INTRODUCTION TO PROGRAMMING LANGUAGE (JAVA)

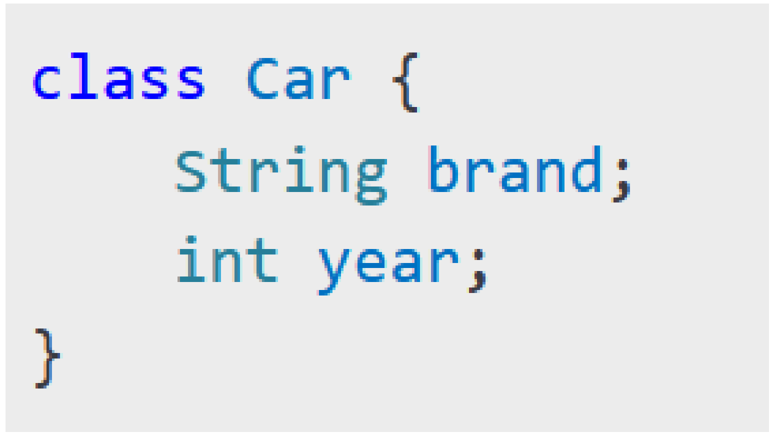
Fariha Zahin
Lecturer
CSE, Southeast University

What is a Class?

•A class is a blueprint for creating objects.

•It defines attributes (variables) and behaviors (methods).

Example:

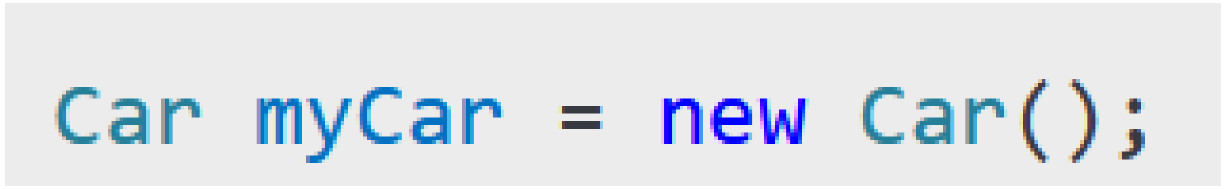


```
class Car {
    String brand;
    int year;
}
```

What is an Object?

•An object is an instance of a class.
•It holds actual values and can invoke methods.

Example:

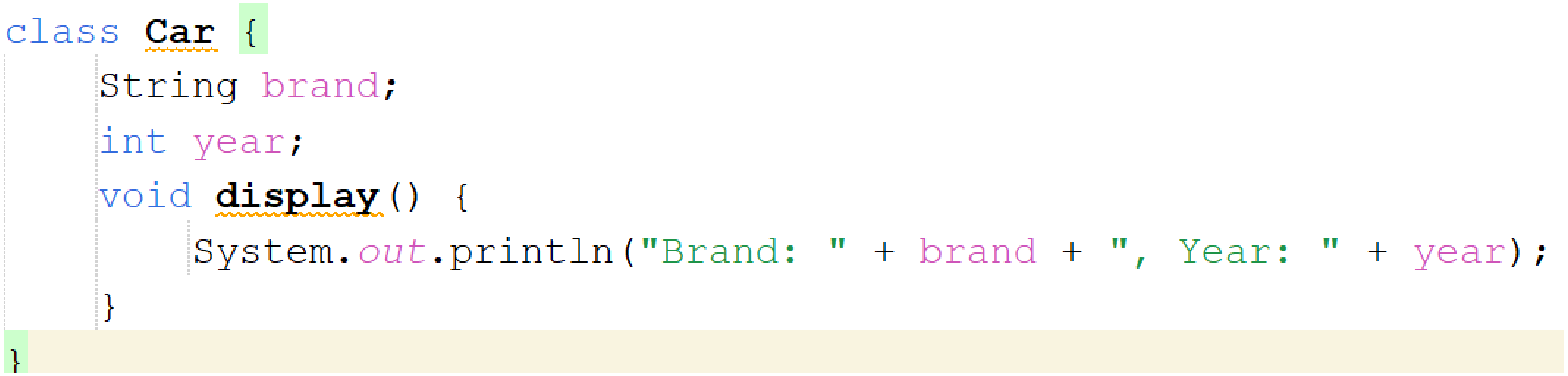


```
Car myCar = new Car();
```

Creating a Class in Java

Define a class with attributes and methods.

Example:



```java
class Car {
    String brand;
    int year;
    void display() {
        System.out.println("Brand: " + brand + ", Year: " + year);
    }
}
```

Creating Objects in Java

1.Instantiate a class using the new keyword..

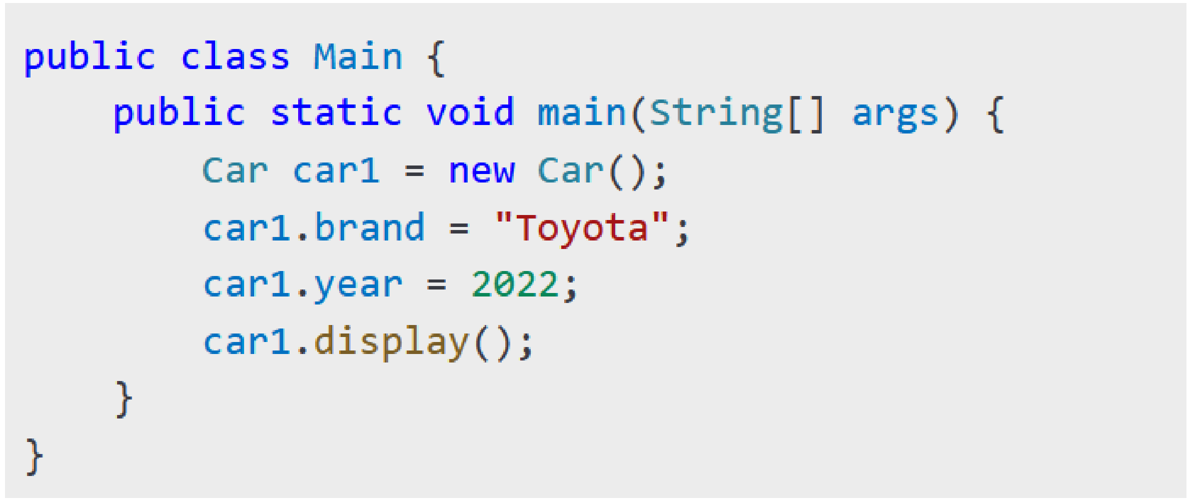
Understanding the new keyword:

•The new keyword dynamically allocates memory for an object.

•It returns a reference to the newly created object.

Steps involved:

1.Memory allocation for the object.
2.Constructor invocation to initialize the object.
3.Reference assignment to the object.



```java
public class Main {
    public static void main(String[] args) {
        Car car1 = new Car();
        car1.brand = "Toyota";
        car1.year = 2022;
        car1.display();
    }
}
```

2.Using Constructors
A constructor initializes objects.

Properties of a Constructor in Java

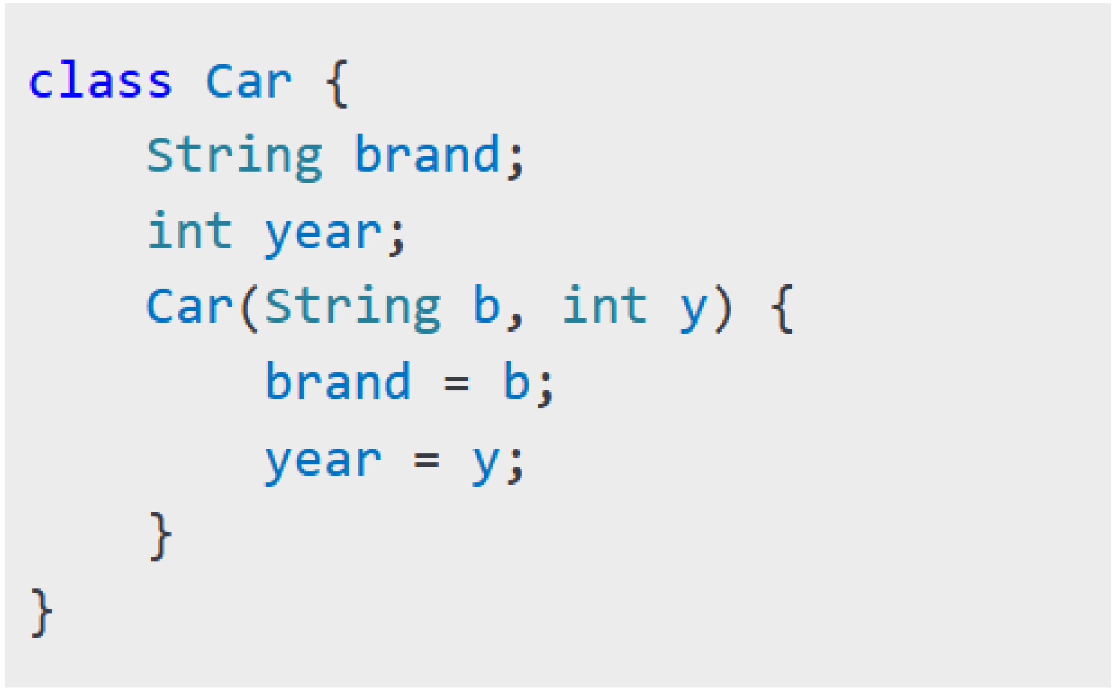
1.Same Name as Class
    A constructor must have the same name as the class.

2.No Return Type
    A constructor does not have a return type, not even void

3.Automatically Called
    It is automatically invoked when an object of the class is created.

4.Can Be Overloaded
    Multiple constructors can be defined with different parameters (Constructor Overloading).

5.Cannot Be Inherited
    Constructors are not inherited by subclasses, but a subclass can call the parent class constructor using super keyword.



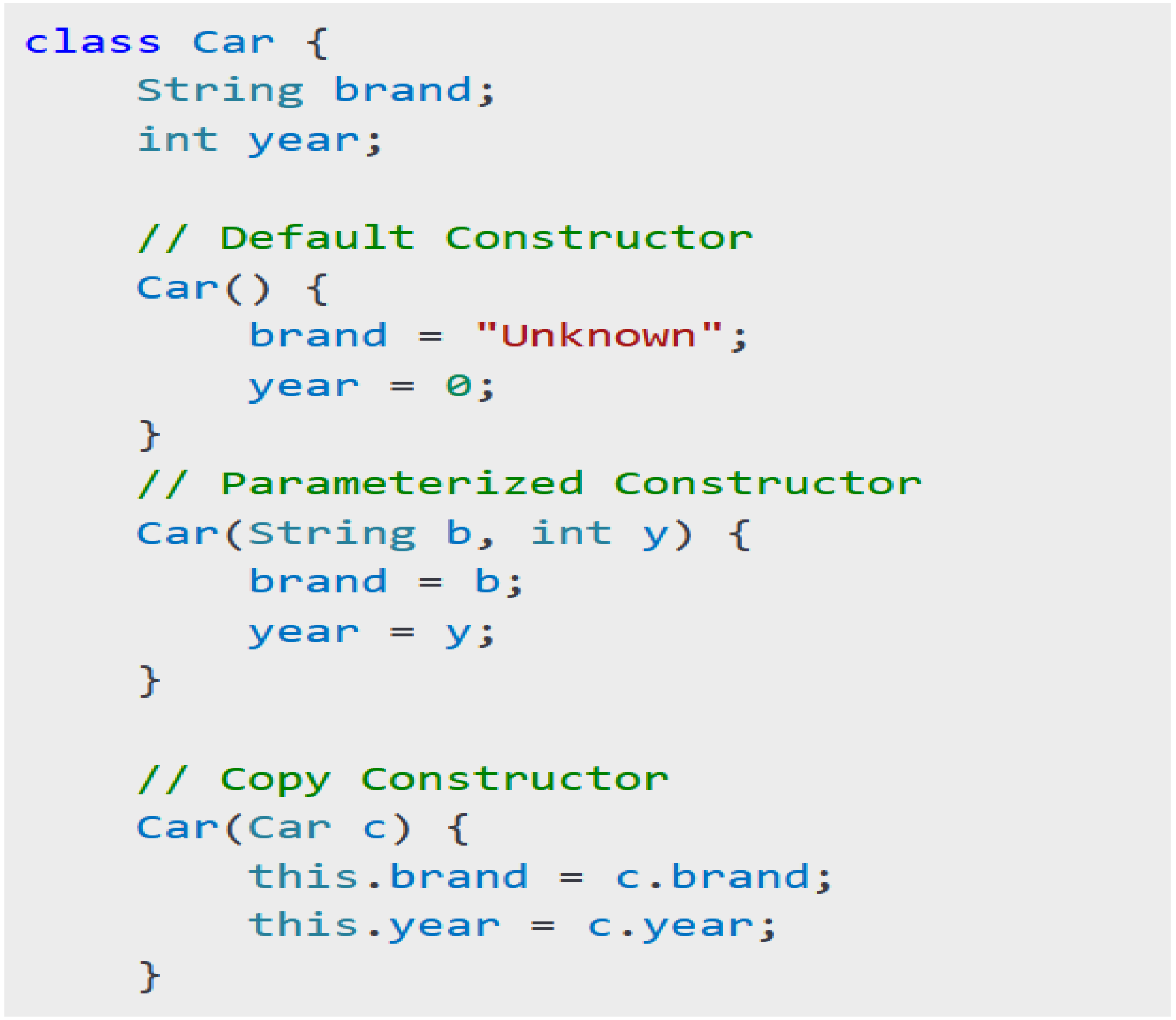
```
class Car {
    String brand;
    int year;
    Car(String b, int y) {
        brand = b;
        year = y;
    }
}
```

Types of Constructors:

1.**Default Constructor** - No parameters, initializes default values.

2.**Parameterized Constructor** - Takes arguments to initialize object properties.

3.**Copy Constructor** - Creates a new object by copying another object's values.

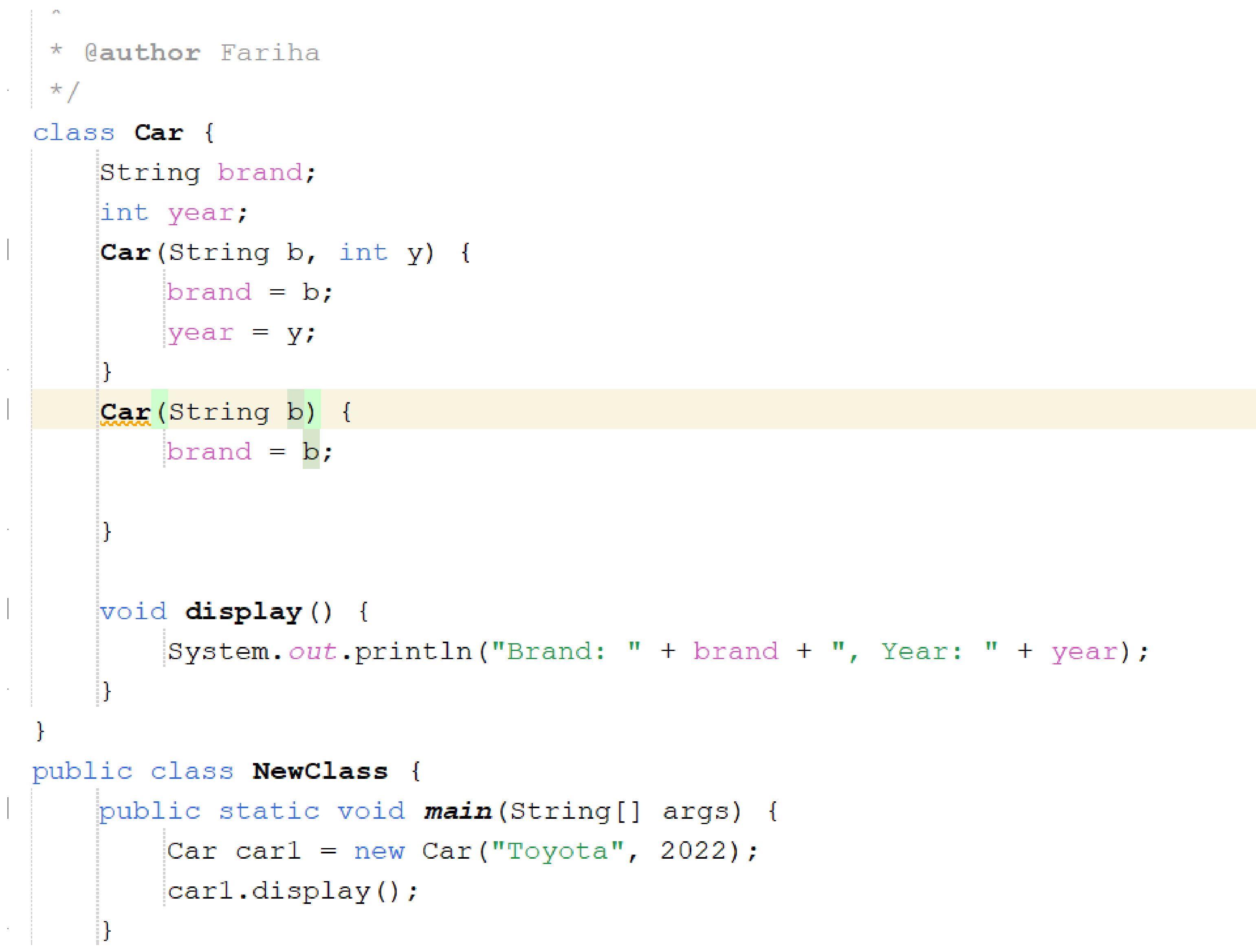


```
class Car {
    String brand;
    int year;

    // Default Constructor
    Car() {
        brand = "Unknown";
        year = 0;
    }
    // Parameterized Constructor
    Car(String b, int y) {
        brand = b;
        year = y;
    }

    // Copy Constructor
    Car(Car c) {
        this.brand = c.brand;
        this.year = c.year;
    }
```

Constructor Overloading in Java
Definition:
Constructor overloading in Java allows a class to have multiple constructors with different parameter lists. The appropriate constructor is selected based on the arguments passed during object creation.



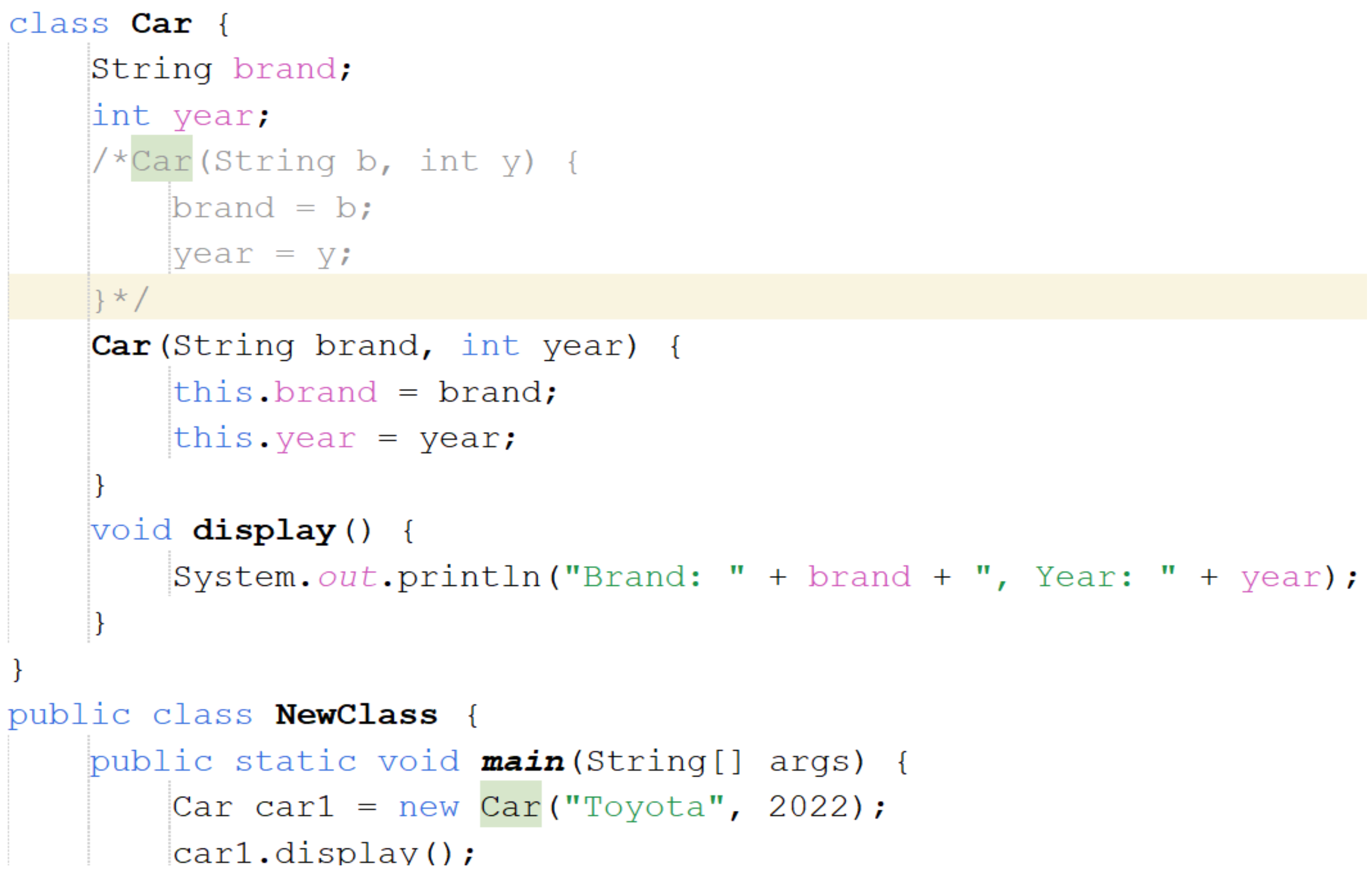
```
 * @author Fariha
 */
class Car {
    String brand;
    int year;
    Car(String b, int y) {
        brand = b;
        year = y;
    }
    Car(String b) {
        brand = b;

    }

    void display() {
        System.out.println("Brand: " + brand + ", Year: " + year);
    }
}
public class NewClass {
    public static void main(String[] args) {
        Car car1 = new Car("Toyota", 2022);
        car1.display();
    }
```

Use of this keyword:

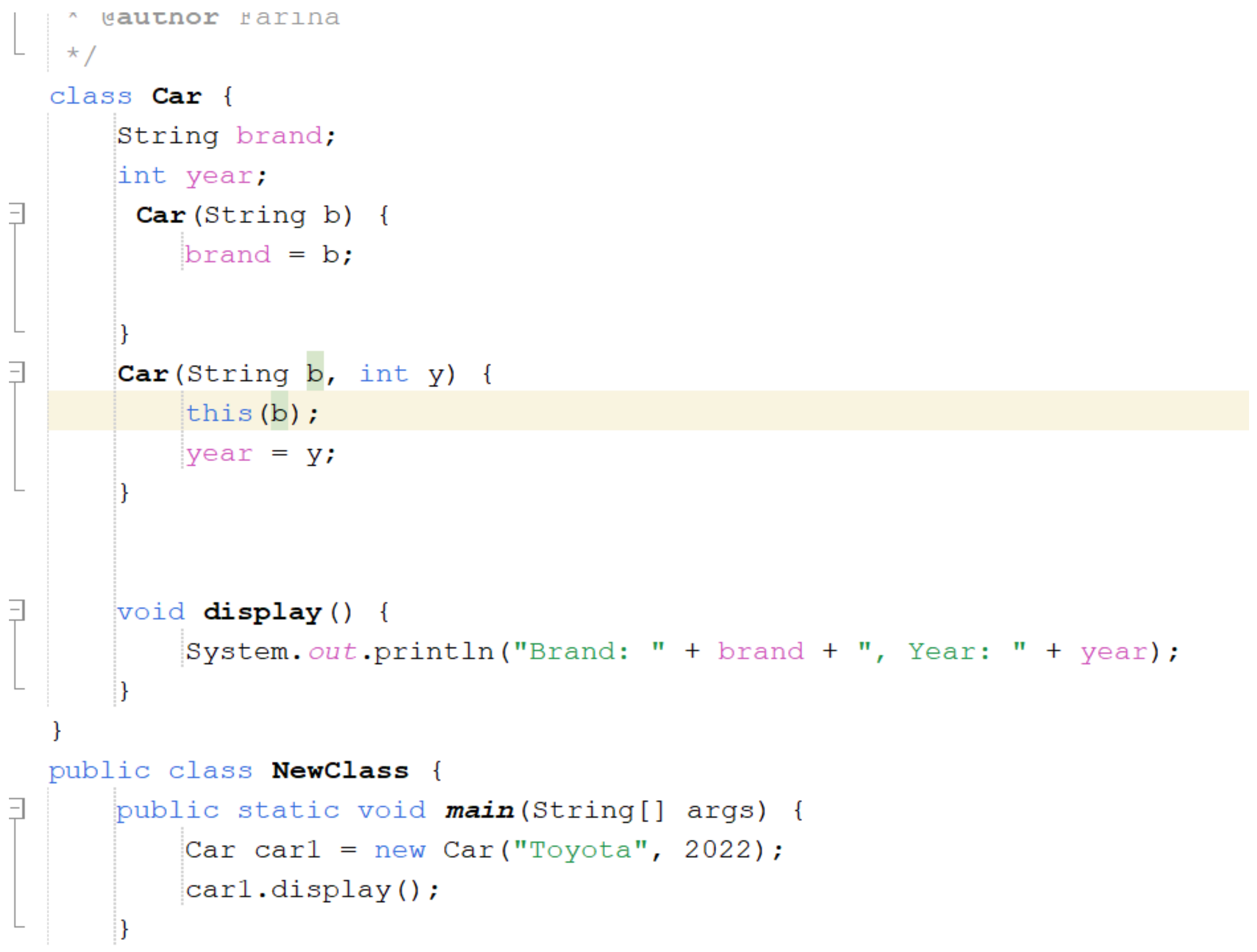
this keyword in Java refers to the current object of the class. It is used to refer to the current object's instance variables, methods, and constructors. It is commonly used in constructors to distinguish between instance variables and parameters when they have the same name.



```
class Car {
    String brand;
    int year;
    /*Car(String b, int y) {
        brand = b;
        year = y;
    }*/
    Car(String brand, int year) {
        this.brand = brand;
        this.year = year;
    }
    void display() {
        System.out.println("Brand: " + brand + ", Year: " + year);
    }
}
public class NewClass {
    public static void main(String[] args) {
        Car car1 = new Car("Toyota", 2022);
        car1.display();
```

• **this()** must be the first statement in a constructor.
•Helps in **constructor chaining** within the same class.

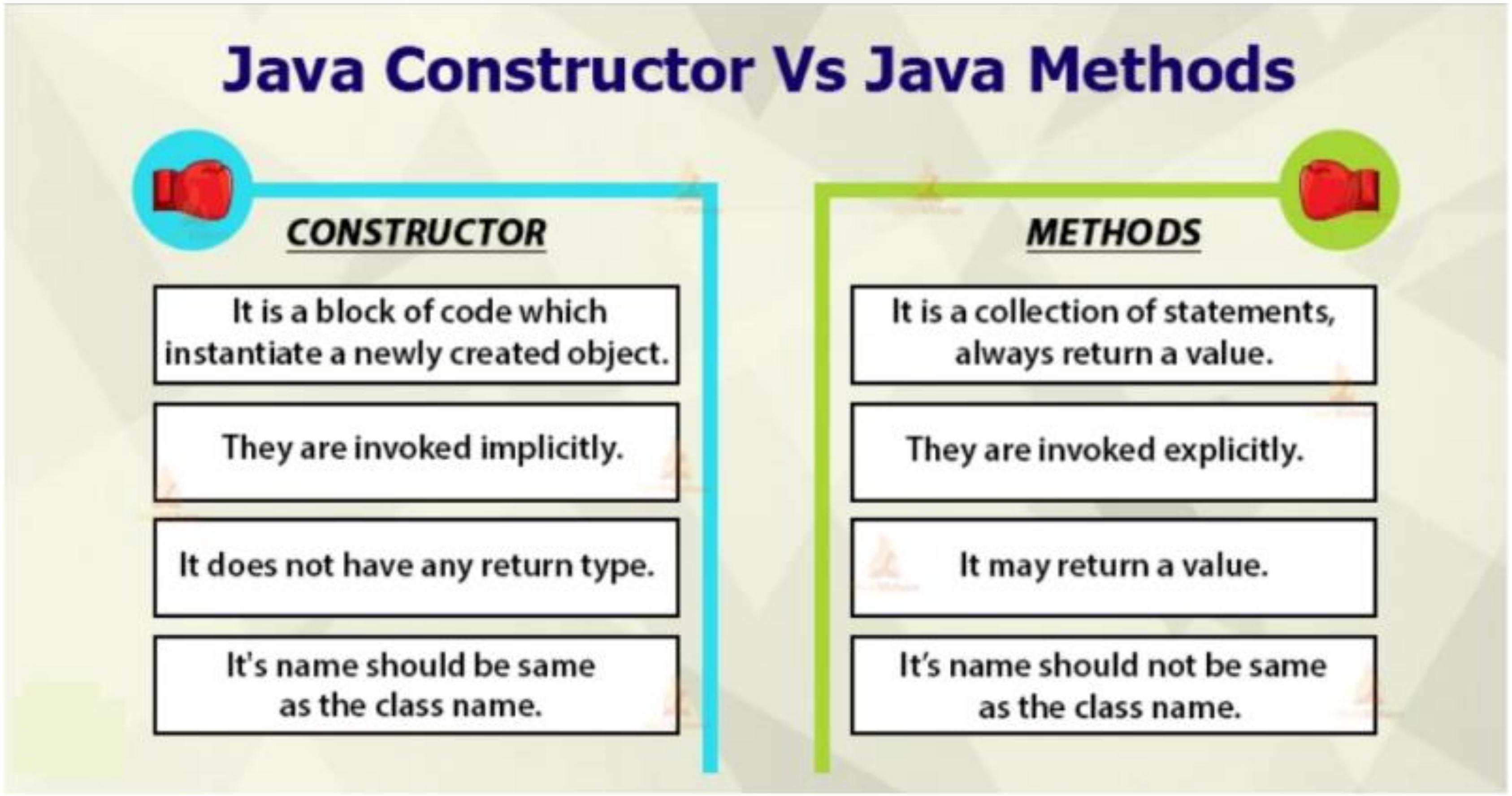
Constructor chaining is the process of calling one constructor from another constructor within the same class or from a parent class using **this().** It helps reduce code duplication and improves reusability.



```
 * @author Farina
 */
class Car {
    String brand;
    int year;
     Car(String b) {
        brand = b;

    }
    Car(String b, int y) {
        this(b);
        year = y;
    }


    void display() {
        System.out.println("Brand: " + brand + ", Year: " + year);
    }
}
public class NewClass {
    public static void main(String[] args) {
        Car car1 = new Car("Toyota", 2022);
        car1.display();
    }
```



Java Constructor Vs Java Methods

CONSTRUCTOR	METHODS
It is a block of code which instantiate a newly created object.	It is a collection of statements, always return a value.
They are invoked implicitly.	They are invoked explicitly.
It does not have any return type.	It may return a value.
It's name should be same as the class name.	It's name should not be same as the class name.