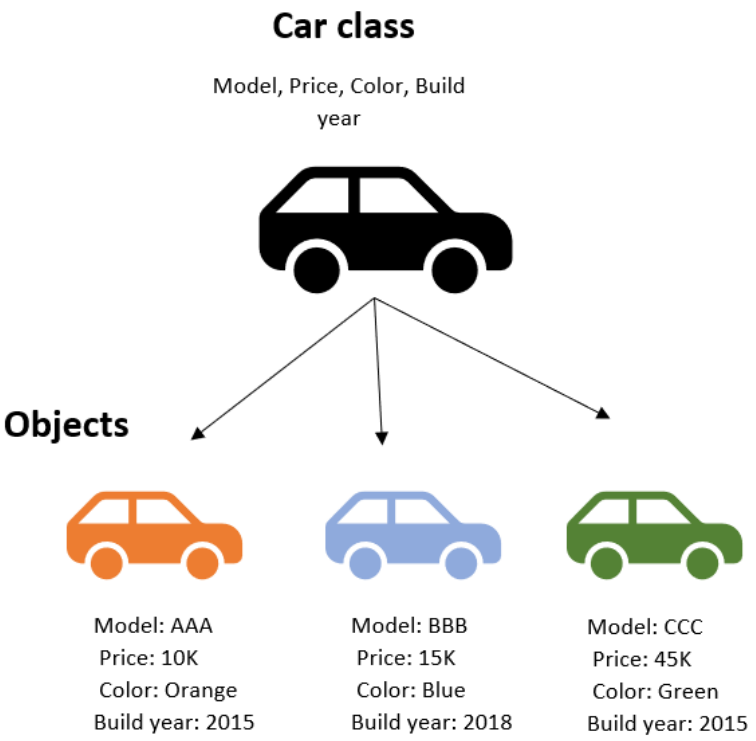
Classes and Objects

Classes and Objects

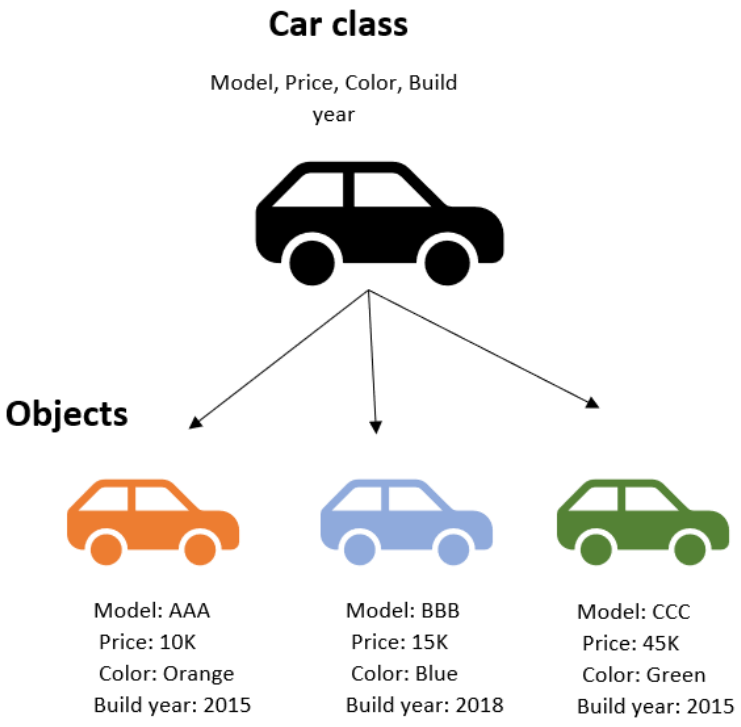
- Object-Oriented Programming (OOP) is a programming paradigm that involves creating objects, which encapsulate both data and methods. In contrast, procedural programming focuses on writing procedures or methods that operate on data.
- Object-Oriented Programming offers numerous advantages compared to procedural programming:
  - Execution Efficiency: OOP tends to be faster and more straightforward to execute, enhancing overall program performance.
  - Clear Program Structure: OOP provides a well-defined structure for programs, making it easier to organize and understand the code.
  - DRY Principle: OOP promotes the "Don't Repeat Yourself" (DRY) principle, reducing redundancy in code. This leads to more maintainable, modifiable, and debuggable Java code.
  - Enhanced Code Maintenance: The modular nature of OOP facilitates easier maintenance, as modifications and debugging become more straightforward, contributing to code robustness.
  - Reusability: OOP allows for the creation of fully reusable applications with less code, resulting in shorter development times and increased efficiency.

Classes and Objects



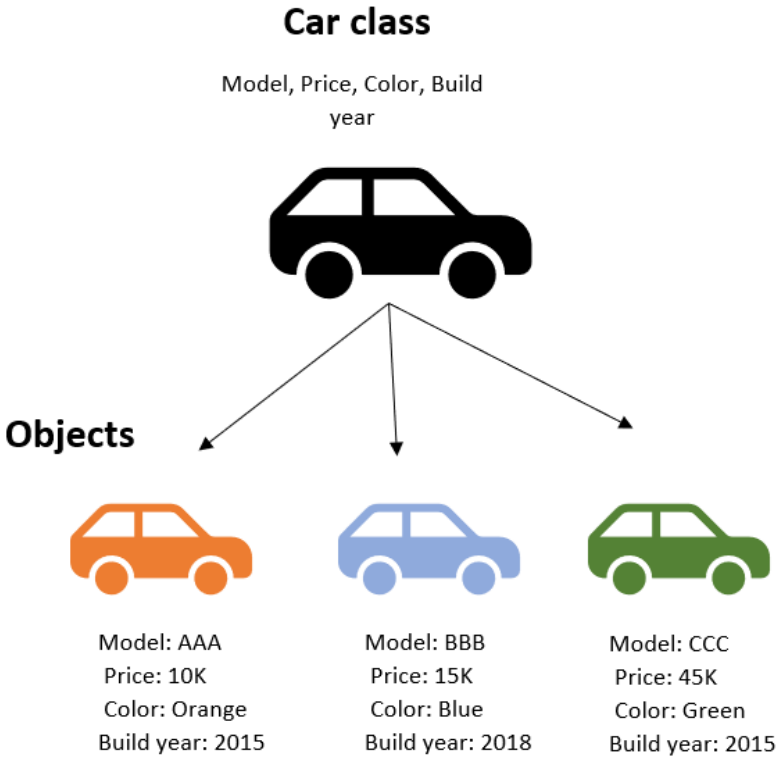
- Class:
  - A class in Java is a blueprint or a template for creating objects.
  - It defines a data structure (attributes or fields) and behavior (methods) that will be shared by all objects of that type.
  - Classes encapsulate the properties and actions that are common to a group of objects.

Classes and Objects



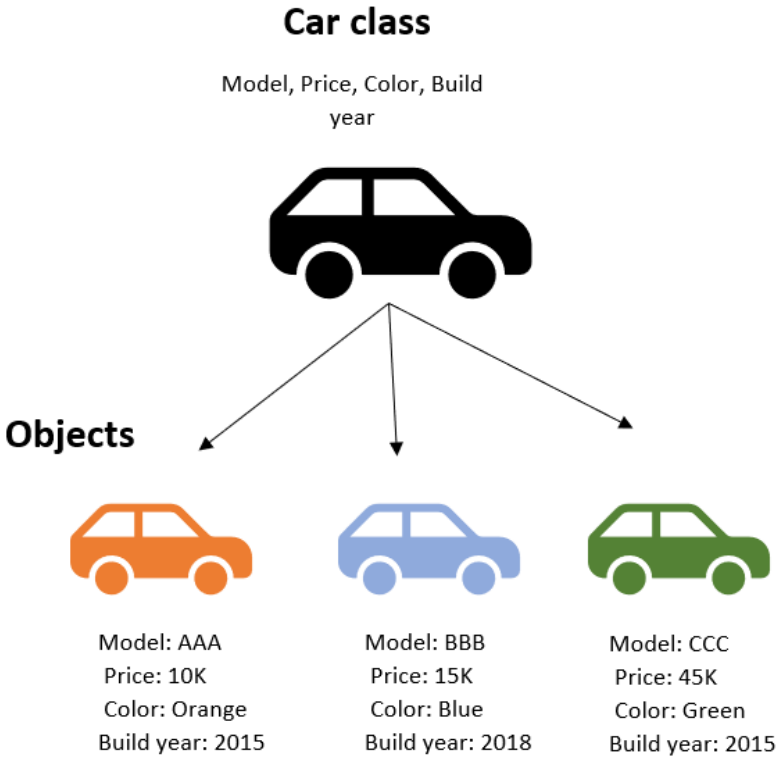
- Object:
  - An object is an instance of a class.
  - When you create an object, you are creating a specific realization of the class, with its own set of values for attributes.
  - Objects represent the entities that your program manipulates.

Classes and Objects




```
1  public class Car {
2      // Attributes or fields
3      String brand;
4      int year;
5      // Methods
6      void startEngine() {
7          System.out.println("Engine started!");
8      }
9      void stopEngine() {
10         System.out.println("Engine stopped!");
11     }
12 }
```

Classes and Objects




```
#Put this code inside Class Car:
public static void main(String[] args) {
        // Creating objects of the Car class
        Car myCar = new Car();
        Car anotherCar = new Car();
        // Setting values for attributes
        myCar.brand = "Toyota";
        myCar.year = 2022;
        anotherCar.brand = "Honda";
        anotherCar.year = 2021;
        // Invoking methods on objects
        myCar.startEngine();
        anotherCar.startEngine();
    }
```

Classes and Objects

- Everything in Java is associated with classes and objects, along with its attributes and methods.
- For example: in real life, a car is an object.
- The car has attributes, such as weight and color, and methods, such as drive and brake
- A Class is like an object constructor, or a "blueprint" for creating objects.

Classes and Objects

- **Create a Class**
- To create a class, use the **keyword *class***

- **Create a class named "Main" with a variable x:**

```
1 public class Main {
2 int x = 5;
3 }
```

Classes and Objects

- ***Create an Object***
- *In Java, an object is created from a class*
- *To create an object of Main, specify the class name, followed by the object name, and use the keyword new:*

- **Create an object called "myObj" and print the value of x:**

```
1  public class Main {
2    int x = 5;
3    public static void main(String[] args) {
4      Main myObj = new Main();
5      System.out.println(myObj.x);
6    }
7  }
```

Classes and Objects

- ***Multiple Objects***
- *We can create multiple objects of one class:*
- *For Example*
- *Create two objects of Main:*

- **Create an object called "myObj" and print the value of x:**

```
1  public class Main {
2    int x = 5;
3    public static void main(String[] args) {
4      Main myObj1 = new Main();  // Object 1
5      Main myObj2 = new Main();  // Object 2
6      System.out.println(myObj1.x);
7      System.out.println(myObj2.x);
8    }
9  }
```

Classes and Objects

- *Using Multiple Classes*
- *We can also create an object of a class and access it in another class.*
- *This is often used for better organization of classes (one class has all the attributes and methods, while the other class holds the main() method (code to be executed)).*
- *Remember that the name of the java file should match the class name.*
- *In this example, we have created two files in the same directory/folder:*

1. ***Main.java***
2. ***Second.java***

- ***Main.java***

```
1  public class Main {
2    int x = 5;
3  }
```

Second.java

```
4  class Second {
5    public static void main(String[] args) {
6      Main myObj = new Main();
7      System.out.println(myObj.x);
8    }
9  }
```

Classes and Objects

- ***Adding Variables /Class Attributes***
- *Another term for class attributes is fields.*
- *In this example, we are use the term "variable" for x.*
- *It is actually an attribute of the class.*
- *Or we could say that class attributes are variables within a class:*

- *Create a class called "Main" with two attributes: x and y:*
  - *public class Main {*
    - *int x = 5;*
    - *int y = 3;*
  - *}*

Classes and Objects

- ***Adding Methods / Java Class Methods***
- *A method is a block of code which only runs when it is called.*

```
1  public class Main {
2    static void myMethod() {
3      System.out.println("Hello World!");
4    }
5    public static void main(String[] args)
   {
6      myMethod();
7    }
8  }
```

Classes and Objects

- ***Static Variable***
- *Static variables are shared among all instances of a class.*
- *They are declared using the **static** keyword.*
- *They are commonly used for values that are constant across all instances of the class or for maintaining a count of instances.*

```
1  public class ExampleClass {
2      static int staticVariable = 10;
3  }
```

Classes and Objects

- ***Static Methods***
- *a static method is a method that belongs to the class rather than an instance of the class*
- *They can be called using the class name without creating an object.*

```
1 public class ExampleClass {
2     static void staticMethod() {
3         System.out.println("This is a static method.");
4     }
5 }
```

Classes and Objects

- ***Static Blocks***
- *In Java, a static block is a block of code enclosed within curly braces ({}) and preceded by the static keyword.*
- *This block of code is executed when the class is loaded into the Java Virtual Machine (JVM).*
- *Static blocks are used to initialize static variables or perform any other one-time actions that should be taken when the class is first loaded.*

```
1 public class ExampleClass {
2     static {
3         System.out.println("Static block executed.");
4     }
5 }
```

Classes and Objects

- ***Static Class***
- *a static method is a method that belongs to a class rather than an instance of the class*
- *Unlike instance methods, which operate on an instance of the class and have access to the instance's data, static methods are associated with the class itself and do not have access to instance-specific data.*

```
1  public class MyClass {
2      // A static variable (class variable)
3      private static String classVariable = "I am a class variable";
4      // A static method
5      public static void staticMethod() {
6          System.out.println("This is a static method");
7          System.out.println("Class variable value: " +
   classVariable);
8      }
9      public static void main(String[] args) {
10         // Calling the static method directly on the class
11         MyClass.staticMethod();
12         // Accessing the class variable directly
13         System.out.println("Class variable value from main
   method: " + MyClass.classVariable);
14     }
15 }
16
```

Classes and Objects

```
public class Car {
    // Static variable
    static int numberOfCars = 0;
    // Instance variables
    String brand;
    int year;

    // Static block (executed when the class is loaded)
    static {
        System.out.println("Static block executed.");
    }
    // Constructor (increments the static variable)
    public Car() {
        numberOfCars++;
    }
    // Instance method
    void startEngine() {
        System.out.println("Engine started!");
    }
    // Static method
    static void displayNumberOfCars() {
        System.out.println("Number of cars: " + numberOfCars);
    }
    public static void main(String[] args) {
        // Creating objects of the Car class
        Car myCar = new Car();
        myCar.brand = "Toyota";
        myCar.year = 2022;
        myCar.startEngine();
        Car anotherCar = new Car();
        anotherCar.brand = "Honda";
        anotherCar.year = 2021;
        anotherCar.startEngine();
        // Calling static method
        Car.displayNumberOfCars();
    }
}
```

Assignment