

Fields

This lesson will go into the details of the fields of a class.

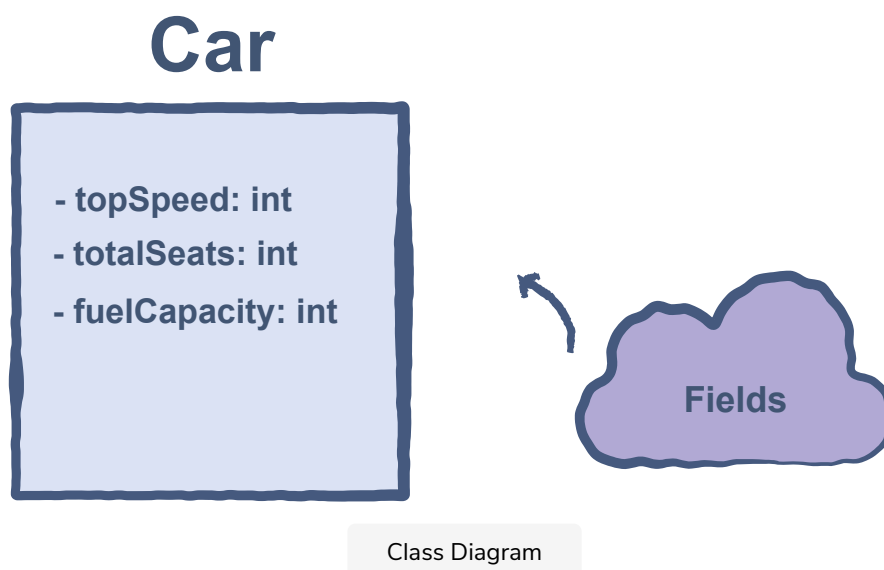
We'll cover the following ^

- Java Fields
- Static and Non-static Fields
 - Static Field
 - Non-Static Field
- Final Fields

Java Fields

Java fields are actually the *data members* inside a class. For instance, in a class representing Car, the Car class might contain the following fields:

- *topSpeed*
- *totalSeats*
- *fuelCapacity*



The Java class could be defined like this:

```
public class Car {  
  
    int topSpeed;  
    int totalSeats;  
    int fuelCapacity;  
  
}
```

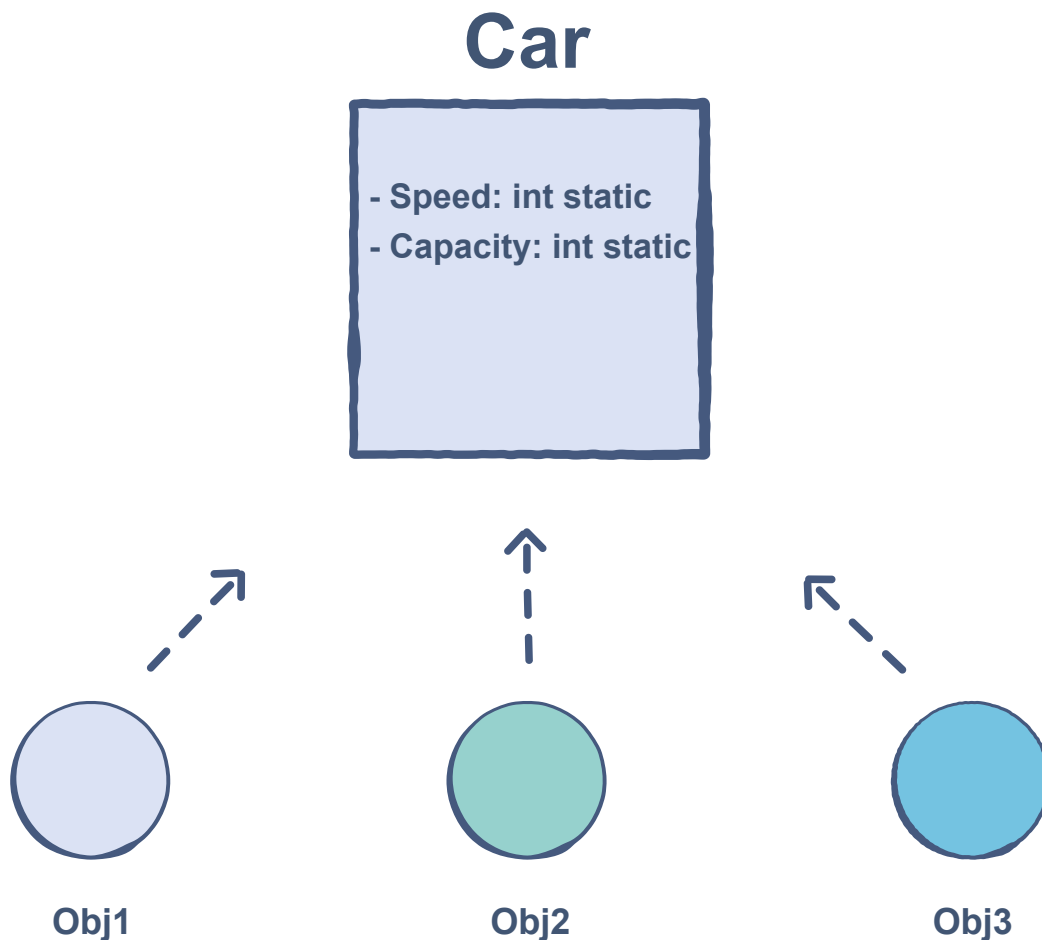


Static and Non-static Fields

Java supports static and non-static fields.

Static Field

A static field resides in a class. All the objects we create will share this field and its value.



You can define a static field by using the `static` keyword in Java:

```
class Car {  
  
    // static fields  
    static int speed;  
    static int capacity;  
  
}
```



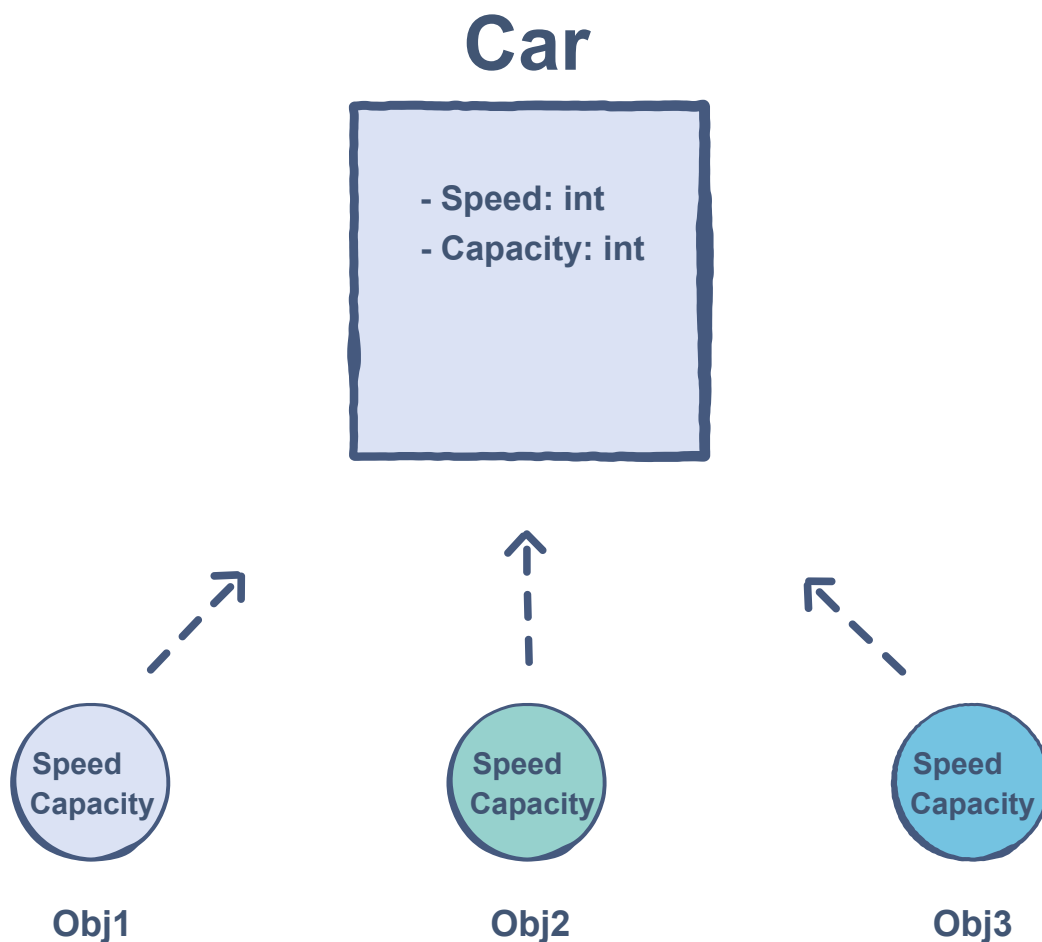
Static fields reside in the class. We don't need an instance of the class to access static fields. We can access the static fields of a class by just writing the class name before the field:

```
// Static fields are accessible in the main
System.out.println(Car.speed);
System.out.println(Car.capacity);
```



Non-Static Field

Non-static fields are located in the instances of the class. Each instance of the class can have its own values for these fields.



You can define a non-static field like this in Java:

```
class Car {

    // Non-Static Fields
    int speed;
    int capacity;

}
```



As non-static fields doesn't reside in the class, So we need an instance of the class to access non-static fields.

```
Car obj1 = new Car();

System.out.println(obj1.speed);
System.out.println(obj1.capacity);
```



Final Fields

A final field cannot have its value changed once it is assigned. We can make a field final by using the keyword **final**.

Here is an example in Java:

```
class Car {
    // Final field of capacity = 4
    // Now Capacity can never be changed from 4
    // to some other value through the program
    final int capacity = 4;
}
```



Car class has the capacity equals to 4 which can't be changed. If you try to do so, you will get a compilation error:

can't assign a value to final variable capacity.

You can check it on your own in the following code widget:

```
class Car {

    // Final variable capacity
    final int capacity = 4;

}

class Demo {

    public static void main() {

        Car car = new Car();
        car.capacity = 5; // Trying to change the capacity value
    }

}
```





In the next lesson, we'll discuss methods in Java.