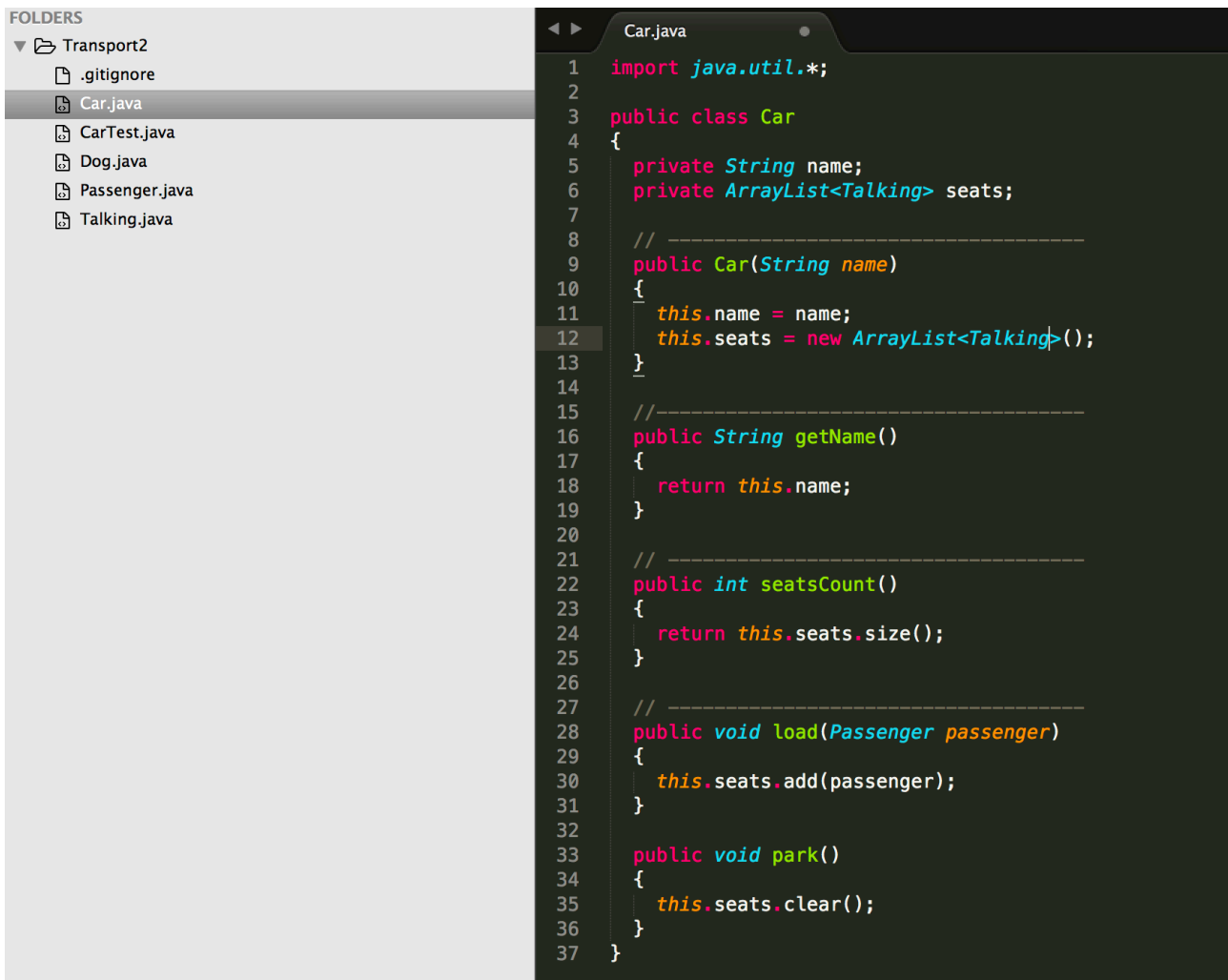


I.T.f

Polymorphism Example

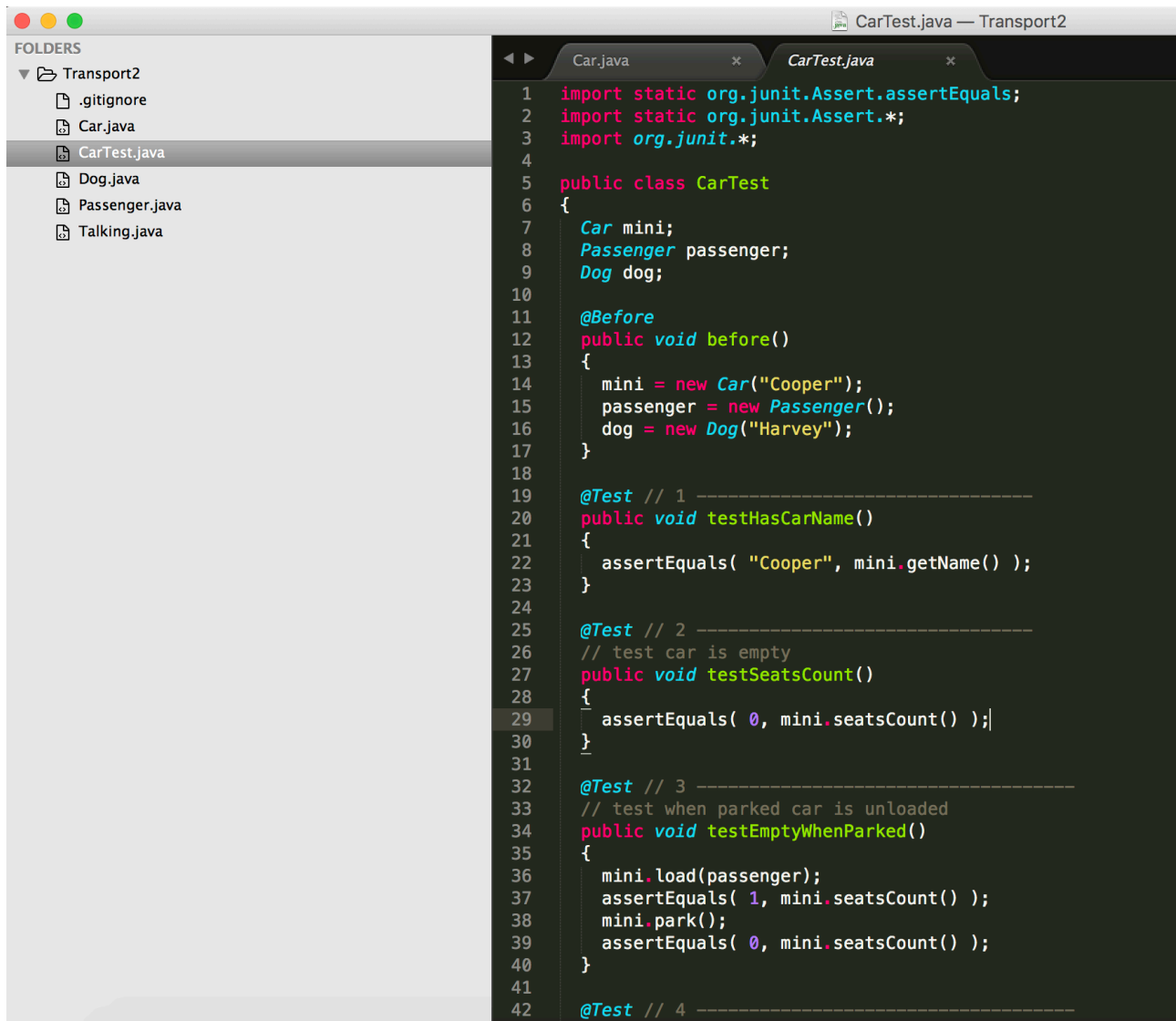
- Carry out a Polymorphism task(T .1) or demonstrate use of Polymorphism in a program you have written.

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object



The screenshot shows an IDE with a project named 'Transport2'. The 'FOLDERS' pane on the left lists the following files: .gitignore, Car.java, CarTest.java, Dog.java, Passenger.java, and Talking.java. The 'Car.java' file is selected and its code is displayed in the main editor. The code defines a 'Car' class with a 'name' attribute, a 'seats' attribute (an ArrayList of Talking objects), and methods for getting the name, getting the seat count, loading a passenger, and parking.

```
1  import java.util.*;
2
3  public class Car
4  {
5      private String name;
6      private ArrayList<Talking> seats;
7
8      // -----
9      public Car(String name)
10     {
11         this.name = name;
12         this.seats = new ArrayList<Talking>();
13     }
14
15     // -----
16     public String getName()
17     {
18         return this.name;
19     }
20
21     // -----
22     public int seatsCount()
23     {
24         return this.seats.size();
25     }
26
27     // -----
28     public void load(Passenger passenger)
29     {
30         this.seats.add(passenger);
31     }
32
33     public void park()
34     {
35         this.seats.clear();
36     }
37 }
```



```
1 import static org.junit.Assert.assertEquals;
2 import static org.junit.Assert.*;
3 import org.junit.*;
4
5 public class CarTest
6 {
7     Car mini;
8     Passenger passenger;
9     Dog dog;
10
11     @Before
12     public void before()
13     {
14         mini = new Car("Cooper");
15         passenger = new Passenger();
16         dog = new Dog("Harvey");
17     }
18
19     @Test // 1 -----
20     public void testHasCarName()
21     {
22         assertEquals( "Cooper", mini.getName() );
23     }
24
25     @Test // 2 -----
26     // test car is empty
27     public void testSeatsCount()
28     {
29         assertEquals( 0, mini.seatsCount() );
30     }
31
32     @Test // 3 -----
33     // test when parked car is unloaded
34     public void testEmptyWhenParked()
35     {
36         mini.load(passenger);
37         assertEquals( 1, mini.seatsCount() );
38         mini.park();
39         assertEquals( 0, mini.seatsCount() );
40     }
41
42     @Test // 4 -----
```

