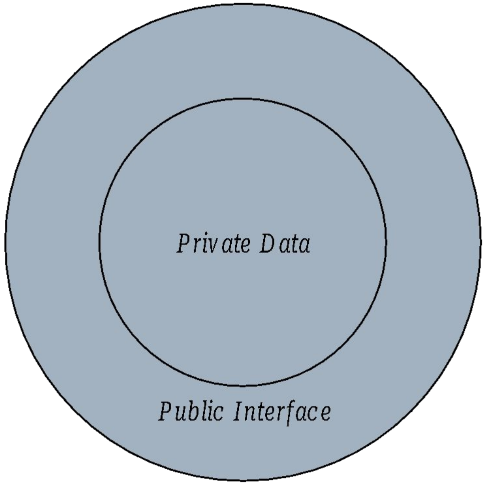
Properties of OOP

SE 206

Encapsulation

- The data (state) of an object is private – it cannot be accessed directly.
- The state can only be changed through its public *interface*.
- This is called *encapsulation*

"The Doughnut Diagram" Showing that an object has private state and public behaviour. State can only be changed by invoking some behaviour



Classes

Preparation

- Scene so far has been background material and experience
  - Variables
  - Data Types
  - Input and output
  - Expressions
  - Assignments
  - Objects
  - Standard classes and methods
  - Decisions (if, switch)
  - Loops (while, for, do-while)

- Now: Experience what Java is really about
  - Design and implement objects representing information and physical world objects

Object-oriented programming

- Basis
  - Create and manipulate objects with attributes and methods that the programmer can specify
- Mechanism
  - Classes
- Benefits
  - An information type is designed and implemented once
    - Reused as needed
      - No need reanalysis and re-justification of the representation

Known Classes

- Classes we’ve seen
  - String
  - Scanner
  - System

The Car class

A new example: creating a Car class

- □ What properties does a car have in the real world?
  - Color
  - Position (x,y)
  - Fuel in tank

- □ We will implement these properties in our Car class

```
ðťÒ ╒ ╒8Ò┤ Ŧ
┘ ┤ ŀ▬Ò ╚¼8δťδ┤ ðδťδ┤ ẁ
┘ ┤ ŀ▬Ò ╚¼ŀŻ ╚◘┘ δ ╒ẁ
┘ ┤ ŀ▬Ò ╚¼ŀŻ ╚◙┘ δ ╒ẁ
┘ ┤ ŀ▬Ò ╚¼ŀŻ ╚ď ╟¼ťẁ

ỳỳǿǿǿ
ŧ
```

Car’s instance variables

ðtʼÒ ╒ ╒8Ò┤ Ŧ

┘ ┤ l·▬Ò╚¼8δtʼδ┤ ðδtʼδ┤ ẁ

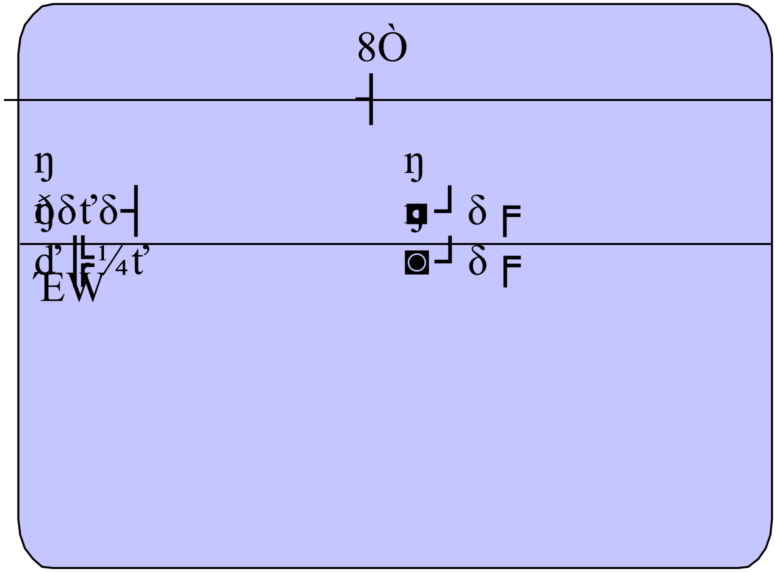
┘ ┤ l·▬Ò╚¼l·Ż ╚◘┘ δ ╒ẁ

┘ ┤ l·▬Ò╚¼l·Ż ╚◙┘ δ ╒ẁ

┘ ┤ l·▬Ò╚¼l·Ż ╚ďʼ╠¼tʼẁ

ỳỳǿǿǿ

ŧ



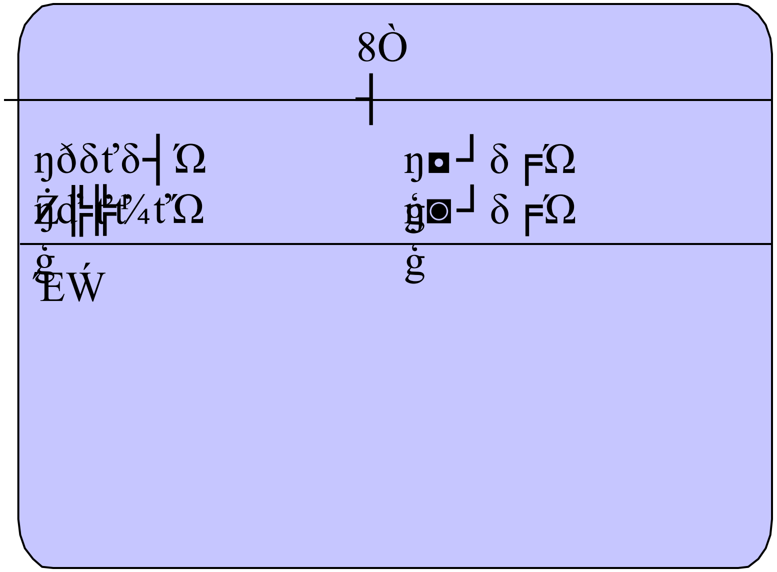
8Ò
┤

ŋ
ðδtʼδ┤
ďʼ╠¼tʼ
ΈW

ŋ
◘┘ δ ╒
◙┘ δ ╒

Instance variables and attributes

- □ Default initialization
  - ■ If the variable is within a method, Java does NOT initialize it
  - ■ If the variable is within a class, Java initializes it as follows:
    - □ Numeric instance variables initialized to 0
    - □ Logical instance variables initialized to false
    - □ Object instance variables initialized to null



Car behaviors or methods

- What can a car do? And what can you do to a car?
  - Move it
    - Change it's x and y positions
  - Change it's color
  - Fill it up with fuel

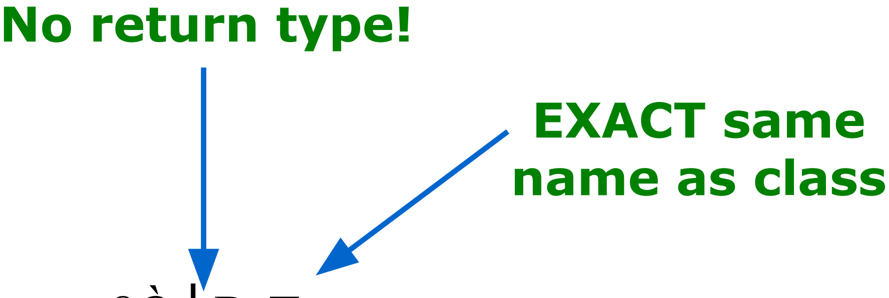
- For our computer simulation, what else do we want the Car class to do?
  - Create a new Car
  - Change Car's condition
- Each of these behaviors will be written as a method

Creating a new car

- □ To create a new Car, we call:
  - ■ Car car = new $\boxed{\text{Car();}}$
- □ Notice this looks like a method
  - ■ You are calling a special method called a constructor
  - ■ A constructor is used to create (or construct) an object
    - □ It sets the instance variables to initial values
- □ The constructor:

```
8Ò┤ ŖŗŦ
    ď╠¼ťΩĤģģģẁ
    ðδťδ┤ ΏΩ8δťδ┤ ǿ7s≤Cẁ
ŧ
```

Constructors



No return type!

EXACT same name as class

```
8Ò┤ ŖŗŦ
      ď’╠¼ťΏĤǵǵǵẁ
      ðδťδ┤ Ώ8δťδ┤ ǿ7s≤Cẁ
ŧ
```

Our Car class so far

```
ðťÒ ╒ ╒8Ò┤ Ŧ
┘┤ ŀ▬Ò╙¼8δťδ┤ ðδťδ┤ ẁ
┘┤ ŀ▬Ò╙¼ŀŻ ╙◘┘ δ ╒ẁ
┘┤ ŀ▬Ò╙¼ŀŻ ╙◙┘ δ ╒ẁ
┘┤ ŀ▬Ò╙¼ŀŻ ╙ď╠¼ťẁ

    8Ò┤ ŖŗŦ
        ď╠¼ťΏĤģģģẁ
        ðδťδ┤ Ώ8δťδ┤ ǿ7s≤Cẁ
    ŧ
ŧ
```

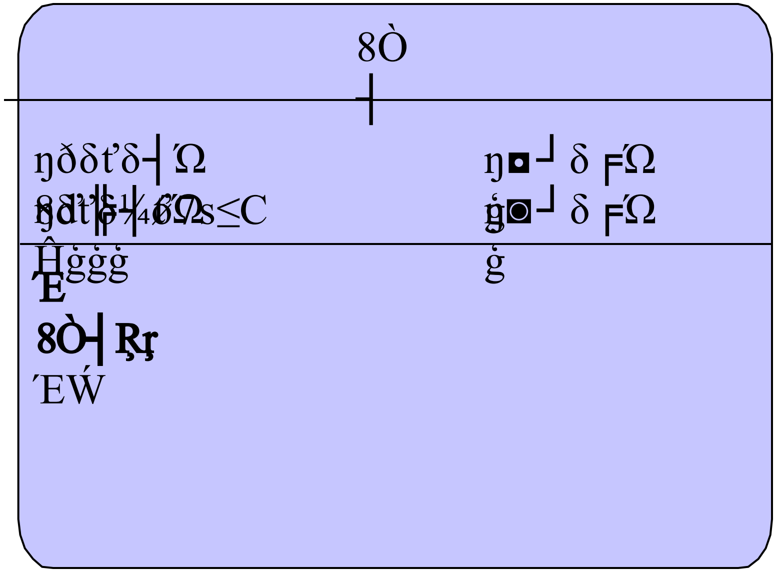
```
ðťÒ ╒ ╒8Ò┤ Ŧ
┘┤ ŀ▬Ò╙¼8δťδ┤ ðδťδ┤  Ώ
                    8δťδ┤ ǿ7s≤C  ẁ
┘┤ ŀ▬Ò╙¼ŀŻ ╙◘┘ δ ╒ẁ
┘┤ ŀ▬Ò╙¼ŀŻ ╙◙┘ δ ╒ẁ
┘┤ ŀ▬Ò╙¼ŀŻ ╙ď╠¼ťΏĤģģģẁ

    8Ò┤ ŖŗŦ
    ŧ
ŧ
```

Our Car class so far

```
ðtťÒ ╒ ╒8Ò┤ Ŧ
┘ ┤ ŀ▬Ò ╚¼8δtťδ┤ ðδtťδ┤ Ώ
                8δtťδ┤ ǿ7s≤Cẁ
┘ ┤ ŀ▬Ò ╚¼ŀŻ ╚◘┘ δ ╒Ώġ ẁ
┘ ┤ ŀ▬Ò ╚¼ŀŻ ╚◙┘ δ ╒Ώġ ẁ
┘ ┤ ŀ▬Ò ╚¼ŀŻ ╚ď ╠¼tŽΩĤġġġẁ

     8Ò┤ ŖŗŦ
     ŧ
ŧ
```



- □ Called the default constructor
  - ■ The default constructor has no parameters
  - ■ If you don’t include one, Java will SOMETIMES put one there automatically

Another constructor

- Another constructor:

```
8Ò┤ Ŗ8δť'δ┤ ð·ŀŻ ╙◘ ·ŀŻ ╙◙ ·ŀŻ ╙ď'ŗŦ
    ðδť'δ┤ ΏðẁΩ
    ◘┘ δ ╒Ώ◘ ẁ
    ◙┘ δ ╒Ώ◙ ẁ
    ď' ╠¼ť'Ωď'ẁ
ŧ
```

- This constructor takes in four parameters
- The instance variables in the object are set to those parameters
- This is called a specific constructor
  - An constructor you provide that takes in parameters is called a specific constructor

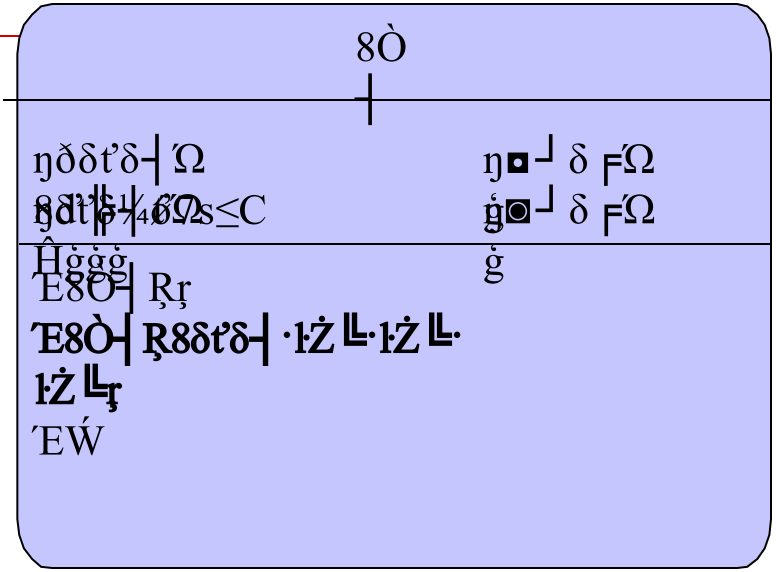
Our Car class so far

```
ðtʼÒ ╒ ╒8Ò┤ Ŧ
┘┤ l▬Ò╚¼8δtʼδ┤ ðδtʼδ┤ Ώ
          8δtʼδ┤ ǿ7s≤Cẁ
┘┤ l▬Ò╚¼l·Ż╚◘┘ δ ╒Ώģẁ
┘┤ l▬Ò╚¼l·Ż╚◙┘ δ ╒Ώģẁ
┘┤ l▬Ò╚¼l·Ż╚ď╠¼tʼΏĤģģģẁ

     8Ò┤ ŖŗŦ
     ŧ

     8Ò┤ Ŗ8δtʼδ┤ ð·l·Ż╚◘·l·Ż╚◙·l·Ż╚ďŗŦ
          ðδtʼδ┤ Ώðẁ
          ◘┘ δ ╒Ώ◘ẁ
          ◙┘ δ ╒Ώ◙ẁ
          ď╠¼tʼΏďẁ
     ŧ

ŧ
```



8Ò
┤

ŋðδtʼδ┤ Ώ
ŋ8δtʼδ╠¼ǿ7s≤C
Ĥģģģ

ŋ◘┘ δ ╒Ώ
ŋ◙┘ δ ╒Ώ
ģ

Έ8Ò┤ Ŗŗ

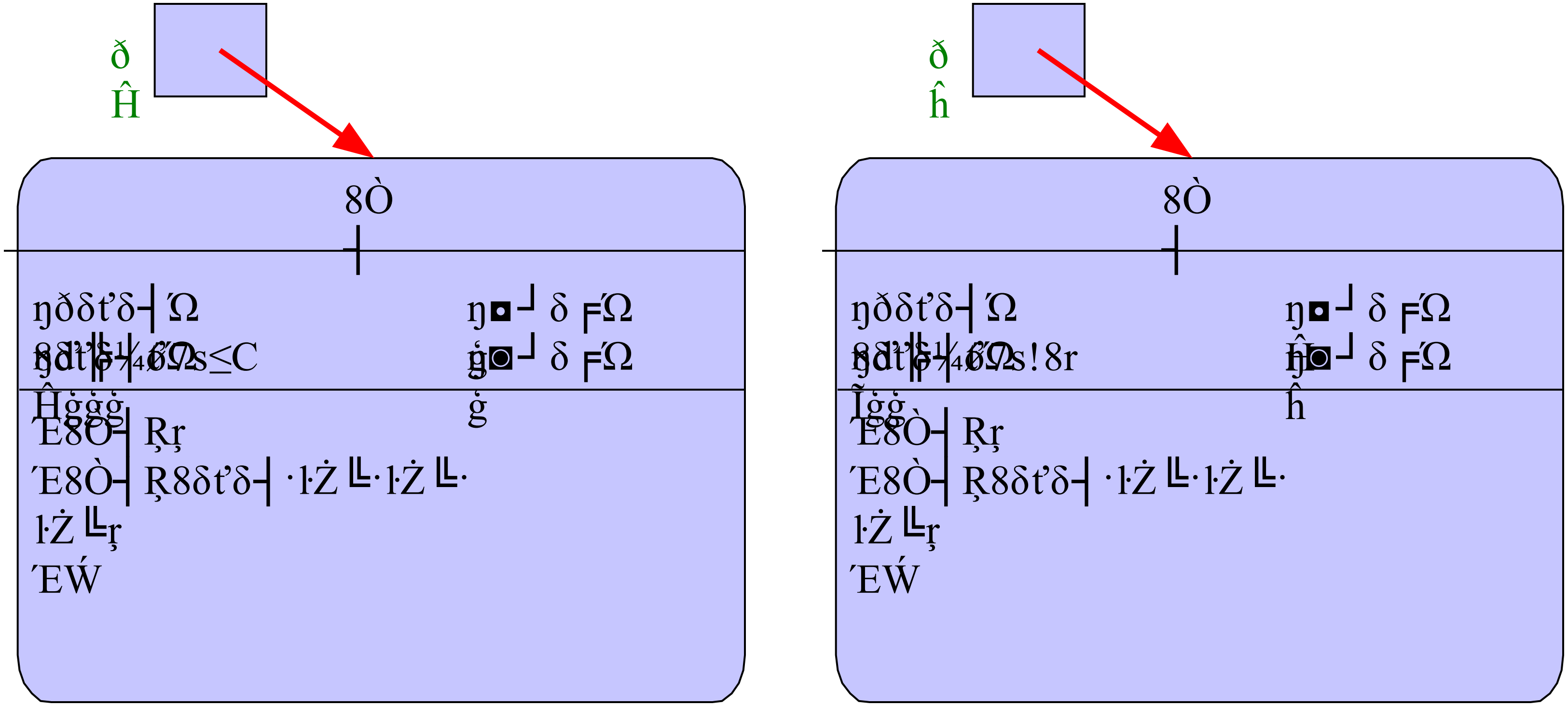
Έ8Ò┤ Ŗ8δtʼδ┤ ·l·Ż╚·l·Ż╚·
l·Ż╚ŗ

ΈẂ

Using our Car class

- □ Now we can use both our constructors:

8Ò┤ ðĤΏΩŻ¼▲ 8Ò┤ Ŗŗẁ
8Ò┤ ðĥΏΩŻ¼▲ 8Ò┤ Ŗ8δťδ┤ ǿ7s!8r·Ĥ·ĥ·Ĩġġŗẁ



So what does private mean?

Note that it’s a different class!

- Consider the following code

```
ðťÒ ╒ ╒8Ò┤ •ŀŹ ╠ťÒ ╚ŀδŻŦ
    ┘ ╠Ðťŀð ╒╚Ò ╚ŀð▬δŀ¹ŹÒŀŻŖ•╚┤ ŀŻÐŜŝÒ┤ Ð ╒ŗŦ
    8Ò┤ ðΏŻ¼▲ 8Ò┤ Ŗŗẁ
    •◙ ╒╚¼Źǿδ╠╚ǿ┘ ┤ŀŻ ╚ťŻŖðǿď╠¼ťŗẁ
    ŧ
ŧ
```

- Recall that fuel is a private instance variable in the Car class
- Private means that code outside the class CANNOT access the variable
  - For either reading or writing
- Java will not compile the above code
  - If fuel were public, the above code would work

So how do we get the fuel of a Car?

- □ Via accessor methods in the Car class:

```
┘ ╠Ðť'l·ðl·Ż ╚Ð¼ ╚W ╠¼ť'ŖŗŦ
    ┤ ¼ ╚ ╠┤ Żď' ╠¼ť'ẁ
ŧ

┘ ╠Ðť'l·ð8δť'δ┤ Ð¼ ╚8δť'δ┤ ŖŗŦ
┤ ¼ ╚ ╠┤ Żðδť'δ┤ ẁ
ŧ
```

```
┘ ╠Ðť'l·ðl·Ż ╚Ð¼ ╚–ùδ ╒ŖŗŦ
┤ ¼ ╚ ╠┤ Ż◙┘ δ ╒ẁ
ŧ

┘ ╠Ðť'l·ðl·Ż ╚Ð¼ ╚œùδ ╒ŖŗŦ
    ┤ ¼ ╚ ╠┤ Ż◘┘ δ ╒ẁ
  ŧ
```

- □ As these methods are within the Car class, they can read the private instance variables
- □ As the methods are public, anybody can call them

So how do we set the fuel of a Car?

□ Via mutator methods in the Car class:

```
┘ ╟Ðt’l·ð▬δl·¹ ╒¼ ╚W ╟¼t’Ŗl·Ż ╚d’ŗŦ
      d’ ╟¼t”Ωd’ẁ
ŧ

┘ ╟Ðt’l·ð▬δl·¹ ╒¼ ╚8δt’δ┤ Ŗ8δt’δ┤ ðŗŦ
      ðδt’δ┤ Ώðẁ
ŧ
```

```
┘ ╟Ðt’l·ð▬δl·¹ ╒¼ ╚œùδ ╒Ŗl·Ż ╚◘ ŗŦ
◘┘ δ ╒Ώ◘ ẁ
ŧ

┘ ╟Ðt’l·ð▬δl·¹ ╒¼ ╚–ùδ ╒Ŗl·Ż ╚◙ ŗŦ
◙┘ δ ╒Ώ◙ ẁ
ŧ
```

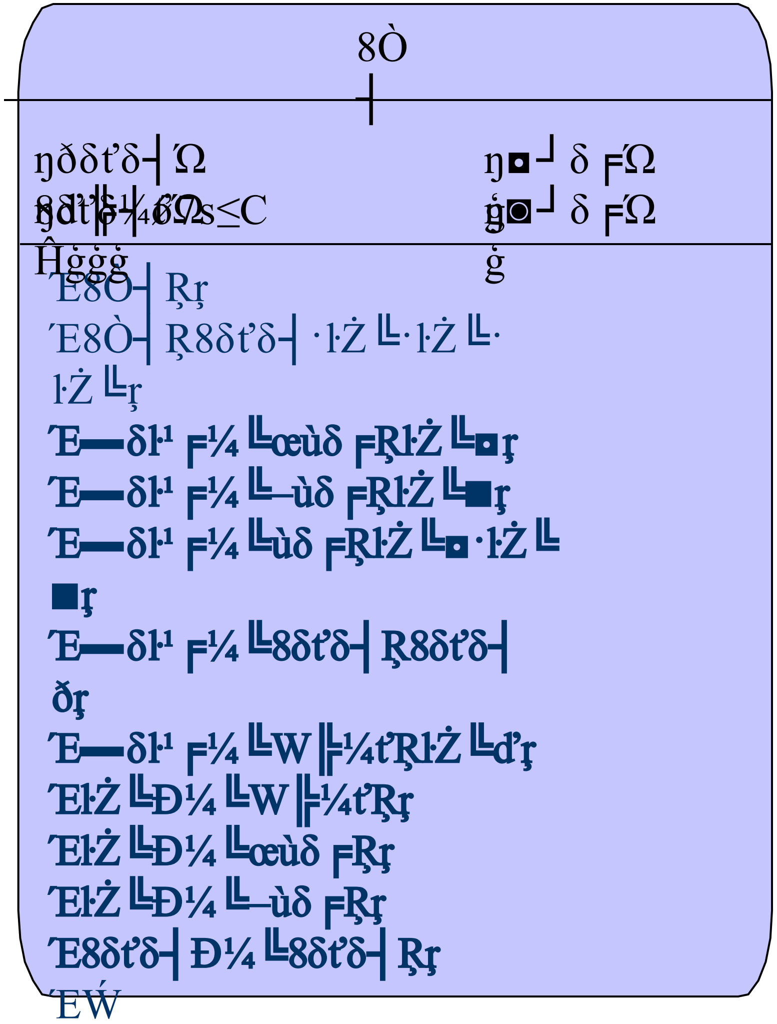
□ As these methods are within the Car class, they can read the private instance variables

□ As the methods are public, anybody can call them

Why use all this?

- □ These methods are called a get/set pair
  - ■ Used with private variables

- □ Our Car so far:



Back to our specific constructor

```
ðťÒ ╒ ╒8Ò┤ Ŧ
    ┘┤ŀ▬Ò╙¼8δťδ┤ ðδťδ┤ Ώ
             8δťδ┤ ǿ7s≤Cẁ
    ┘┤ŀ▬Ò╙¼ŀŻ ╙◘┘ δ ╒Ώģẁ
    ┘┤ŀ▬Ò╙¼ŀŻ ╙◙┘ δ ╒Ώģẁ
    ┘┤ŀ▬Ò╙¼ŀŻ ╙ď╠¼ťΏĤģģģẁ

    8Ò┤ Ŗ8δťδ┤ ð·
        ŀŻ ╙◘ ·ŀŻ ╙◙ ·ŀŻ ╙ďŗŦ
        ðδťδ┤ Ώðẁ
        ◘┘ δ ╒Ώ◘ẁ
        ◙┘ δ ╒Ώ◙ẁ
        ď╠¼ťΏďẁ
ŧ
ŧ
```

```
ðťÒ ╒ ╒8Ò┤ Ŧ
    ┘┤ŀ▬Ò╙¼8δťδ┤ ðδťδ┤ Ώ
             8δťδ┤ ǿ7s≤Cẁ
    ┘┤ŀ▬Ò╙¼ŀŻ ╙◘┘ δ ╒Ώģẁ
    ┘┤ŀ▬Ò╙¼ŀŻ ╙◙┘ δ ╒Ώģẁ
    ┘┤ŀ▬Ò╙¼ŀŻ ╙ď╠¼ťΏĤģģģẁ

    8Ò┤ Ŗ8δťδ┤ ð·
        ŀŻ ╙◘ ·ŀŻ ╙◙ ·ŀŻ ╙ďŗŦ
        ╒¼ ╙8δťδ┤ Ŗðŗẁ
        ╒¼ ╙œùδ ╒Ŗ◘ŗẁ
        ╒¼ ╙–ùδ ╒Ŗ◙ŗẁ
        ╒¼ ╙W╠¼ťŖďŗẁ
ŧ
ŧ
```

Back to our specific constructor

- Using the mutator methods (i.e. the ‘set’ methods) is the preferred way to modify instance variables in a constructor

So what’s left to add to our Car class?

- What else we should add:
  - A mutator that sets both the x and y positions at the same time
  - A means to “use” the Car’s fuel

- Let’s do the first:

```
┘ ╠Ðtľl·ð▬δl·¹ ╒¼ ╚ùδ ╒Ŗl·Ż ╚◘ ·l·Ż ╚◙ŗŦ
      ╒¼ ╚œùδ ╒Ŗ◘ŗẁ
      ╒¼ ╚–ùδ ╒Ŗ◙ŗẁ
ŧ
```

- Notice that it calls the mutator methods

Using the Car’s fuel

- Whenever the Car moves, it should burn some of the fuel
  - For each pixel it moves, it uses one unit of fuel
    - We could make this more realistic, but this is simpler

```
┘ ╟Ðťŀð▬δŀ¹ ╒¼ ╙œùδ ╒ŖŀŻ ╙◘ŗŦ
    ◘┘ δ ╒Ώ◘ẁ
ŧ
```

```
┘ ╟Ðťŀð▬δŀ¹ ╒¼ ╙–ùδ ╒ŖŀŻ ╙◙ŗŦ
    ◙┘ δ ╒Ώ◙ẁ
ŧ
```

```
┘ ╟Ðťŀð▬δŀ¹ ╒¼ ╙œùδ ╒ŖŀŻ ╙◘ŗŦ
    ď’ ╟¼ťŋΏxÒ ╙ĺǿÒĐ ╒
              ŖĐ¼ ╙œùδ ╒Ŗŗŋ◘ŗẁ
    ◘┘ δ ╒Ώ◘ẁ
ŧ
```

```
┘ ╟Ðťŀð▬δŀ¹ ╒¼ ╙–ùδ ╒ŖŀŻ ╙◙ŗŦ
    ď’ ╟¼ťŋΏxÒ ╙ĺǿÒĐ ╒
              ŖĐ¼ ╙–ùδ ╒Ŗŗŋ◙ŗẁ
    ◙┘ δ ╒Ώ◙ẁ
ŧ
```

Using the Car’s fuel

```
┘ ╠ĐťŀðŠ▬δŀ¹ ╒¼ ╚ùδ ╒ŖŀŻ ╚◘ ·ŀŻ ╚◙ŗŦ
       ╒¼ ╚œùδ ╒Ŗ◘ŗẁ
       ╒¼ ╚–ùδ ╒Ŗ◙ŗẁ
ŧ
```

□ Notice that to access the instance variables, the accessor methods are used

□ Math.abs() gets the absolute value of the passed parameter

The main() method

- □ Consider a class with many methods:

```
┘ ╠Ðťľ·ðδťÒ ╒ ╒…Í¼┤ ¼≠δ•╚Ò┤ ╚Ŧ
    ┘ ╠Ðťľ·ð ╒╚Ò╚ľ·ð▬δľ·¹ďδδŖľŻ╚◘ŗŦ
        ỳỳøǿǿ
    ŧ
    ┘ ╠Ðťľ·ð ╒╚Ò╚ľ·ð▬δľ·¹ÐÒ┤ ŖŗŦ
        ỳỳøǿǿ
    ŧ
    ┘ ╠Ðťľ·ð ╒╚Ò╚ľ·ð▬δľ·¹ŹÒľ·ŻŖ•╚┤ ľ·ŻÐŜŝÒ┤ Ð ╒ŗŦ
        ỳỳøǿǿ
    ŧ
ŧ
```

- □ Where does Java start executing the program?
  - ■ Always at the beginning of the main() method!

Running a class without a main() method

- Consider the Car class
  - It had no main() method!
  - Create another class named “CarSimulation” where main function and Car class is declared.

- So let’s try running it…