

# Lecture 4: Writing Classes

CSC 1214: Object-Oriented Programming

# Writing Classes

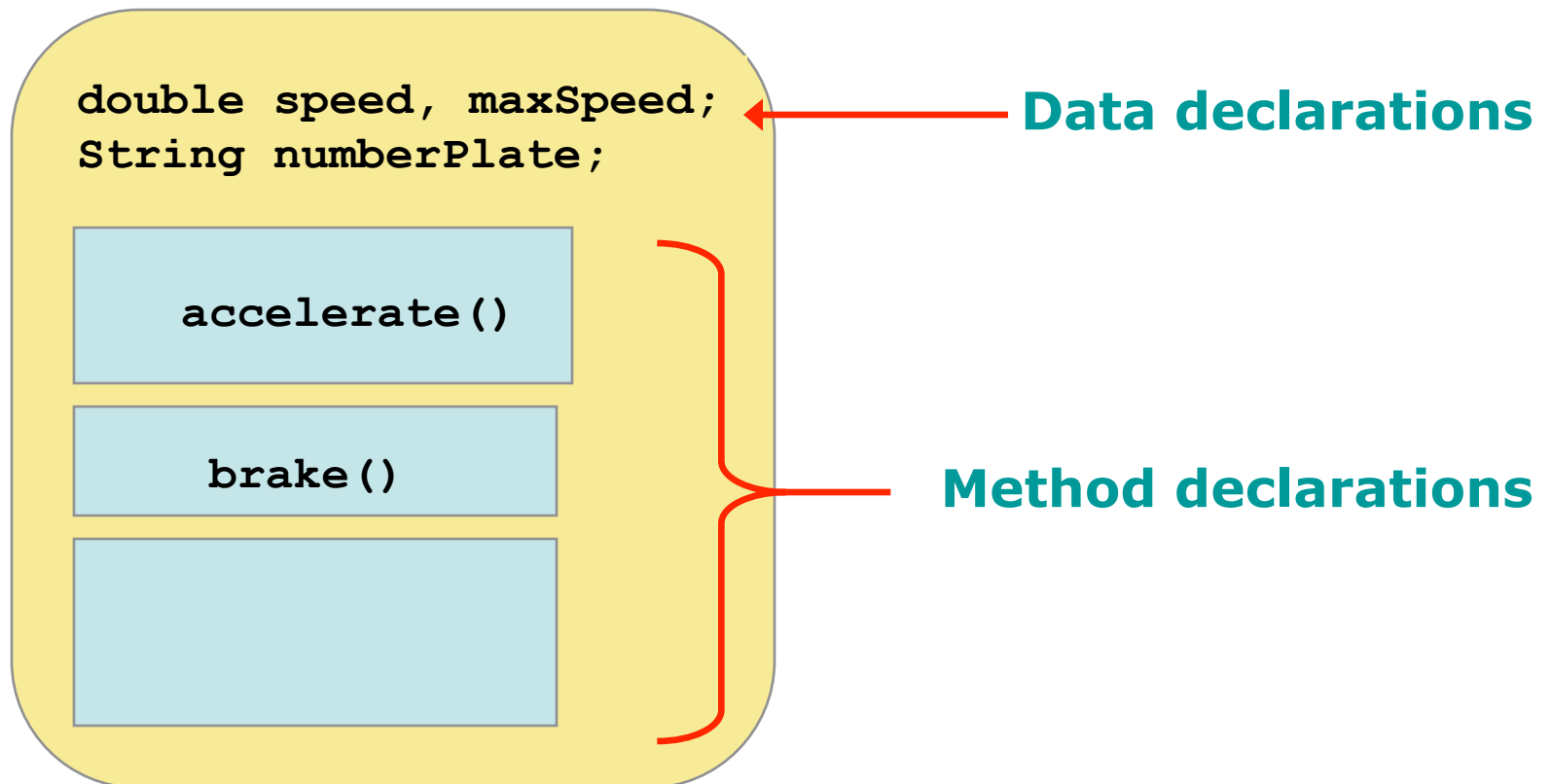
- The programs we have written in previous examples have used classes defined in the Java standard class library
- Now we will begin to design programs that rely on classes that we write ourselves
- The class that contains the **main** method is just the starting point of a program
- True object-oriented programming is based on defining classes that represent objects with well-defined characteristics and functionality

# Classes and Objects

- Recall from Lecture 2 about Object-Oriented programming that: an object has *state* and *behaviour* and a class is *blueprint* of an object.
- **Example:** Suppose you need to write a traffic simulation program that monitors cars going past a junction. Each car has a speed, a maximum speed, and a number plate that uniquely identifies it. In traditional programming languages you'd have two floating point and one string variable for each car. With a class you combine these into one entity.
- We can represent a car in software by designing a class called **Car** that models this state and behaviour (The class serves as the blueprint for a car object). We can then instantiate as many **car** objects as we need from the **Car** class

# The Car Class

- Remember a class can contain data declarations (*state*) and method declarations (*behaviour*).



# The Car Class

```
class Car {  
    String numberPlate; // e.g. "UAN 485E"  
    double speed = 0.0; // in kilometers per hour  
    double maxSpeed; // in kilometers per hour  
}  
}
```

# The Car Class

```
class Car {  
    String numberPlate; // e.g. "UAN 485E"  
    double speed = 0.0; // in kilometers per hour  
    double maxSpeed; // in kilometers per hour  
}  
}
```

**Question:** What other fields might this class need?

# Object Instantiation

```
Car myCar; //declares the type of the variable myCar  
myCar = new Car(); // creates an object and assigns it to myCar
```

or


```
Car myCar = new Car();
```

- To instantiate an object in Java, you use the keyword **new** followed by a call to the class's constructor
- Classes are types and variables of a class type need to be declared just like variables that are ints or doubles
- A constructor is a special method that creates a new instance of a class. A constructor has the same name as the class

# Accessing an Object's Fields

- Object fields are accessed by their name (within its class)
- Code that is outside the object's class must use an object reference , followed by the **dot (.) operator**, followed by a simple field name, as in: **objectReference.fieldName**
- `myClass` is an object of class `Car` . It has three fields which can be accessed as follows:

```
myCar.numberPlate;  
myCar.speed;  
myCar.maxSpeed;
```



The dot (.)  
operator



# Object Example

```
Car myCar = new Car();
```

```
myCar.numberPlate = "UAN 485E";
```

```
myCar.speed = 47.0;
```

```
myCar.maxSpeed = 180.0;
```

```
System.out.print(myCar.numberPlate);
```

```
System.out.print("is moving at" + myCar.speed);
```

```
System.out.print("kilometers per hour");
```

# Using a Car Object in a Different Class

```
class CarTest {  
    public static void main(String args[]) {  
        Car myCar = new Car();  
        myCar.numberPlate = "UAN 485E";  
        myCar.speed = 47.0;  
        myCar.maxSpeed = 180.0;  
  
        System.out.print(myCar.numberPlate);  
        System.out.print("is moving at" + myCar.speed);  
        System.out.print("kilometers per hour");  
    }  
}
```

# Using a Car Object in a Different Class

- This program requires not just the CarTest class but also the Car class. To make them work together put the Car class in a file called Car.java. Put the CarTest class in a file called CarTest.java. Put both of these files in the same directory.
- Then compile both files in the usual way. Finally run CarTest
- Note that Car does not have a main() method so you cannot run it. It can exist only when called by other programs that do have main() methods.
- Many of the applications you write will use multiple classes. It is customary in Java to put every class in its own file. You'll learn how to use packages to organize your commonly used classes in different directories. For now keep all your .java source code and .class byte code files in one directory

# Methods

- Data is not of much use unless you can do things with it. For this purpose classes have methods.
- Fields say what a class is. Methods say what a class does.
- The fields and methods of a class are collectively referred to as the members of the class.
- The classes you've encountered up till now have mostly had a single method, `main()`. However, in general classes can have many different methods that do many different things.
- For instance the `Car` class might have a method to make the car go as fast as it can.

# Car Methods

```
class Car {  
    String numberPlate; // e.g. "UAN 485E"  
    double speed = 0.0; // in kilometers per hour  
    double maxSpeed; // in kilometers per hour  
  
    //accelerate to maximum speed  
    //put the pedal to the metal  
    void accelerate() {  
        this.speed = this.maxSpeed;  
    }  
}
```

You use the `this` keyword to refer to fields in the current object

# Invoking/Calling Methods

```
class CarTest {  
    public static void main(String args[]){  
        Car myCar = new Car();  
        myCar.numberPlate = "UAN 485E";  
        myCar.speed = 47.0;  
        myCar.maxSpeed = 180.0;  
  
        System.out.print(myCar.numberPlate);  
        System.out.print("is moving at"+ myCar.speed);  
        System.out.print("kilometers per hour");  
  
        myCar.accelerate();  
  
        System.out.print(myCar.numberPlate);  
        System.out.print("is moving at"+ myCar.speed);  
        System.out.print("kilometers per hour");  
    }  
}
```

# Exercises

- Reading exercise:
  1. Class members (class variables and class methods)
  2. Data scope
- Practice exercise:
  3. Extend the Car class to include:
    - (I) a field that hold the car's model year, and a field that hold's the car's make.
    - (II) A brake method that reduces the current car's speed by 5 whenever it is called.

Demonstrate the class in a program that creates a Car object, and then calls the accelerate method once. After the call to the accelerate method, get the current speed of the car and display it. Then call the brake method three times. After each call to the brake method, get the current speed of the car and display it.