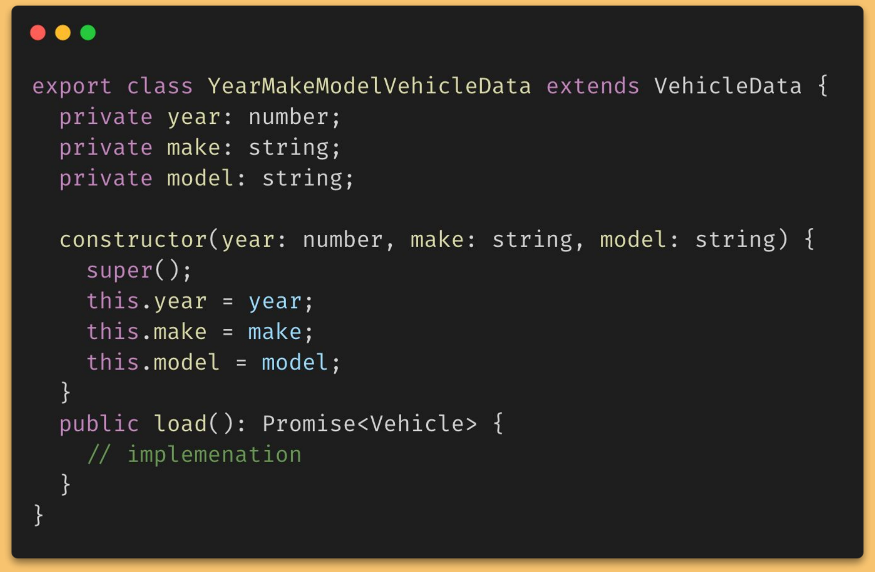
```tsx
interface Props {
  data: VehicleData;
}

export const VehicleDashboard = (props: Props) => {
  const [vehicle, setVehicle] = useState<Vehicle>();

  useEffect(() => {
    props.data.load().then(setVehicle);
  }, []);

  return (
    <div>
      <h1>
        {vehicle.year} {vehicle.make} {vehicle.model}
      </h1>
      {vehicle.vin && <span>{vehicle.vin}</span>}
      <div>
        <div>
          <button href={vehicle.warranty}>Warranty</button>
          <button href={vehicle.owners}>Owner's Manual</button>
          <button href={vehicle.maintenance}>Maintenance Schedule</button>
        </div>
        <img src={vehicle.image}/>
        {vehcile.vin && <div>
          <span>{vehcile.currentRecalls}</span>
          <span>{vehcile.recallHistory}</span>
        <div>}
      </div>
    </div>
  );
};
```



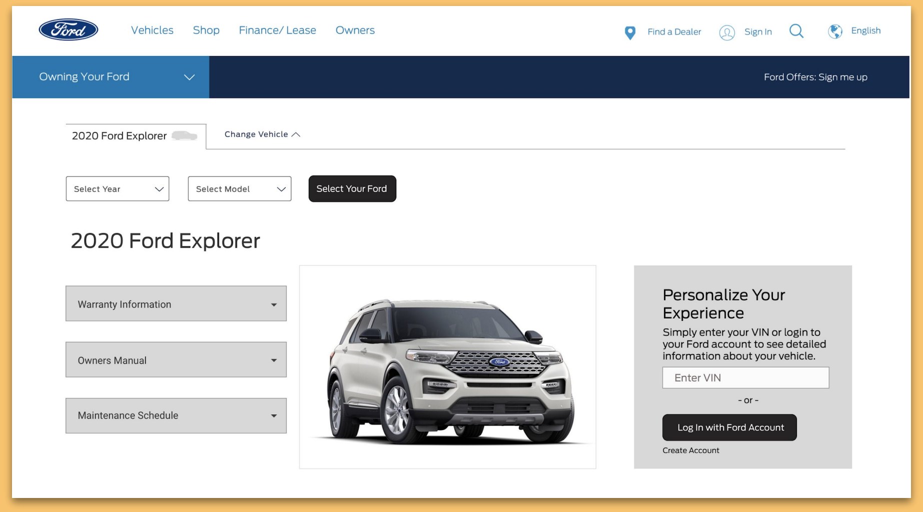
```
export abstract class VehicleData {
  public abstract load(): Promise<Vehicle>;
}
```



```
export class VinVehicleData extends VehicleData {
  private vin: string;

  constructor(vin: string) {
    super();
    this.vin = vin;
  }
  public load(): Promise<Vehicle> {
    // implementation
  }
}
```



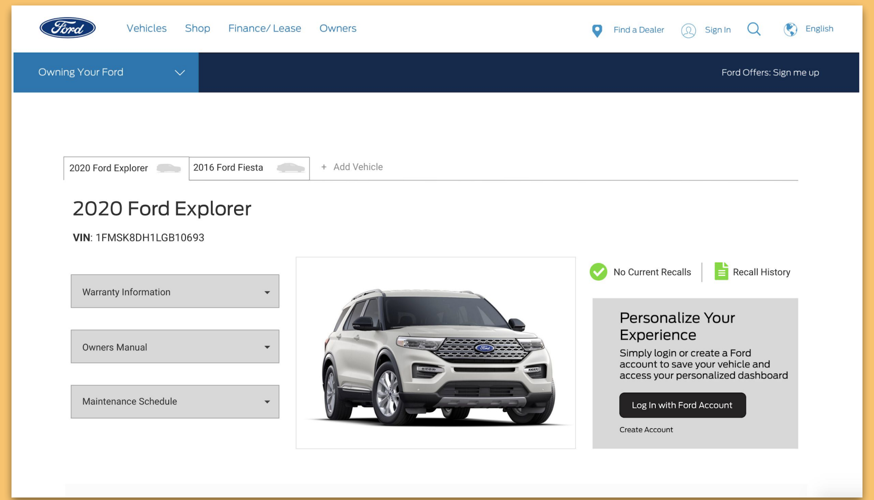
```
export class YearMakeModelVehicleData extends VehicleData {
  private year: number;
  private make: string;
  private model: string;

  constructor(year: number, make: string, model: string) {
    super();
    this.year = year;
    this.make = make;
    this.model = model;
  }
  public load(): Promise<Vehicle> {
    // implemenation
  }
}
```



2020 Ford Explorer

Personalize Your Experience

Simply enter your VIN or login to your Ford account to see detailed information about your vehicle.



2020 Ford Explorer

VIN: 1FMSK8DH1LGB10693

Personalize Your Experience

Simply login or create a Ford account to save your vehicle and access your personalized dashboard

Questions?

Thank you for attending. Goodbye.