Object Oriented Programming - OOPs

By Watcharin Sarachai

Topic

- Introduction
- Classes and Objects
- Designing classes

Introduction

Why Software Development?

- Software development brings a business to new **heights of integration**.

- It allows company to be accessible from almost anywhere via **smartphone** or **computer**.

- **Software development** improves the performance of sales and service.

https://www.it-thailand.com/software/operations.htm, https://www.pithhub.com/business-software/

How Many Lines of Code Does It Take In Software?

- The control software to run a U.S. military drone uses 3.5 million lines of code.
- A Boeing 787 has 6.5 million lines.
- Google Chrome (browser) runs on 6.7 million lines of code.
- The Android operating system runs on 12-15 million lines.
- Not including backend code, Facebook runs on 62 million lines of code.
- All Google services combine for a whopping 2 billion lines.
    - Applying the math above – that means it would take 36,000,000 pages to "print out" all of the code behind all Google services. That would be a stack of paper 2.2 mi (3.6 km) high!
- https://www.visualcapitalist.com/millions-lines-of-code

Divide and Conquer

- Most useful computer programs are much longer than our five line "**Hello World**" program.
- Many programs are **HUGE** – *literally tens of millions of lines of code*.
- No human can possibly understand all 10 million lines of a program all at once.
- **In fact**, most programmers will admit that they can only focus on at most a **couple hundred lines of code** at a time, and experienced programmers prefer chunks of **no more than 50 lines** or so *(about what you can see on a screen at a one time)*.
- To be able to create large programs, we have to be able to break the job down into small, manageable parts.

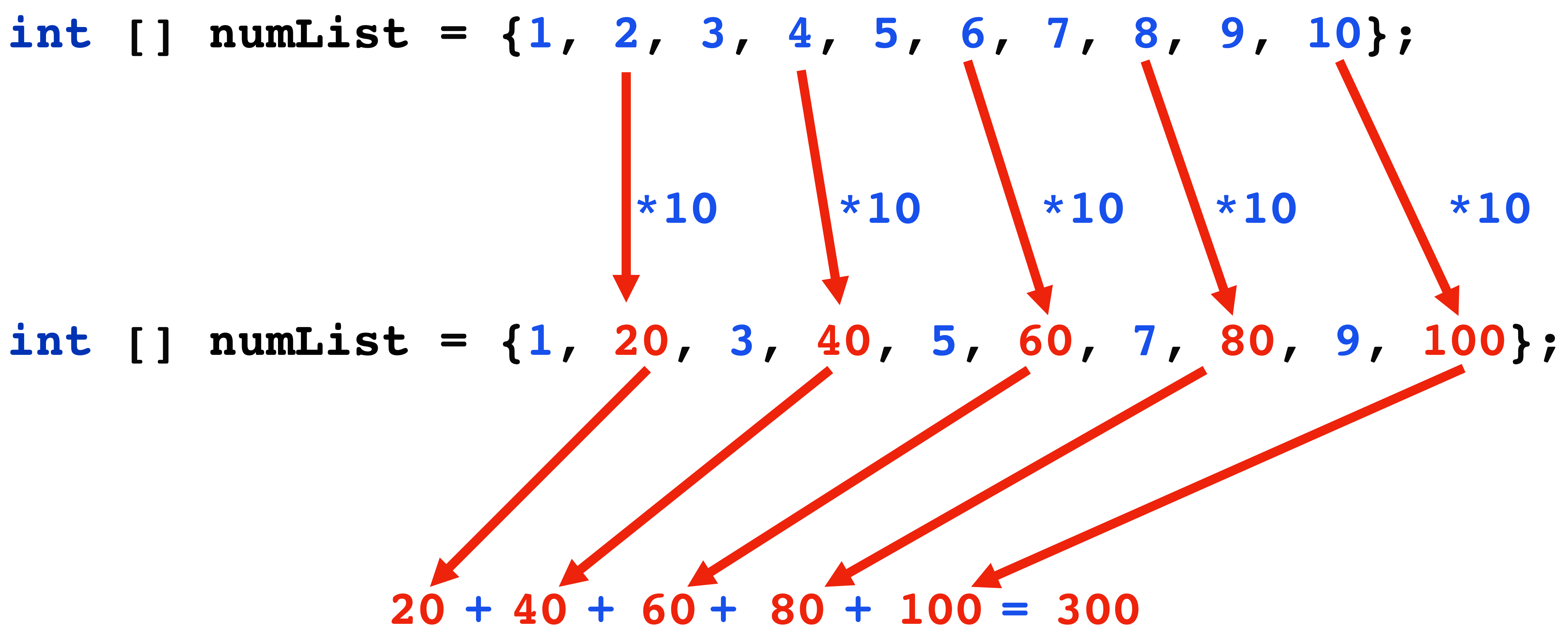
Procedural Decomposition

- To be able to solve the problem by creating a program, we have to break the job down into small, manageable parts.
- The process of breaking a task down into smaller parts is called "**procedural decomposition**", and almost programming languages provide a way to do this.
- Java does it using what it calls "**methods**".
- A simple **Hello World program** had a single method called **main**, but most Java programs have several methods.

Programming Paradigm

- What is a programming paradigm?
  - It is a style of programming, a way of thinking about **software construction**.
  - A **programming paradigm** does not refer to a specific language but rather to a way to program, a methodology.
    - Imperative programming
    - Functional programming
    - Object oriented programming
    - etc.

Solutions



```
int [] numList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int [] numList = {1, 20, 3, 40, 5, 60, 7, 80, 9, 100};
```

20 + 40 + 60 + 80 + 100 = 300

Programming Paradigm

- **Imperative programming** is a programming paradigm that uses **statements** that change a program's state.

- Imperative programming focuses on describing how a program operates. — Wikipedia

```java
package camt;
public class Imperative {
    public void m1() {
        int [] numList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int result = 0;
        for (int i=0; i<numList.length; i++) {
            if (numList[i] % 2 == 0) {
                result += numList[i] * 10;
            }
        }
        System.out.printf("Total is %d\n", result);
    }
}
```

Programming Paradigm

- **Functional programming (FP)** is a programming paradigm — a style of building the structure and elements of computer programs — that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. —Wikipedia

```java
package camt;
import java.util.Arrays;
public class Functional {
    public void m1() {
        int [] numList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int result = Arrays.stream(numList)
                .filter(x -> x % 2 == 0)
                .map(x -> x * 10)
                .sum();
        System.out.printf("Total is %d\n", result);
    }
}
```

```java
package camt;
public class Imperative {
    public void m1() {
        int [] numList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int result = 0;
        for (int i=0; i<numList.length; i++) {
            if (numList[i] % 2 == 0) {
                result += numList[i] * 10;
            }
        }
        System.out.printf("Total is %d\n", result);
    }
}
```

Imperative programming

```java
package camt;
import java.util.Arrays;
public class Functional {
    public void m1() {
        int [] numList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int result = Arrays.stream(numList)
                .filter(x -> x % 2 == 0)
                .map(x -> x * 10)
                .sum();
        System.out.printf("Total is %d\n", result);
    }
}
```

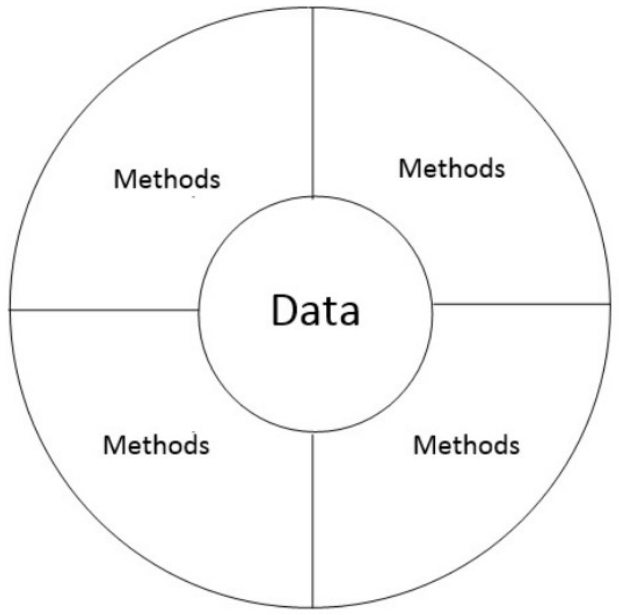
Functional programming

Output:

```
Total is 300
Total is 300
```

What is OOP ?

- OOP stands for Object-Oriented Programming.
- A programming paradigm that focus on **object** and **data**.
- Functional programming treats computation as the evaluation of **mathematical functions** that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.



https://www.w3schools.in/java-tutorial/object-oriented-programming-oops/

What is OOP ?

- Object-Oriented Programming has several advantages over other programming paradigms:
  - OOP is faster and easier to execute.
  - OOP provides a clear structure for the programs.
  - OOP helps to keep the Java code **DRY** "***D**on't **R**epeat **Y**ourself*", and makes the code easier to maintain, modify and debug.
  - OOP makes it possible to create full reusable applications with less code and shorter development time.
  - Possible to map objects in a problem domain within a program.
  - Use of **inheritance** can eliminate redundant codes in a program.

Tip: The "Don't Repeat Yourself" (DRY) principle is about reducing the repetition of code. You should extract out the codes that are common for the application, and place them at a single place and reuse them instead of repeating it.

https://www.w3schools.com/java/java_oop.asp

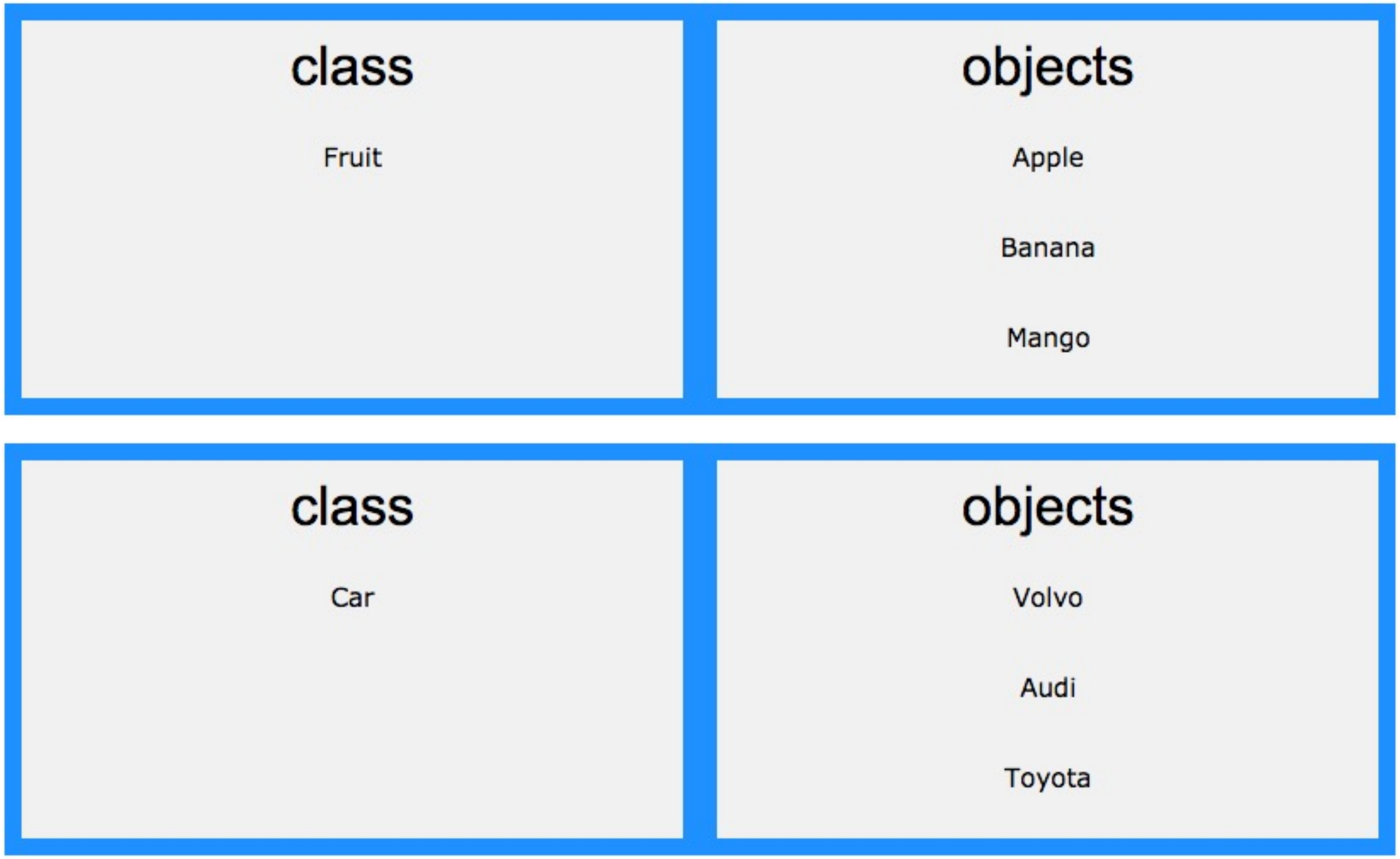
Basic Term and Feature That are used and Provided by OOP

- Classes and Objects
- Data Abstraction
- Data encapsulation
- Inheritance
- Polymorphism

Question ?

Classes and Objects

What are Classes and Objects?



https://www.w3schools.com/java/java_oop.asp

What are Classes and Objects?

- Each **object** comes in different forms and shapes, but we can **classify** different versions of the same **object** into a **category** or **group.**

- It's why we can go to a furniture store and recognize different items as "**chairs**" even if they look very different from one another.

What are Classes and Objects?

- We recognize all of these different objects as being part of the same **group** or **type**.

What are Classes and Objects?

- For example, there are different kinds of books out there, but they all tend to have a **title**, an **author**, **a cover**, **pages**, etc.

- In other words, individual books all have similar **attributes** that allow you to classify them in your mind as part of the **category** “**book**".

What are Classes and Objects?

- A **book** acts as a kind of blueprint for that **object.** In programming, it's called a class.
- We create a class name by group lots of details together, called a named type.

What are Classes and Objects?

- A **class** is a template for **objects**, and an **object** is an **instance** of a **class**.
- When the individual objects are created, they **inherit** all the **variables** and **methods** from the **class**.
- 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.

Java Class

HelloWorld.java

```
public class HelloWorld {

    public static void main(String[] args) {
       System.out.println("Hello World!!”);
    }

}
```

Output: → `Hello World!!`

Java prints text out to the console using a System.out.println statement. Here’s the one you used in your **HelloWorld** program:

```
System.out.println("Hello World!!”);
```

- System essentially refers to the computer as a whole
- out refers to the standard output device: the console
- println indicates that you want to “print” as a complete line (ln) of text

Java Class

HelloWorld.java

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!!”);
    }

}
```

The **main method** is the wrapper that tells your computer which lines of code to run.

In Java it has a very specific series of keywords:

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

Question ?

Designing classes

Designing classes

- To see how to design a class, let's continue with the book example.
- Below, we've identified a sample of **information** that could **describe** any given book:
  - title
  - author
  - number of pages
  - publisher

Create a Class

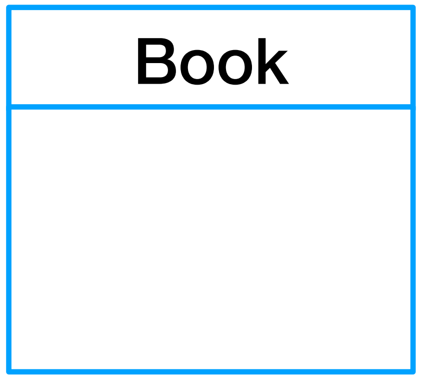
- To declare a class in Java, use a keyword class followed by a custom name.

Book.java

Create a class named "Book":

```
class Book {
// functionality of the class
}
```

Class



```
A class should always start with an uppercase first letter, and that
the name of the java file should match the class name.
```

Create a Class

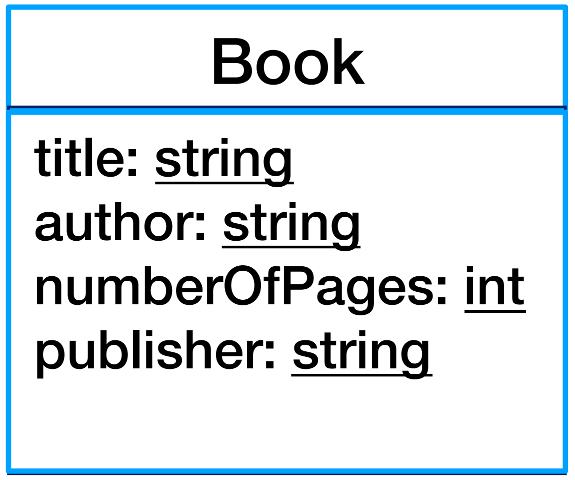
- Now let's add fields as defined:

Book.java

Create a class named "Book":

```java
public class Book {
   String title;
   String author;
   int numberOfPages;
   String publisher;
}
```

Class

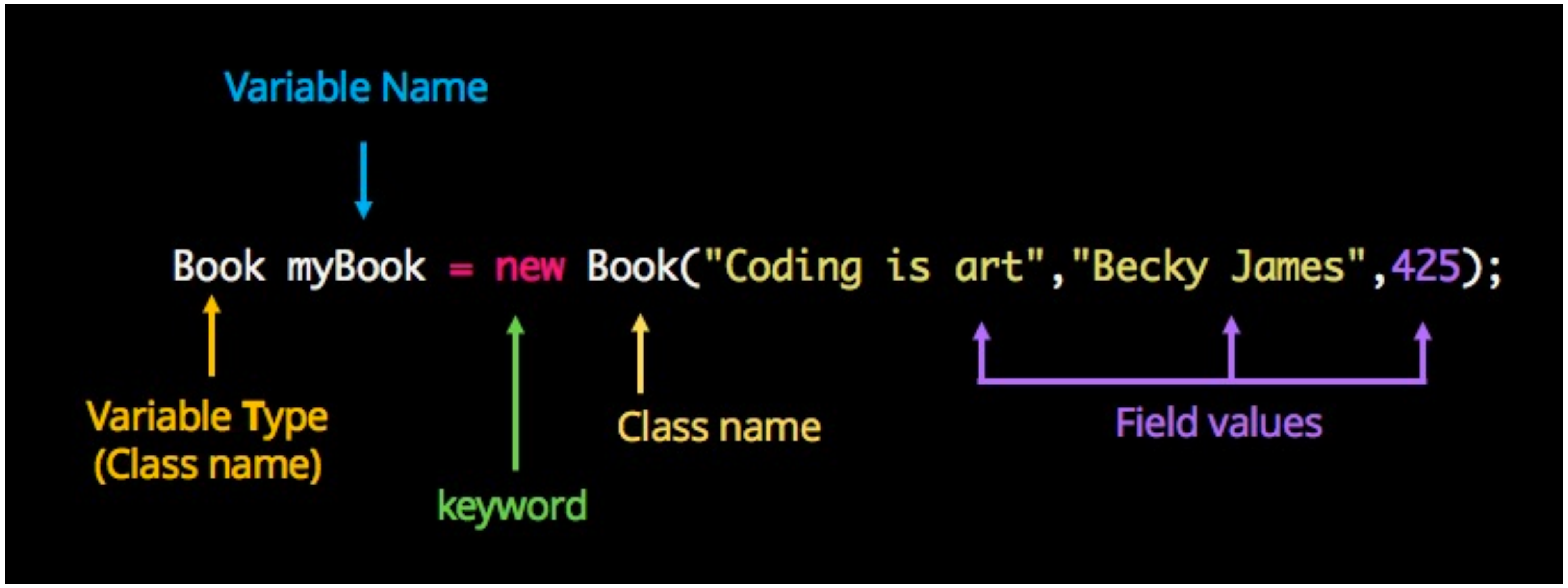


Utilizing classes

- How do we find a book from an online store?
  - when you're searching, you don't just type "book," right?
  - You need a specific instance of a book, say, **Alice in Wonderland**.
- To work with a class, we need to create a **concrete object** of that class.
- We need a specific object, like a particular book (*Alice in Wonderland*).
- The specific book is called an **instance** of a class!

Create an Object

- The example code for creating a book with values provided:



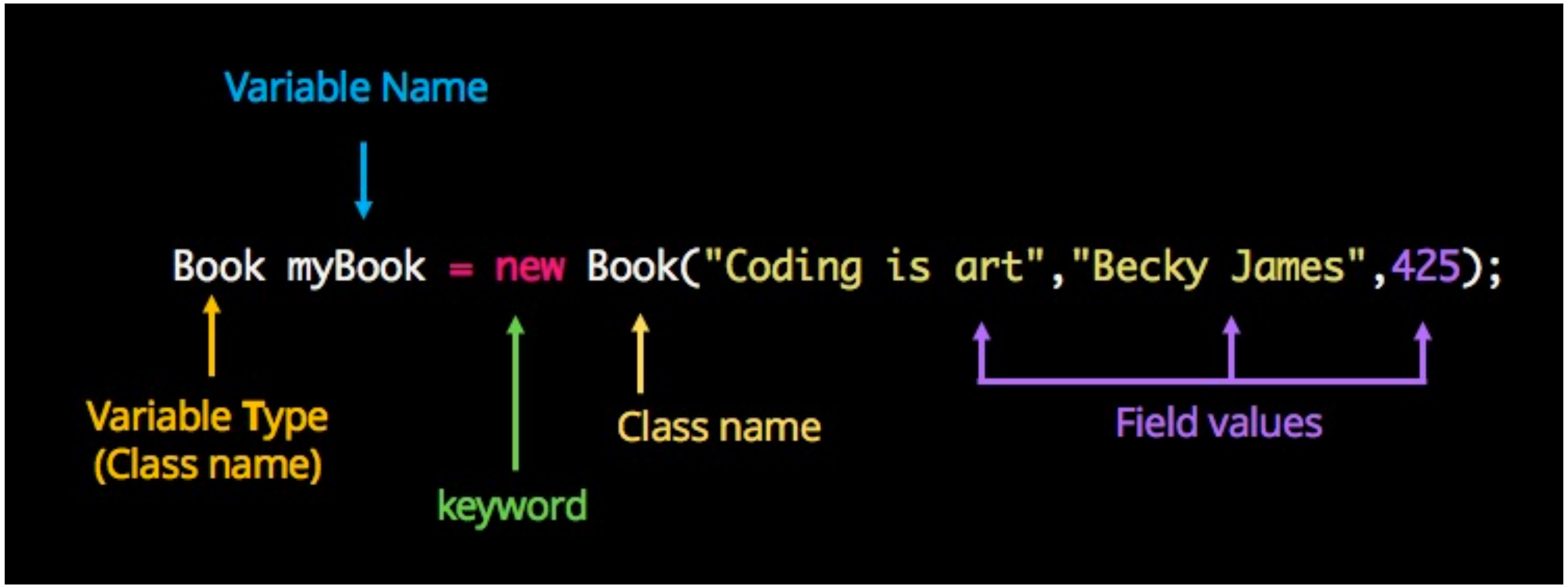
```
Book myBook = new Book("Coding is art","Becky James",425);
```

First, do a regular declaration of a variable, with its name myBook and its type Book.

https://openclassrooms.com/en/courses/5667431-learn-programming-with-java/5906497-define-objects-and-their-attributes-with-classes

Create an Object

- The example code for creating a book with values provided:



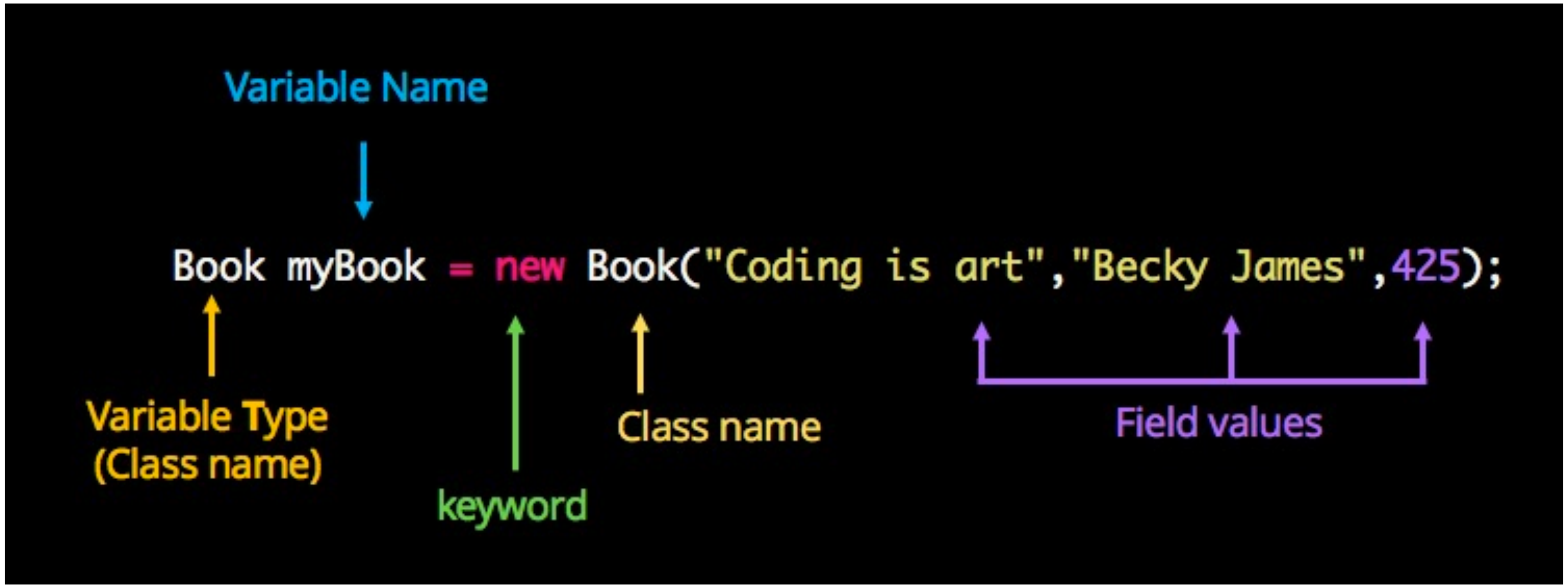
```
Book myBook = new Book("Coding is art","Becky James",425);
```

A class is a named complex type, instead of int ,double or string, our type is the class.

https://openclassrooms.com/en/courses/5667431-learn-programming-with-java/5906497-define-objects-and-their-attributes-with-classes

Create an Object

- The example code for creating a book with values provided:



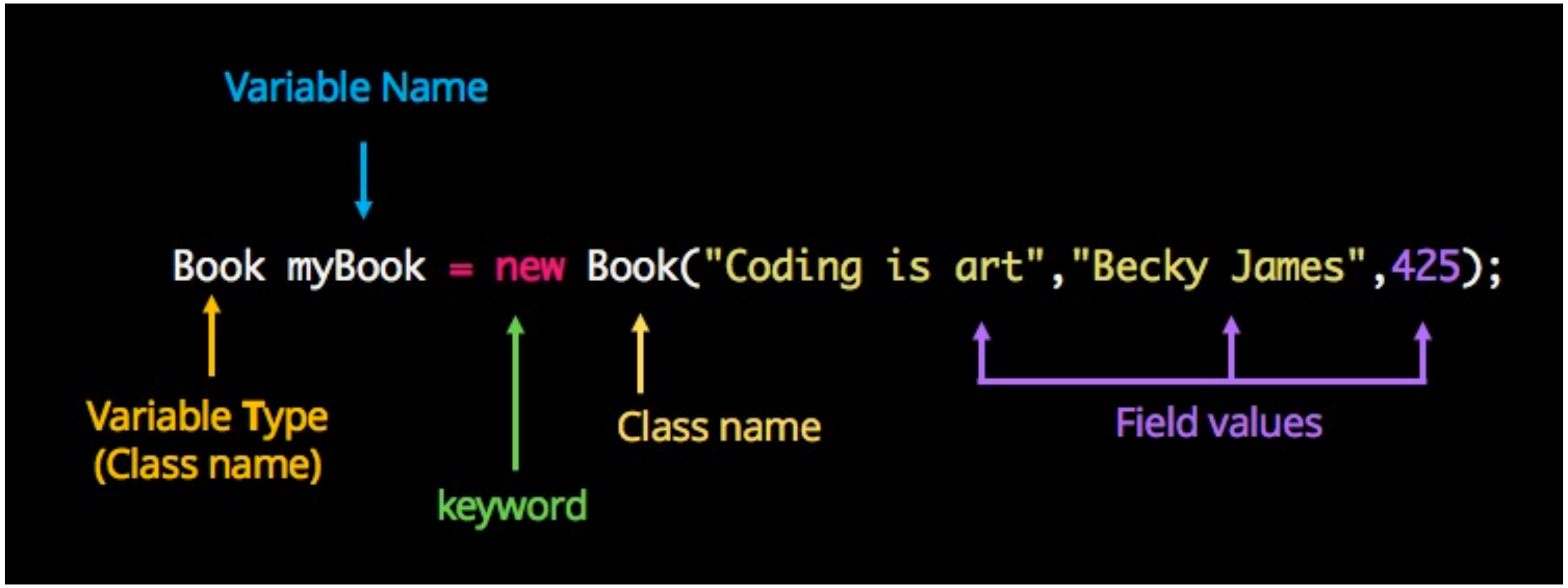
```
Book myBook = new Book("Coding is art","Becky James",425);
```

The variable myBook is initialized with the object creation expression:
new Book("Coding is art","Becky James",425);

This expression is composed of the **new** keyword, followed by the name of the class again (**Book**), and some parentheses **()** with values inside.

Create an Object

- The example code for creating a book with values provided:



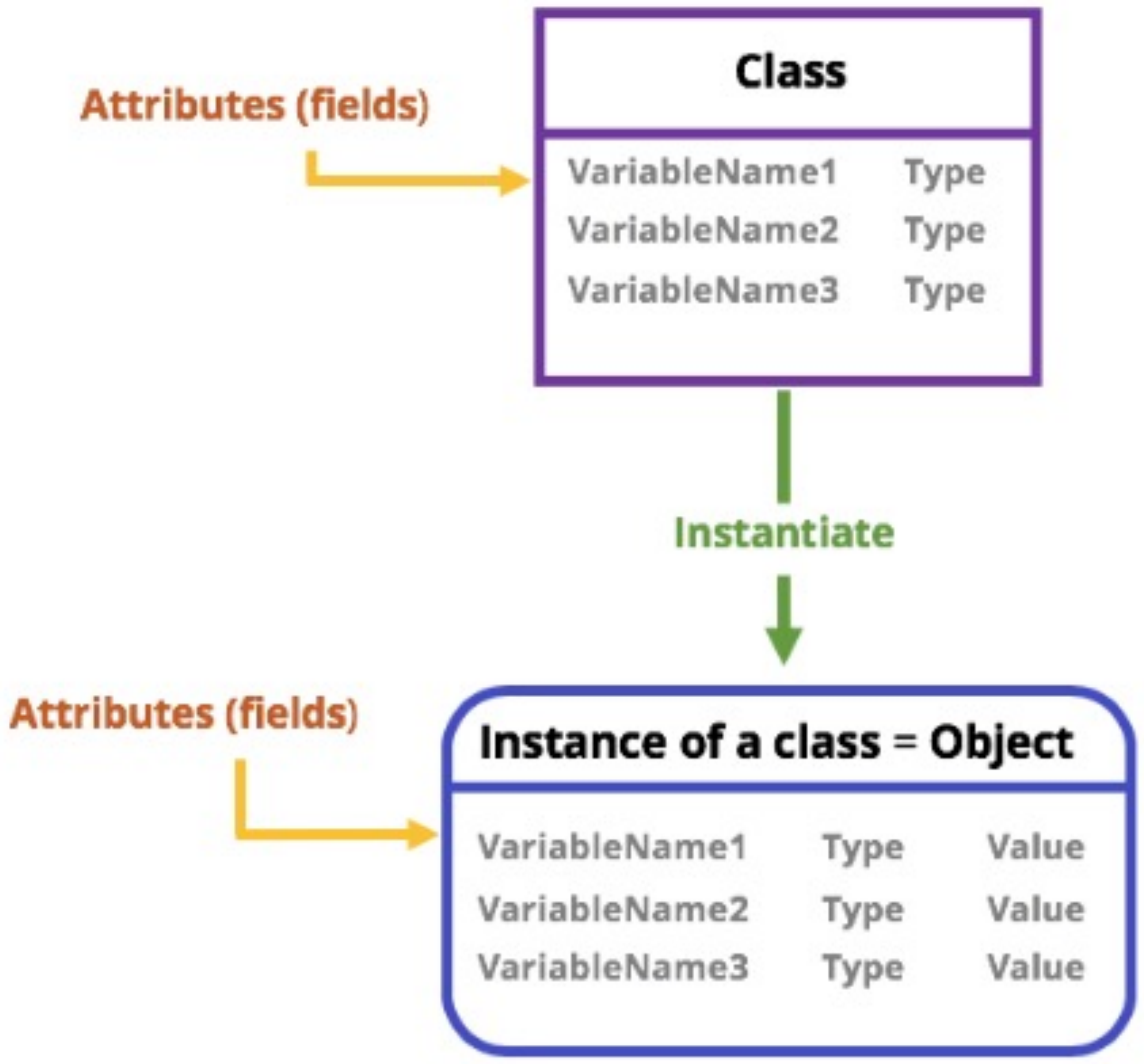
```java
Book myBook = new Book("Coding is art","Becky James",425);
```

The parenthesis contain a specified value for each of the original fields:

title, author, and numberOfPages, separated each one with a comma.

https://openclassrooms.com/en/courses/5667431-learn-programming-with-java/5906497-define-objects-and-their-attributes-with-classes

Working with attributes (fields)



Attribute (Which are called field in java) are the **variable** you **define** when crating a class

Instantiate an object you declare a variable of that class

When you instantiate an object from the class, you also set the **value** for each field in the object

A common way to access fields

- A common way to access fields in many programming languages is using what's called a **dot notation.**
- It means you need to write the name of an instance or object followed by an attribute name of interest, separated by a dot:

```
myBook.title = "Coding is Art";
myBook.author = "Becky James";
myBook.numberOfPages = myBook.numberOfPages + 10;
```

Try It Yourself #1

- Open the file in folder: camt/day1/ex01
  - Book.java
  - Lesson.java
- Edit file name **Lesson.java** and follow the instructions:
  1. Under the first //TODO statement, create a variable named myBook and initialize it with an instance of the Book class.
  2. Under the second //TODO statement, assign values to the title, author and numberOfPages fields of your myBook object using the dot operator.

Try It Yourself #1

1. Under the first //TODO statement, create a variable named myBook and initialize it with an instance of the Book class.

```
Book myBook = new Book();
```

2. Under the second //TODO statement, assign values to the title, author and numberOfPages fields of your myBook object using the dot operator.

```
myBook.title = "Going Down Home with Daddy";
myBook.author = "Starling Lyons, Daniel Minter";
myBook.numberOfPages = 400;
```

Book.java

```java
class Book {
  String title;
  String author;
  int numberOfPages;
}
```

Book.java

```java
class Lesson {
  public static void main(String [] args) {
    //TODO Step 1 - instantiate an object of class Book and assign it to a
variable named myBook

    //TODO Step 2 - assign a value to the title, author and numberOfPages fields
of your object.

    //Print the values
    System.out.println("The title of the book is " + myBook.title + "\nIts author
is" + myBook.author + "\nIt contains " + myBook.numberOfPages);
  }
}
```

Try It Yourself #2

ava to create the book1 and book2 with the following information and print th

book1

- title: Milkman: A Novel
- author: Anna Burns
- numberOfPages: 200

book2

- title: The Undefeated
- author: Kwame Alexander, Kadir Nelson
- numberOfPages: 300

```
Output:


The title of the book is Milkman: A Novel
Its author isAnna Burns

It contains 200

The title of the book is The Undefeated

Its author isKwame Alexander, Kadir Nelson

It contains 300
```

Multiple Book Instance

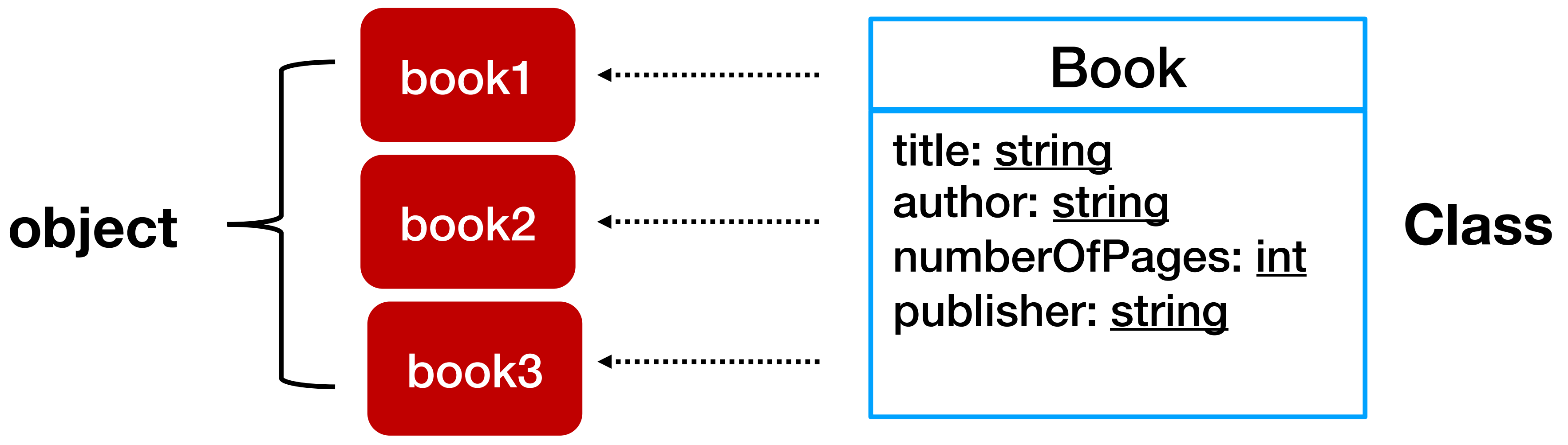
- We can create multiple objects of one class:

Example

Create three objects of "Book":

```
public class Lesson {

    public static void main(String[] args) {
        MyClass book1 = new Book();   // Object 1
        MyClass book2 = new Book();   // Object 2
        MyClass book3 = new Book();   // Object 3
    }
}
```



One file per a public class

Book.java

```java
public class Book {
   String title;
   String author;
   int numberOfPages;
}
```

Filename: Lesson.java

```java
public class Lesson {

    public static void main(String[] args) {
        MyClass book1 = new Book();  // Object 1
        MyClass book2 = new Book();  // Object 2
        MyClass book3 = new Book();  // Object 3
    }
}
```

Try It Yourself #3

MyClass

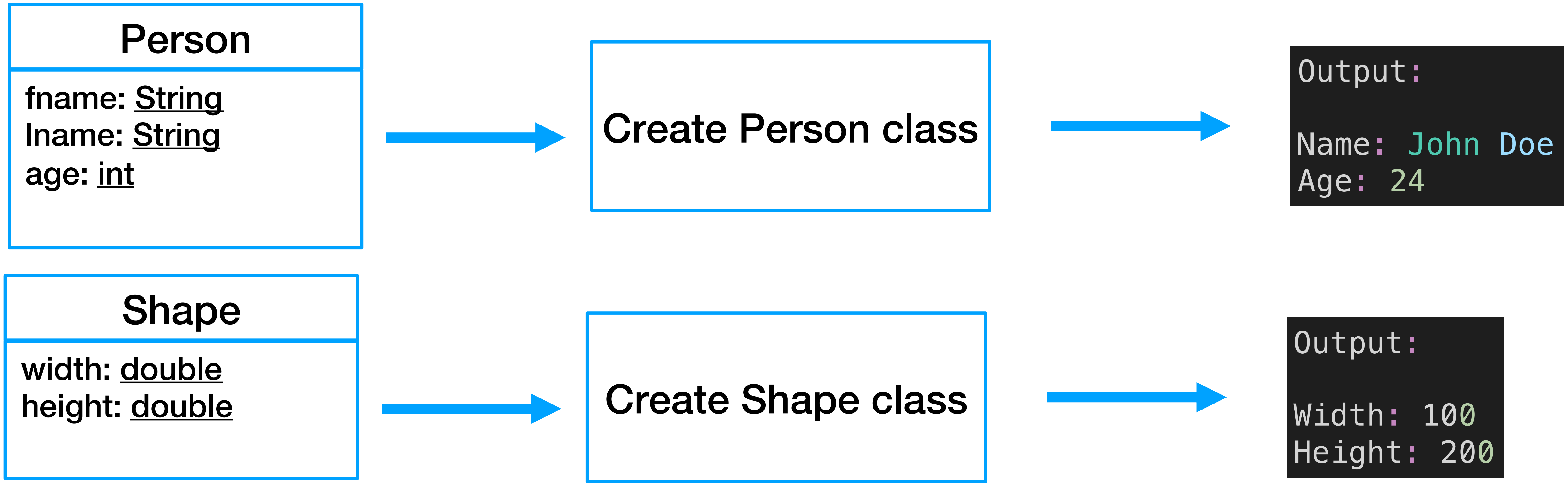
x: int

```
class MyClass {
  int x = 5;

  public static void main(String [] args) {
    MyClass myObj = new MyClass();
    System.out.println(myObj.x);
  }
}
```

```
Output:

5
```



```
Output:

Name: John Doe
Age: 24
```

```
Output:

Width: 100
Height: 200
```

Summary

- A **class** is a blueprint of an object.
- A variable of a class is called an **instance of a class** or an **object.**
- A class allows you to **create complex types** by grouping its attributes by defining **fields.**
- To create an **object,** you need to declare a variable of a class and instantiate it.
- The **dot notation** provides access to **fields.**

Question ?

Reference

- **Java Tutorial https://www.w3schools.com/java/ Accessed: 2020-07-04**
- **Static Keyword in Java | Static Block, Variable, Method & Class | Edureka** https://www.edureka.co/blog/static-keyword-in-java/ Accessed: 2020-07-04
- Learn programming with Java – OpenClassrooms https://openclassrooms.com/en/courses/5667431-learn-programming-with-java Accessed: 2020-07-04