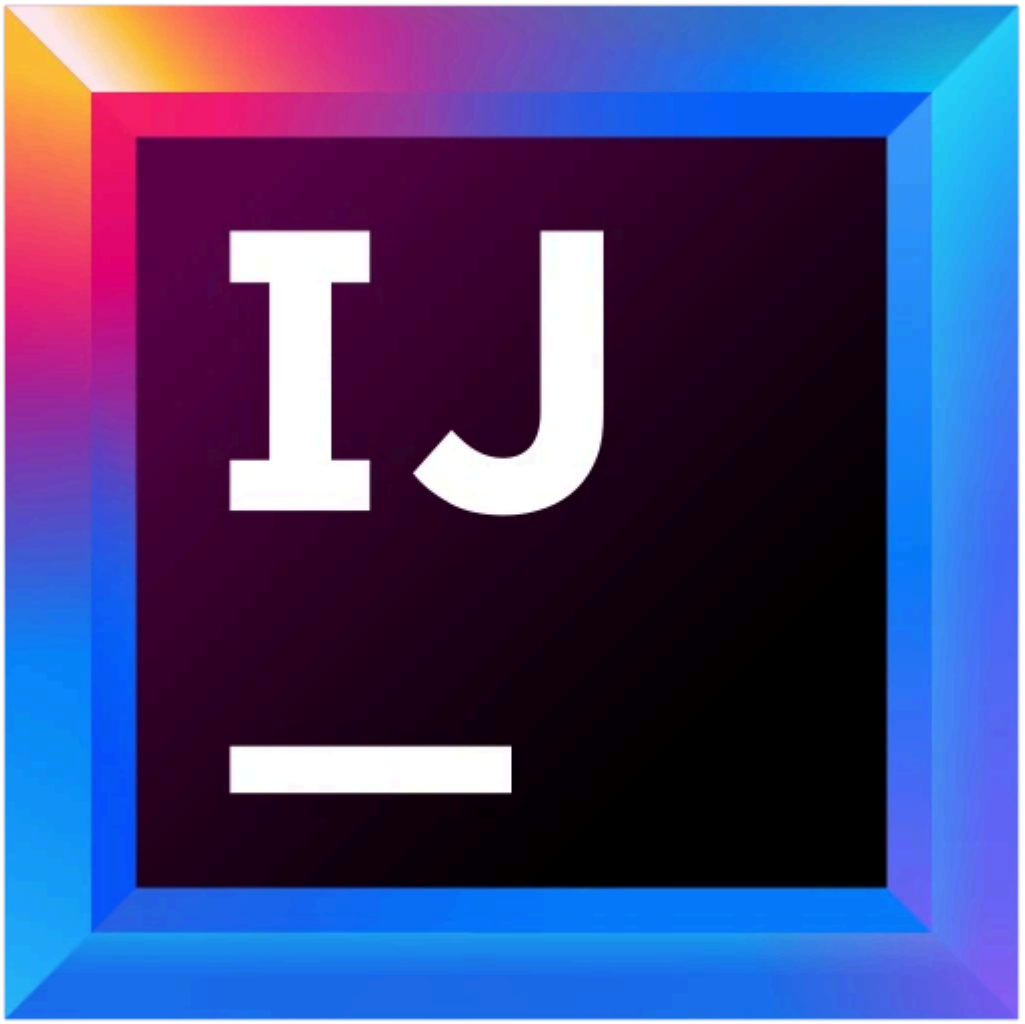
In this lab, you'll create and use classes to instantiate (create) objects in Java. A class provides a template that can be used to build real-world objects. You can use this template as often as you want to build multiple objects of the same kind.

You'll build a simple program simulating a car using concepts covered in previous lessons, like Strings, loops, and conditions.

🖥️ Note: When encountering this icon, it's time to get into your IDE and start coding!

In your lab environment, open IntelliJ by double-clicking on the icon.



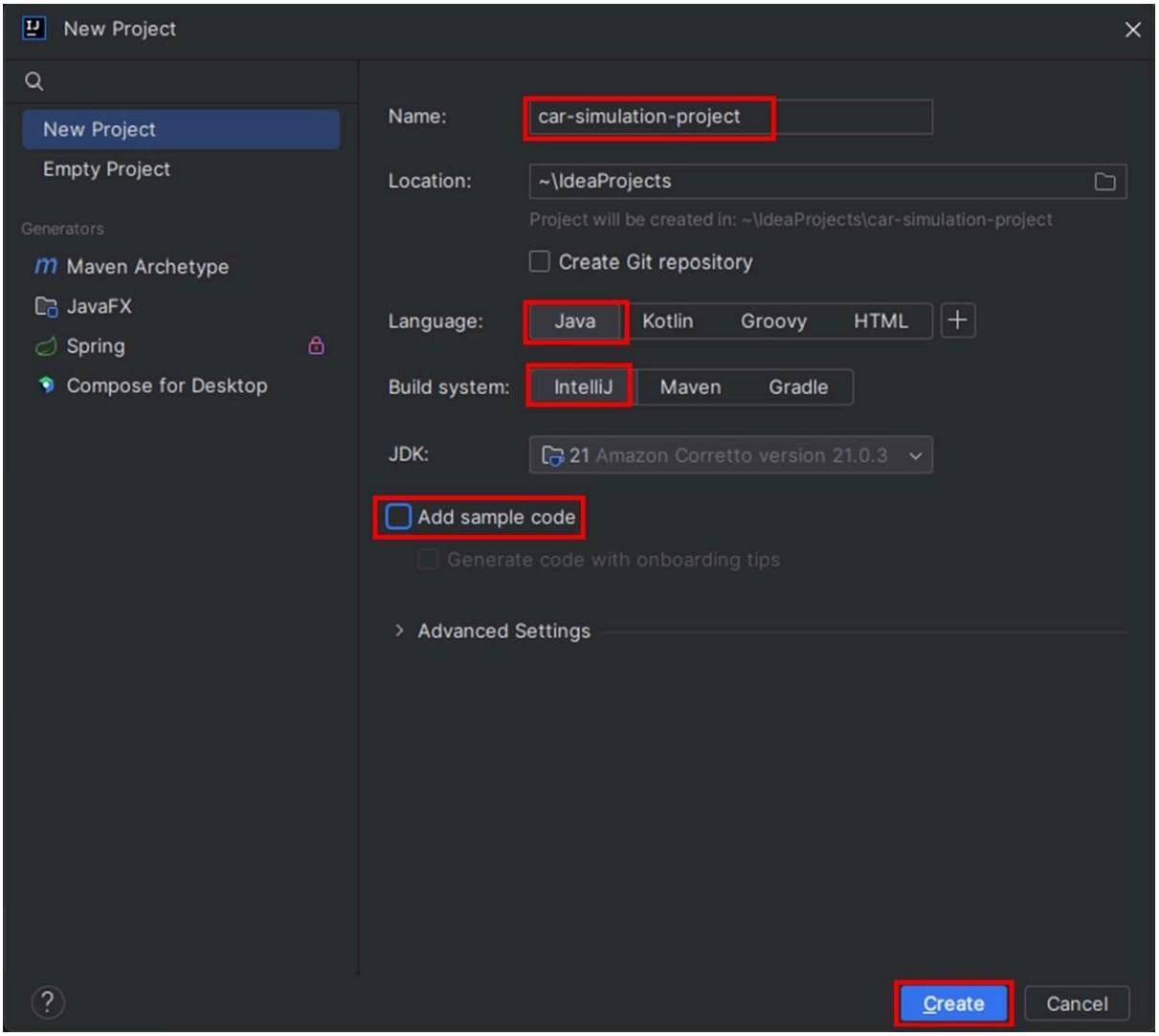
Create a class named `Car` to represent the properties of a car.

🖥️ **It's time to get coding!**

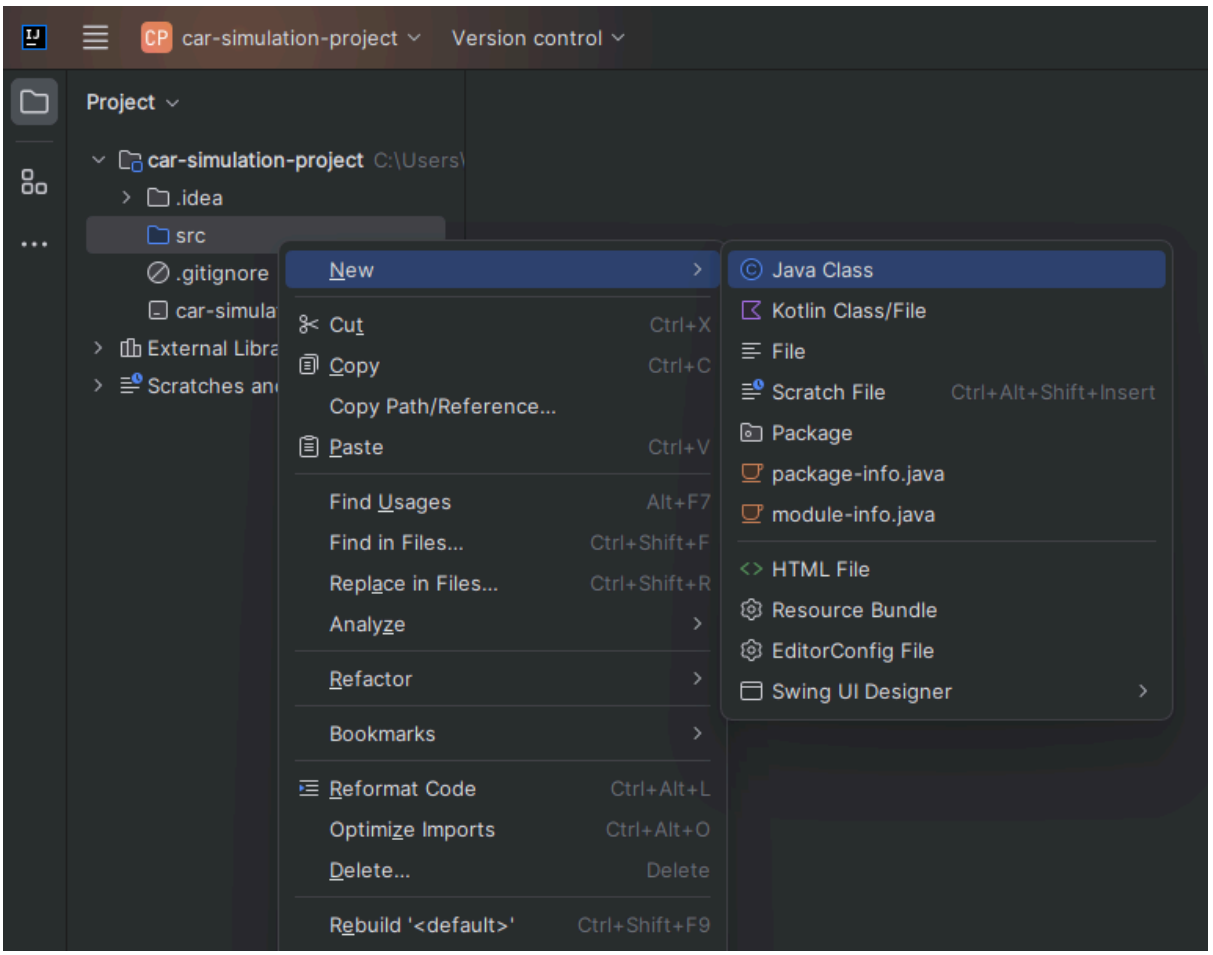
Step 1: Create a class named Car

Use IntelliJ to create the project and the `Car` class.

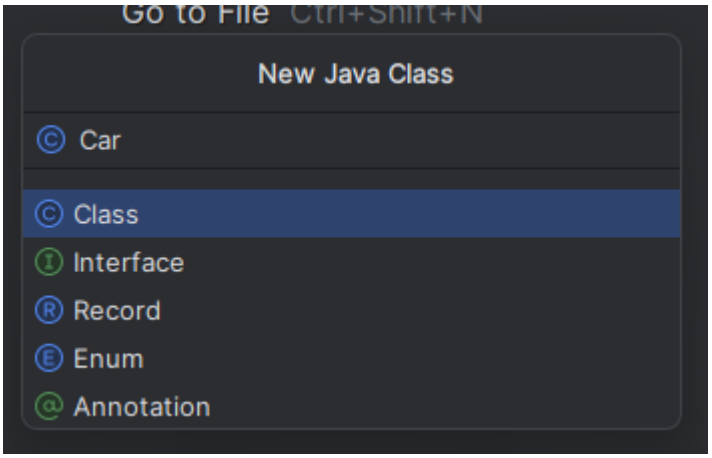
- 🖥️ **TODO 1: Open IntelliJ and create a new Java Project.**
  - **Name your project car-simulation-project.**
  - **Select the Language as Java, and Build system as IntelliJ.**
  - **Untick Add sample code, and click Create.**



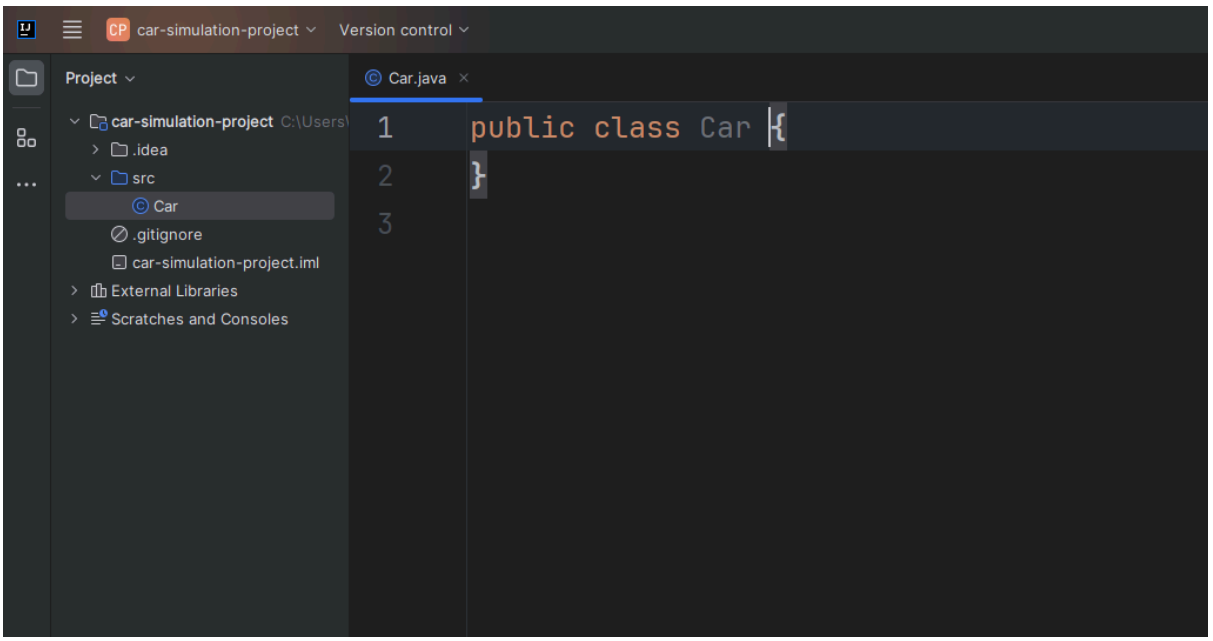
- 🖥️ **TODO 2: When the project is created, right-click on the *src* folder, select New, and select Java Class.**



- 🖥️ **TODO 3: In the pop-up window, notice that the Class option is preselected, you must only enter the name of your class, that is, `Car`.**



- 🖥️ **TODO 4: Press the enter key, and a file named *Car.java* is created containing the class.**



Step 2: Add properties to the Car class

The `Car` class will store information such as the make, model, year, and color of the car.

- 🖥️ **TODO 5: Create variables with appropriate data types.**

```
public class Car {

    String make; // stores the car's make (like Ford, Toyota)

    String model; // stores the car's specific model name (like Mustang, Camry)

    String color; // stores the car's color (like Red, Silver)

    int year; // stores the car's manufacturing year (like 2020)

}
```

Step 3: Add a simple method to the Car class

Create a method that returns the car details with each detail contained on a new line.

- 🖥️ **TODO 6: Create a method named `getCarDetails` inside the `Car` class. Make the return type of the method as String.**

```
public String getCarDetails() {

        // build, and return a string by adding each car property with a label
```

```
        // and using new line character \n

        return "Make: " + make + "\nModel: " + model + "\nColor: " + color +
"\nYear: " + year;

    }
```

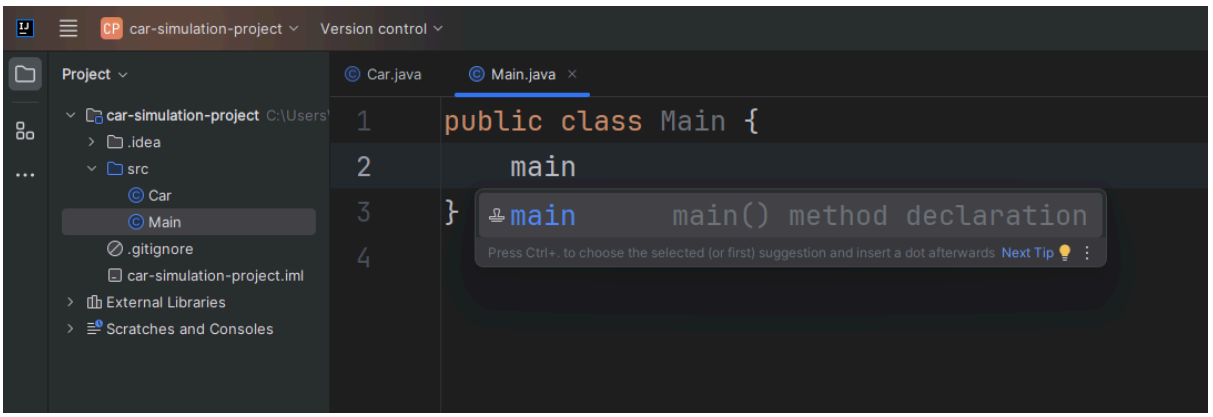
Create a `Car` object inside your main program to represent a single car. Then provide values to all the properties of your car object, and display the car details.

Remember that characteristics are technically called properties of an object.

🖥️ **It's time to get coding!**

Step 1: Create the Main class and main method

- 🖥️ **TODO 7: Right click on the *src* folder, select New, and click on Java Class.**
  - **Name the class `Main` and press enter.**
  - **Inside the `Main` class, type main, and IntelliJ will suggest completing the method signature.**
  - **Press Enter.**



Step 2: Create a Car object

- 🖥️ **TODO 8: Inside the `main` method, declare a `Car` reference variable, named `myFirstCar`, and initialize it with an object using the new keyword.**

```
public class Main {

    public static void main(String[] args) {

        Car myFirstCar = new Car();
```

```
    }
```

Step 3: Provide values to all the Car properties

- 🖥️ **TODO 9: Assign values to the variables of the `Car` object using the `dot` operator.**

```
public class Main {

    public static void main(String[] args) {

        Car myFirstCar = new Car();



        myFirstCar.make = "Toyota";

        myFirstCar.model = "Corolla";

        myFirstCar.color = "Black";

        myFirstCar.year = 2019;

    }

}
```

Step 4: Display the car details

- 🖥️ **TODO 10: Call the `getCarDetails` method with `myFirstCar` using the `dot` operator.**

```
public class Main {

    public static void main(String[] args) {

        Car myFirstCar = new Car();



        myFirstCar.make = "Toyota";

        myFirstCar.model = "Corolla";

        myFirstCar.color = "Black";

        myFirstCar.year = 2019;
```

```
        // display the car details for myFirstCar

        myFirstCar.getCarDetails();

    }

}
```

In your main program, create an array of `Car` objects to represent your car collection. You will then loop through this array, create individual `car` objects, and ask the user to input the values for their properties.

🖥️ **It's time to get coding!**

Step 1: Create an array of Car Objects

- 🖥️ **TODO 11: Inside the `main` method, declare an array of `car` objects, and name it `cars`. You can fix the size of the array as 5.**

```
public static void main(String[] args) {

        Car myFirstCar = new Car();

        myFirstCar.make = "Toyota";

        myFirstCar.model = "Corolla";

        myFirstCar.color = "Black";

        myFirstCar.year = 2019;



        // display the car details for myFirstCar

        myFirstCar.getCarDetails();



        // array of Car objects

        Car[] cars = new Car[5];

    }
```

Step 2: Create an object of Scanner class

Remember, if you want to get user input you're going to need a Scanner, and if you need a Scanner, you will have to import it from the *java.util* package!

- 🖥️ **TODO 12: Import `Scanner`. Keep in mind that the `import` statement must be written at the top of the program, before class `Main`.**
- 🖥️ **TODO 13: Once you've imported the `Scanner`, you must declare it.**

```
public class Main {

    public static void main(String[] args) {

        Car myFirstCar = new Car();

        myFirstCar.make = "Toyota";

        myFirstCar.model = "Corolla";

        myFirstCar.color = "Black";

        myFirstCar.year = 2019;



        // display the car details for myFirstCar

        myFirstCar.getCarDetails();



        // array of Car objects

        Car[] cars = new Car[5];



        // import Scanner class as java.util.Scanner

        // create an object of Scanner class

        Scanner scanner = new Scanner(System.in);

    }

}
```

Step 3: Create individual Car objects, and prompt the user for values

- 🖥️ **TODO 14: Write a for loop inside the `main` method to loop through the `cars` array.**

```
// array of Car objects

Car[] cars = new Car[5];



// object of Scanner class

Scanner scanner = new Scanner(System.in);



// loop through the array of cars

for (int index = 0; index < 5; index++) {



}
```

- 🖥️ **TODO 15: Inside the for loop, create a new `car` object using the index position of the loop.**
  - **Prompt the user for the `make`, `model`, `color`, and `year` of the `car`.**
  - **Assign these values directly to the variables of the `Car` object.**

Tip

When you use `nextInt()` to read the year, Java might get a little tricky! After grabbing an integer, Java leaves behind an extra "invisible" newline, which can mess things up when you try to read the next line of text.

To fix this, you add a simple line of code right after reading the year: `keyBoard.nextLine();`. This way, everything will run smoothly and Make and Model inputs won't get combined.

```
// array of Car objects

        Car[] cars = new Car[5];
```

```
        // object of Scanner class
        Scanner scanner = new Scanner(System.in);


        // loop through the array of cars
        for (int index = 0; index < 5; index++) {
            cars[index] = new Car(); // create a new car object at current index


            // prompt the user for details
            System.out.println("Enter details for car " + (index + 1));


            // get user input and assign to 'make' of the car object at current
index
            System.out.print("Make: ");
            cars[index].make = scanner.nextLine();


            // repeat for model, color, and year


            // consume the newline character after nextInt()
            keyBoard.nextLine();
```

In your `main` method, after creating the `Car` objects with values input by the user, you can call the `getCarDetails` method to print the details of all `car` objects. Remember that the `getCarDetails` method returns a String value.

🖥️ **It's time to get coding!**

For this task, you will again use a for loop.

- 🖥️ **TODO 16: Inside the main method, write another for loop to iterate through the `cars` array.**
    - **Call the `getCarDetails` method with the `car` object at the current index position.**
    - **Store the return value from the method in a variable called `carDetails`.**
    - **Print the `carDetails`.**

```
// call getCarDetails for each car object in the collection

        System.out.println("Your Car Collection");

        for (int index = 0; index < 5; index++) {

            String carDetails = cars[index].getCarDetails();

            System.out.println(carDetails);

        }
```

Before concluding, run your program to test its functionality. Input car details when prompted and observe the output. If you notice any issues (for example, the Make and Model input appearing combined), ensure you've handled input properly with the `Scanner` class. Reflect on the output, and make adjustments to fix any errors.

By testing and debugging, you'll enhance your understanding of object-oriented programming in Java and be better prepared for more complex applications. Good work!

In this lab, you were introduced to the fundamental concepts of classes and objects in Java. Classes provide a blueprint to create objects that contain data (properties) and functionalities (methods). Objects interact with each other to simulate real-world entities and processes.

You created a simple program to help Maya simulate a single car, as well as a collection of cars. You focused on defining a `Car` class to represent a car's properties and created objects to represent individual cars. You also modified the car's data and implemented a basic method to display the car object's information.

By understanding classes and objects, you've laid the groundwork for building more complex programs in Java.