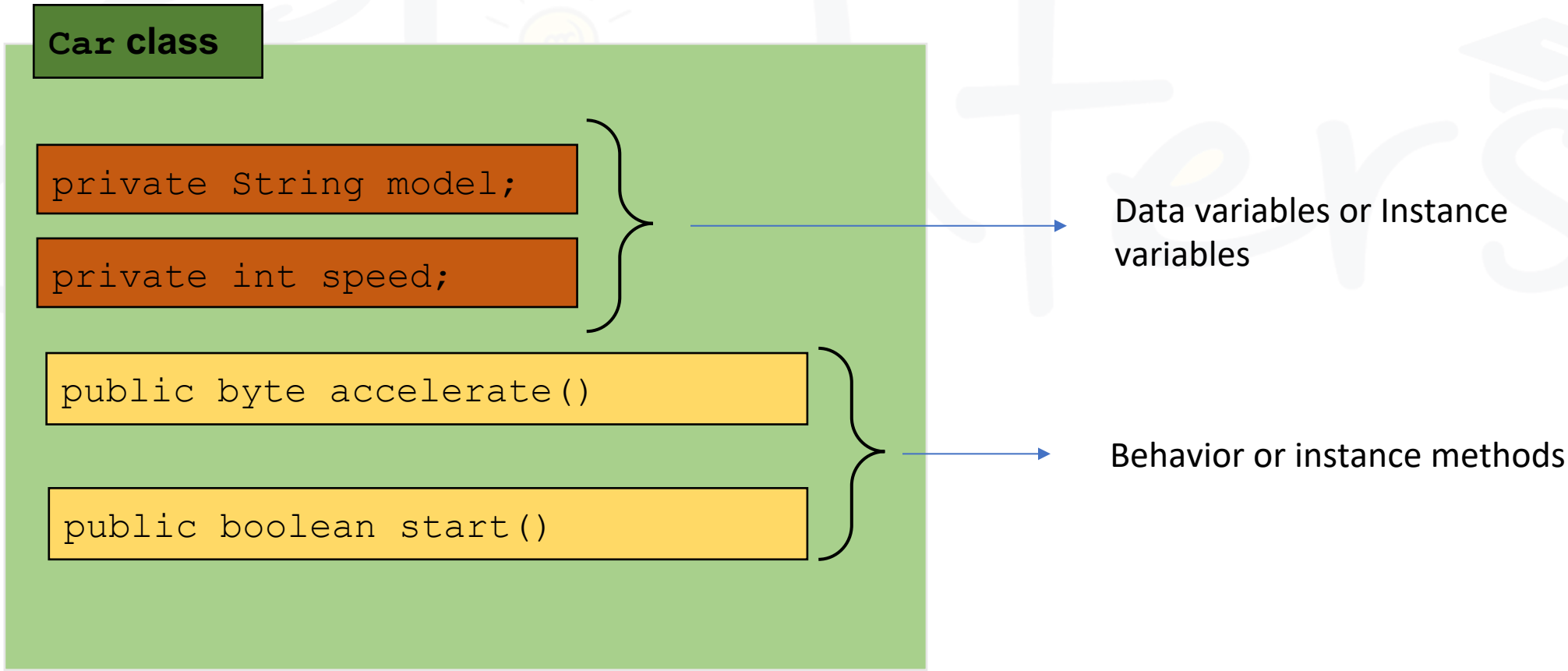
Classes, Objects & Constructors

Class

- classes are template/blueprint that describes the behavior/state that the object of its type support

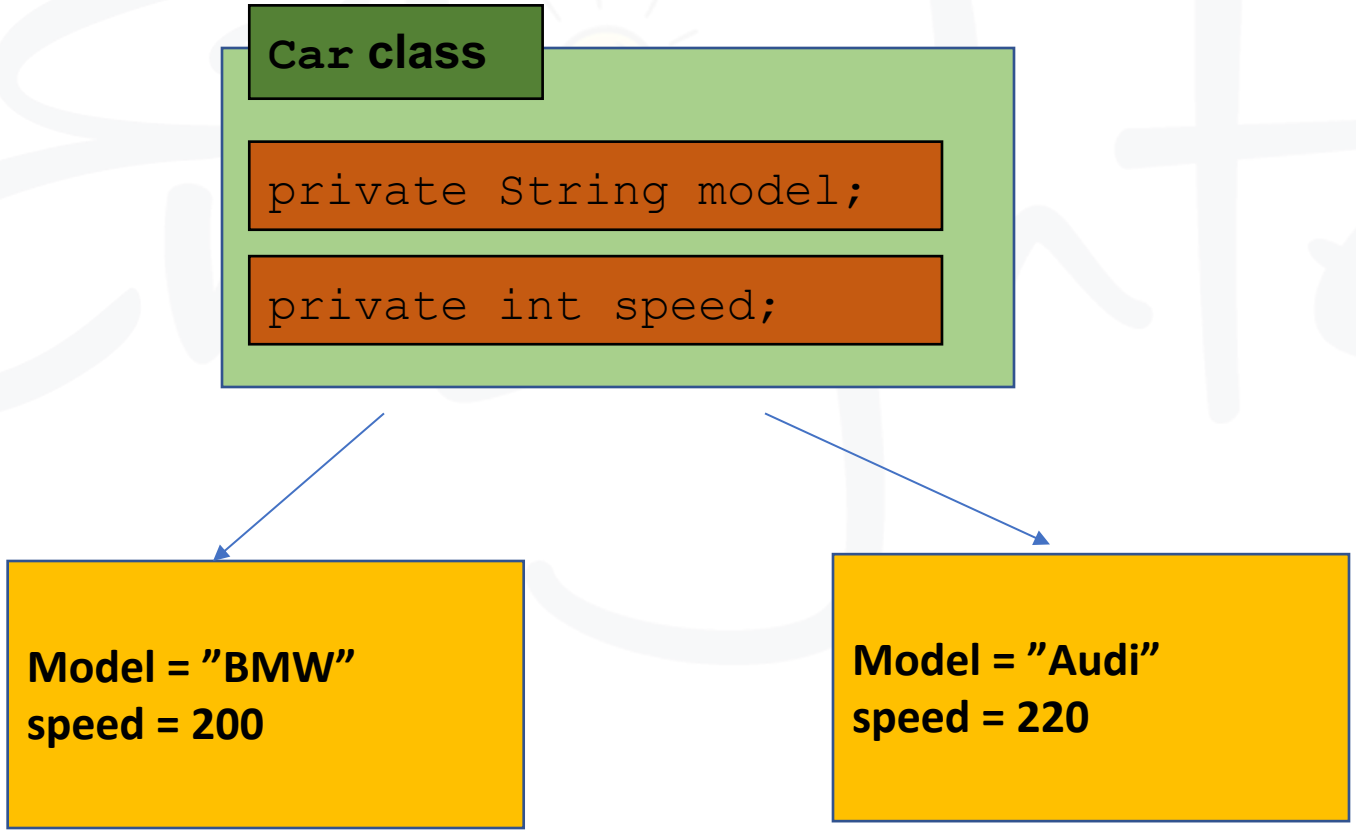


Class

- A class can have any number of methods including both static and non-static methods
- A class can contain any or all of the following variable types.
  - **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
  - **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
  - **Class variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

Objects

Objects are the instance of class and have states and behaviors



Creating Objects

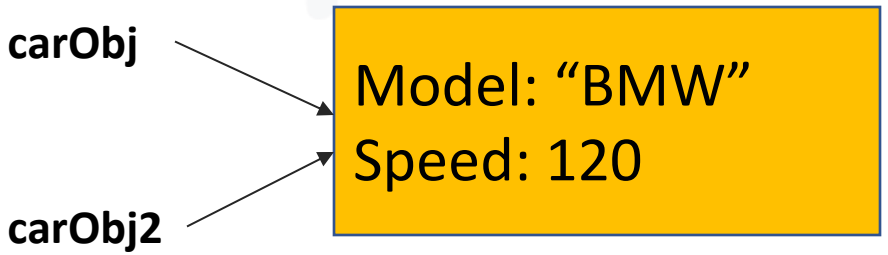
- Objects are created using the `new` operator
- The `new` operator performs the following actions:
  - Allocates memory for the new object
  - Calls a special initialization method in the class, called a *constructor*
  - Returns a *reference* to the new object

```
Car carObj = new Car();
```

Object references

- Assigning one reference to another results in two references to the same object

```
Car carObj = new Car("BMW");
Car carObj2 = carObj;
```



Accessing variables/methods

- Instance variables and methods can be accessed using the dot operator on each object created
- Class level variables and methods can be accessed using the dot operation on the Class itself

Questions??

- Can we access local variables?
- What is the best way to access instance variables?

Methods in java

- A *method* is equivalent to a function or subroutine in other languages
- A method can only be defined within a class definition

```
modifier returnType methodName (argumentList) {
    // method body
    …
}
```

Passing objects as method arguments

- Even though Java does manipulate objects by reference, Java doesn't pass method arguments by reference but it passes them by value

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

        Car initialObj = new Car("BMW", 100);
        ObjectsInJavaEx ex = new ObjectsInJavaEx();
        ex.replaceCarObject(initialObj);

        System.out.println("Current value of passed car object model: "
                + initialObj.model + " speed: " + initialObj.speed);
    }
    private void replaceCarObject(Car carObj) {

        System.out.println("Passed car object model: " + carObj.model
                + " speed: " + carObj.speed);

        // Changing the values of the passed car object
        carObj.model = "Ford";
        carObj.speed = 80;

        // Changing the Object reference itself
        carObj = new Car("Audi", 120);
    }
}

class Car {
    Car(String model, int speed) {
        this.model = model;
        this.speed = speed;
    }
    public String model;
    public int speed;

}
```

Car object created with model as BMW and speed 100

Initial Car object reflecting the values changed within the method call

Changed the values of Model and speed of passed Object

The Car Object is been referenced to a new object with value Audi and speed 120 but after the method call the initial Car object still points to the old updated object

Constructors

- For proper initialization, a class should provide a “constructor”
- Constructor is called automatically, when an object is created
- Characteristics:
  - Usually declared public
  - Has the same name as the class
  - No specified return type
- Compiler supplies a do-nothing no-arg constructor by default
- Constructors also can be overloaded
- Compiler differentiates which constructor is to be called depending upon the number of parameters and their sequence of data types

this & super keyword

- **super** is used to access methods/instance variables of the base class while **this** is used to access methods/instance variables of the current class
- **super(),** it refers to constructor of the base class, and if you write **this(),** it refers to the constructor of the very class where you are writing this code

- **Rules of using this() or super()**
  - If included, this() or super() statement must be the first one in the constructor
  - With the above rule, there cannot be two this() or super() statements in the same constructor (because both cannot be the first).
  - this() or super() must be used with constructors only, that too to call the same class constructor (but not super class constructor)

Static block

- Static block is used for initializing the static variables
- This block gets executed when the class is loaded in the memory
- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program

```
static int number = 5;
static String value = "Test String";
static {
    number = 30;
    value = "Static String";

}
```

Instance block

- Initializer block contains the code that is always executed whenever an new instance of the class is created
- It can used to initialize/execute the common logic of various constructors of a class

```
int number = 5;
String value = "Test String";
{
    number = 30;
    value = "Instance String";
    System.out.println("Instance block number: " + number + " value: " + value);

}
public static void main(String[] args) {
    ObjectsInJavaEx obj1 = new ObjectsInJavaEx();
    ObjectsInJavaEx obj2 = new ObjectsInJavaEx();
```

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=Cp12
Instance block number: 30 value: Instance String
Instance block number: 30 value: Instance String
```

Exercise

- Create a Student class with name, student id, age and add methods to access this values
- Have multiple constructors with one taking name, another one with name and student id and another with all three arguments
- Have the logic to print out these values as part of the constructor which accepts all the values.
- From rest of constructor call this constructor
- Have a static and instance block