Christoph Baitis

Java Course

13. Oktober 2022

First: Which Language?

Anyone in here who needs us to speak English?

About me

Christoph Baitis

- Christoph.baitis@tu-dresden,de
- GitHub: ein-christoph

What are we doing here?

- Introduction to programming
- Getting to know the basics of Java
- Preparation for upcoming courses
  (e.g 'Softwaretechnologie', 2nd Semester)

- Slides and material:
  https://ein-christoph.github.io/java-tud
- Thanks to
  Florian Kluge, Moritz Schulz
  (https://trivo25.github.io/tud-java-course)

Structure

- 15 lessons
- Thursday, 14:50 - 16:20
- APB/E040/E (right here)
- Attendance list

Attendance

- This course is held on a voluntary basis.
- You’re here voluntarily.
- If you want to quit, please let me know so we can invite students from the waiting list.
- If you don’t attend the course for 2 weeks in a row without notice I will give your slot to other students.

Course philosophy

- This course is centered around you.
- Coding is best learned by doing it.
- Illustrative examples help.
- Mistakes are good because they help us learn.
- I’m not flawless expert either.
- Please ask questions
- because in the end, it’s about your understanding.
- I’ll walk through the class room to check that everyone gets along.
- Ask each other or ask me.

Why Java?

Why Java?

- widely used & modern programming language
- helpful ways of structuring code
- can be used for lots of things
- the same program can run on most computers
- good for getting started

Why Java?

- widely used & modern programming language
- helpful ways of structuring code
- can be used for lots of things
- the same program can run on most computers
- good for getting started

- Android development
- Web applications
- Desktop GUI applications
- … and much more

Who are you?

- Do you have any programming experience already?

  https://strawpoll.com/polls/jVyGJAoVYZ7

We're about to get started…

- we need Java OpenJDK 11
  - https://adoptium.net
- check if it's installed properly:
  - open a terminal
    - Windows: Windows+R => cmd => Enter
    - MacBook: ⌘ + T
    - Linux (depends): Ctrl+Shift+T
- enter: `javac -version`
- it should say: `javac 11.0.12`

We’re about to get started…

- we need Java OpenJDK 11
  - https://adoptium.net
- check if it’s installed properly:
  - open a terminal
    - Windows: Windows+R => cmd => Enter
    - MacBook: ⌘ + T
    - Linux (depends): Ctrl+Shift+T
- enter: `javac -version`
- it should say: `javac 11.0.12`

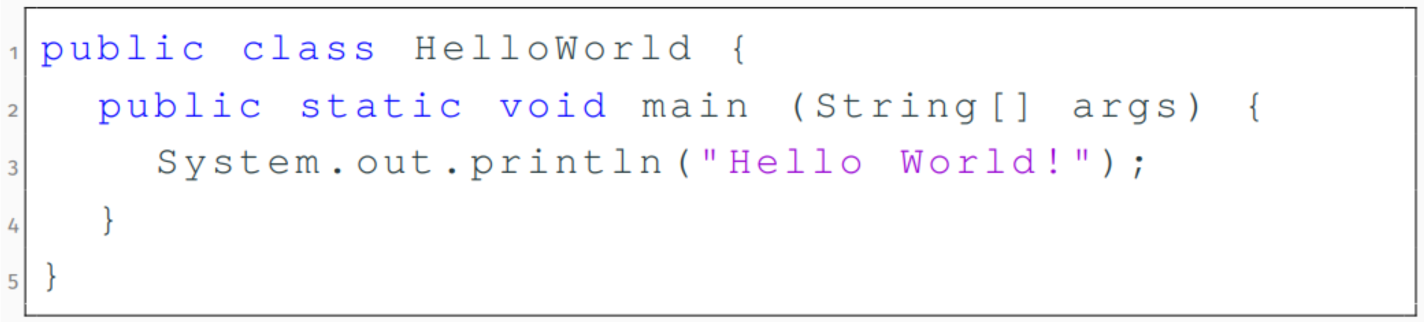
Doesn’t work?
Use an online compiler for now.

https://www.jdoodle.com/online-java-compiler/

Your first piece of code

- Create a new folder
  - Open the terminal and navigate into that folder using
    - `$ cd /to/my/folder`
    - Create a new file by either typing
    - `$ touch helloWorld . java`
  - Or right-clicking in your folder
    - Right click -> new -> text document
    - and save it as a `. java` file

Your first piece of code



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

../code_samples/HelloWorld.java

Run the program

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

../code_samples/HelloWorld.java

- save the file: File > Save
- For VS Code users:
  - open the terminal: View > Terminal
  - type: `javac HelloWorld.java`
  - type: `java HelloWorld`
  - see: `Hello World!`

Let’s play around a bit

- change the text
- try to run the program
    - … (like we did before)

Let’s explain... (1/4)

- Coding (= Programmieren) is telling the computer what to do.

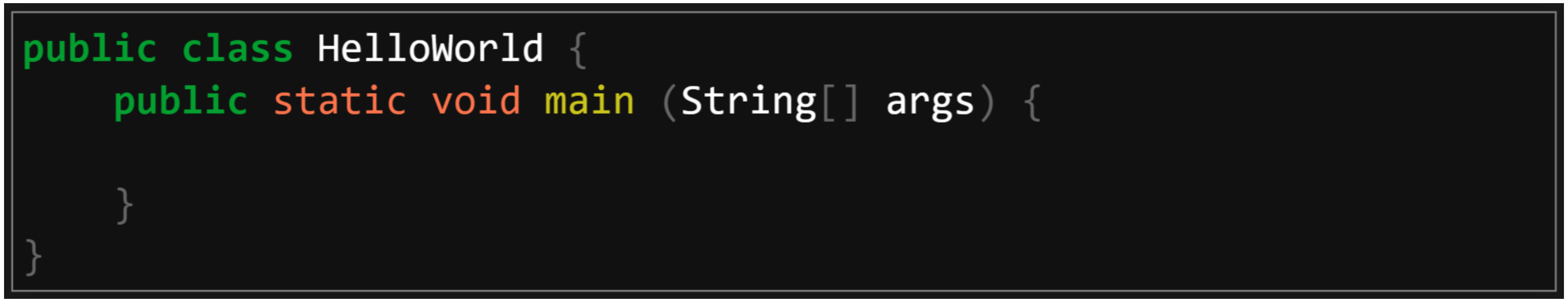
1. Coding = We list instructions for the computer.
   - precise
   - step by step
2. A program called compiler translates code so the computer can understand it.
3. The computer runs the program.

Let’s explain… (2/4)

1. We write code that humans can read.
   - HelloWorld.java
   - let’s look at the code again (next slide)
2. The compiler javac translates the code so the computer understands it.
   - HelloWorld.java => HelloWorld.class
3. The computer runs the program.
   - command: java HelloWorld

Let’s explain… (3/4)

- This is the framework of every Java program:
- HelloWorld is the class name and should be like the file name, but without .java
- start inside public static void main (String[] args) { … }



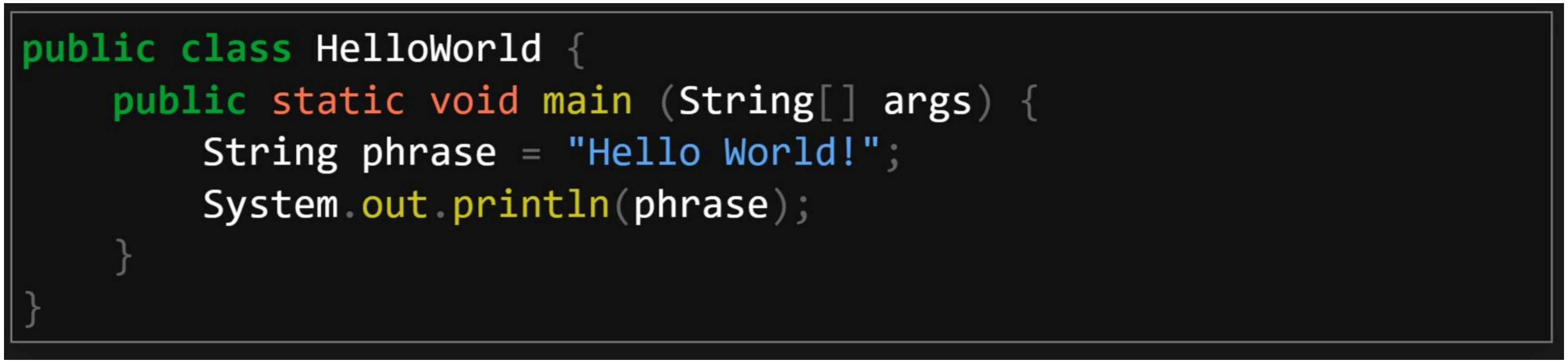
```java
public class HelloWorld {
    public static void main (String[] args) {

    }
}
```

Let’s explain... (4/4)

- This is the piece of code
- that prints `Hello World!`

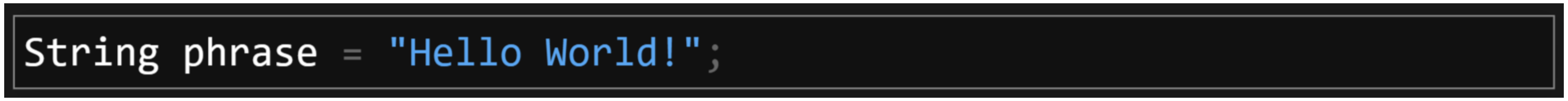
```java
System.out.println("Hello World!");
```

Introducing: Variables



```
public class HelloWorld {
    public static void main (String[] args) {
        String phrase = "Hello World!";
        System.out.println(phrase);
    }
}
```

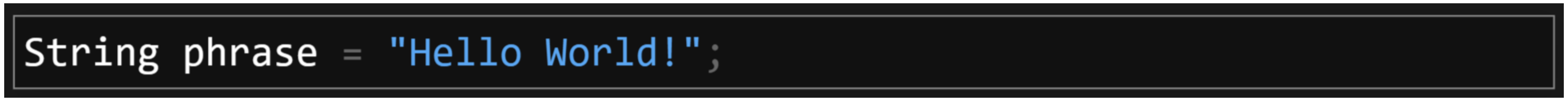
About Variables



```java
String phrase = "Hello World!";
```

- they have a type: this one is a String (basically a piece of text)
- they have a name: this one is called phrase
- they can be created (formally: declared): =
- they have a value: "Hello World!"
- note the "": they tell Java that this is text, not code
- think of them like a box that can only store things of a specific type

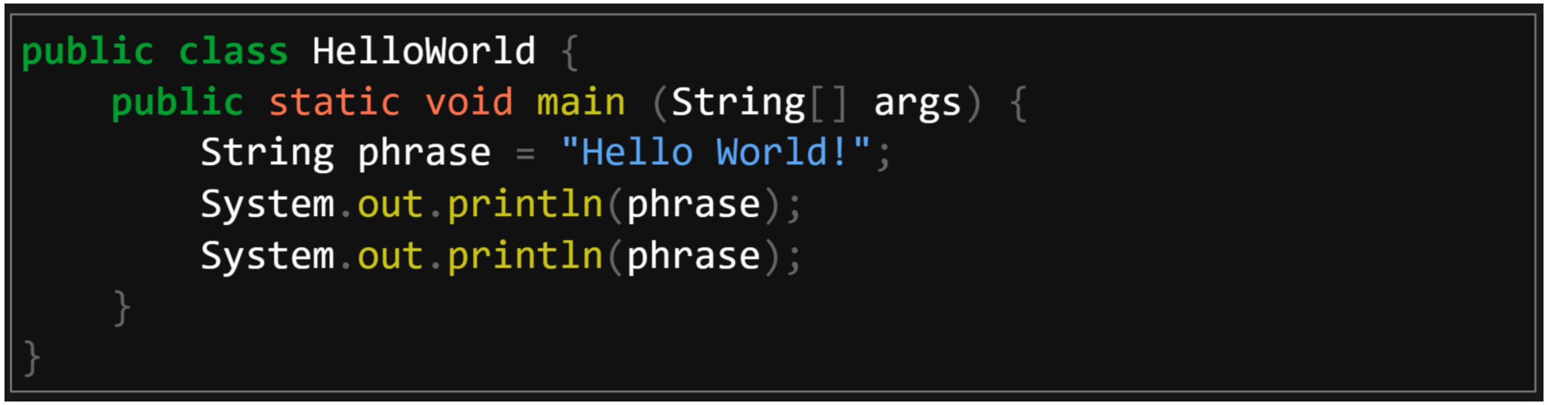
About Variables



```
String phrase = "Hello World!";
```

- they have a type: this one is a String (basically a piece of text)
- they have a name: this one is called phrase
- they can be created (formally: declared): =
- they have a value: "Hello World!"
- note the "": they tell Java that this is text, not code
- think of them like a box that can only store things of a specific type

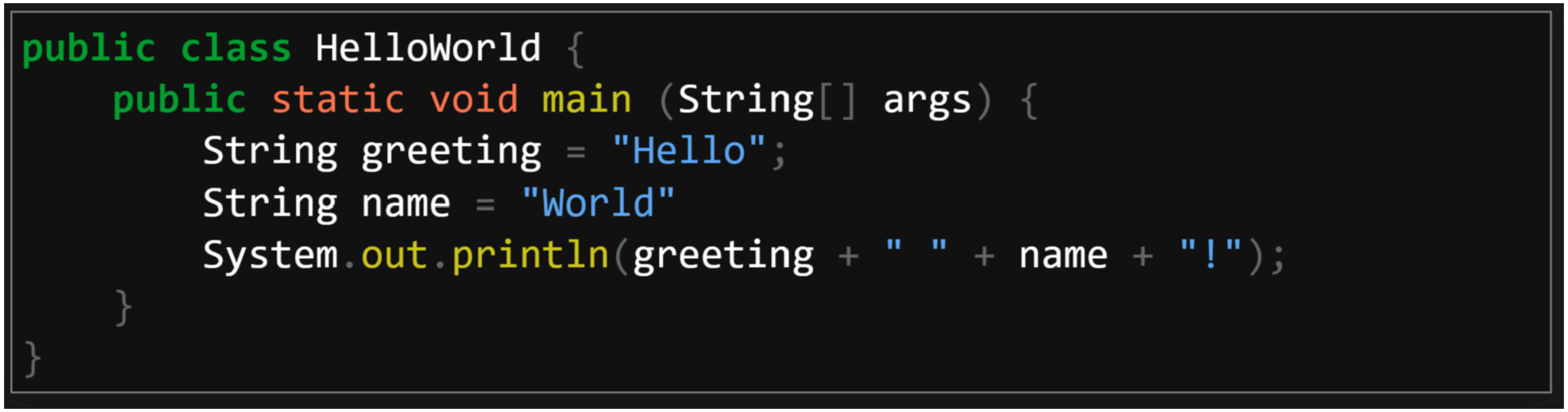
About Variables



```
public class HelloWorld {
    public static void main (String[] args) {
        String phrase = "Hello World!";
        System.out.println(phrase);
        System.out.println(phrase);
    }
}
```

- we can store data in them
- we can re-use them
- avoid typing their values twice

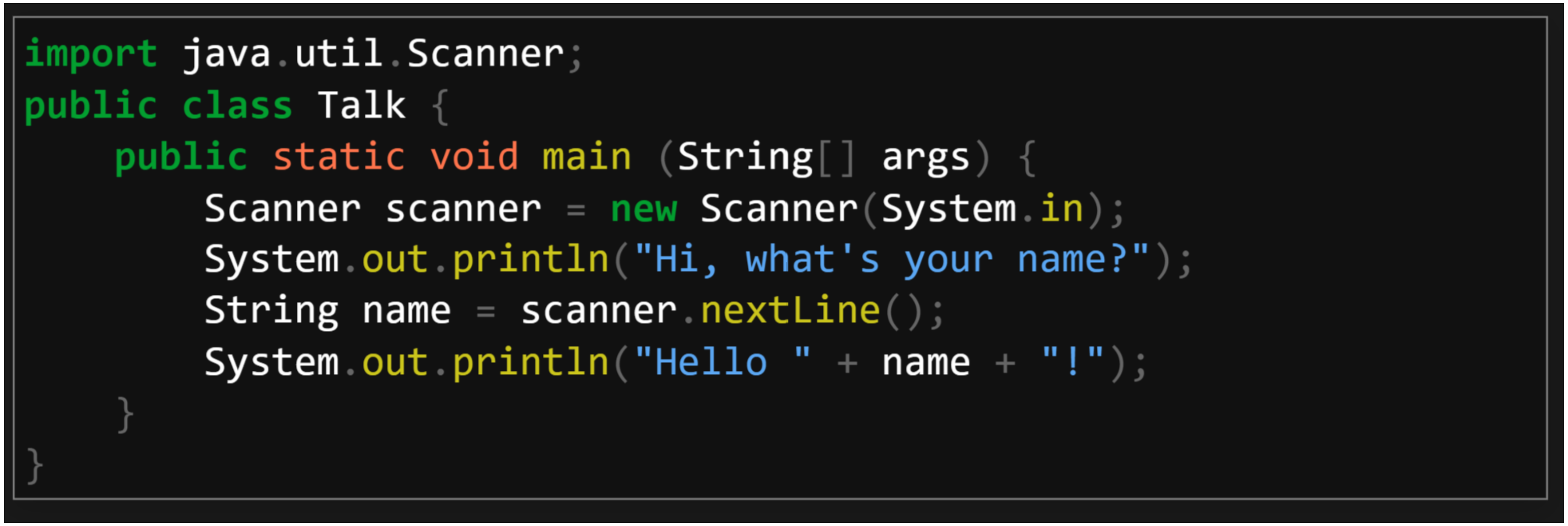
About Variables



```java
public class HelloWorld {
    public static void main (String[] args) {
        String greeting = "Hello";
        String name = "World"
        System.out.println(greeting + " " + name + "!");
    }
}
```

- Strings can be merged (concatenated)
- prints: Hello World! (just as before)

Let’s talk to our program!

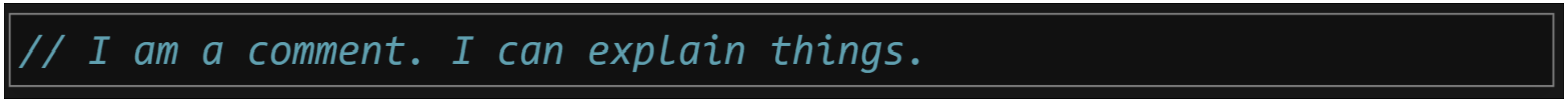


```java
import java.util.Scanner;
public class Talk {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Hi, what's your name?");
        String name = scanner.nextLine();
        System.out.println("Hello " + name + "!");
    }
}
```

```java
import java.util.Scanner;
public class Talk {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Hi, what's your name?");
        String name = scanner.nextLine();
        System.out.println("Hello " + name + "!");
    }
}
```

File: `Talk.java`

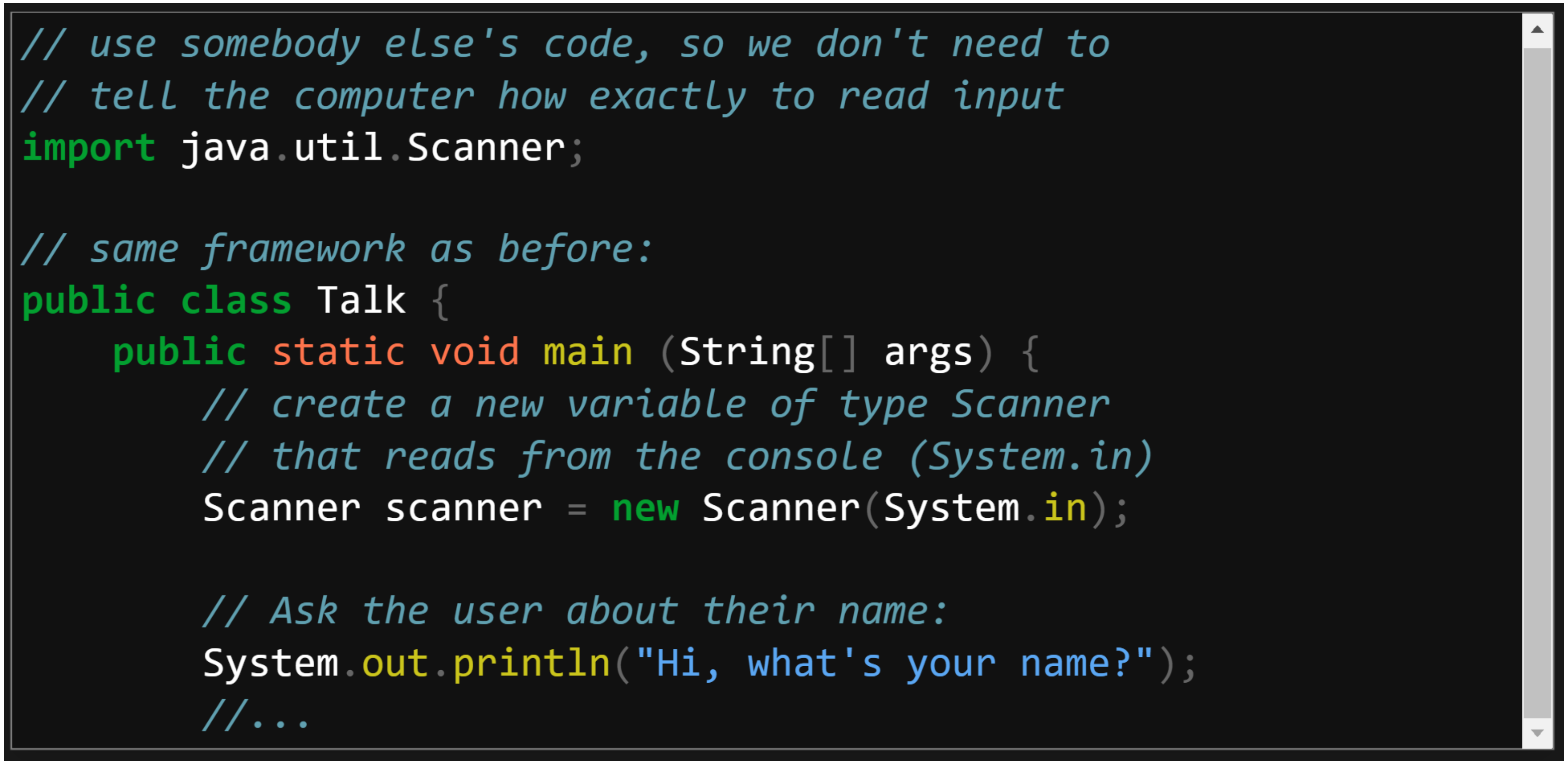
Always comment your code!



```
// I am a comment. I can explain things.
```

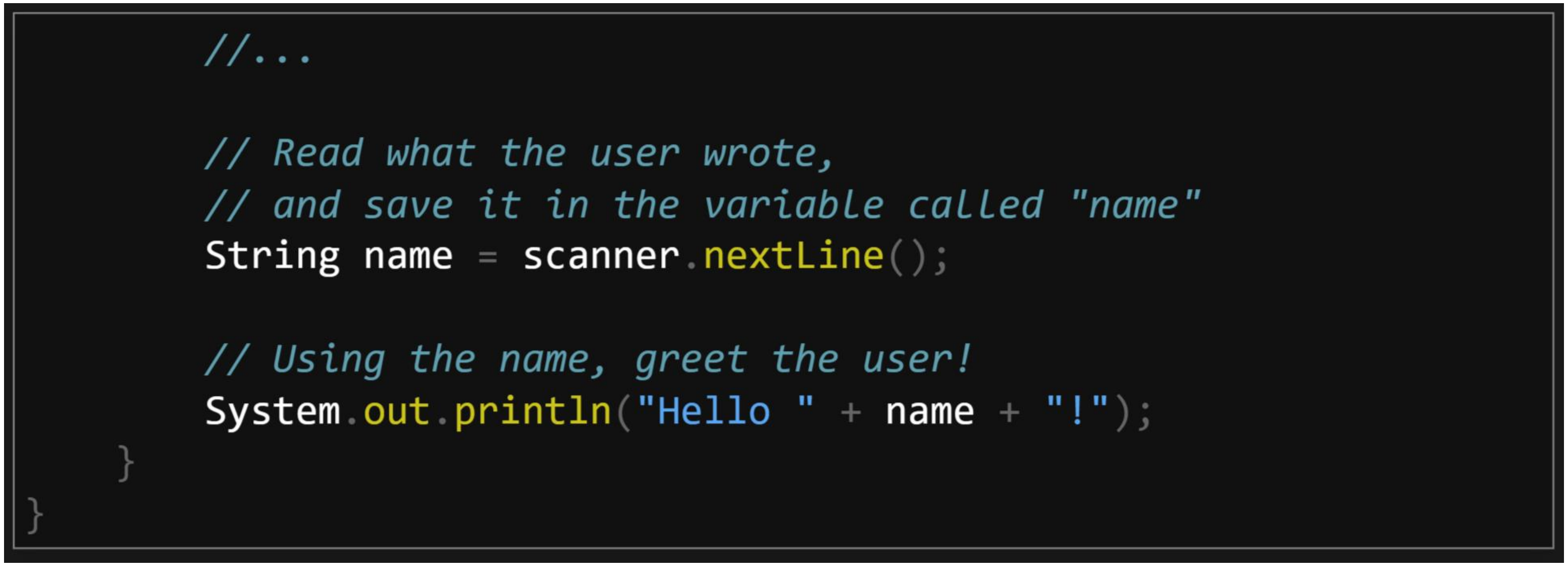
- comments are ignored by Java
- we can use them to explain our code (to ourselves)
- next, I'll use comments to explain the previous code

Let’s explain #2… (1/2)



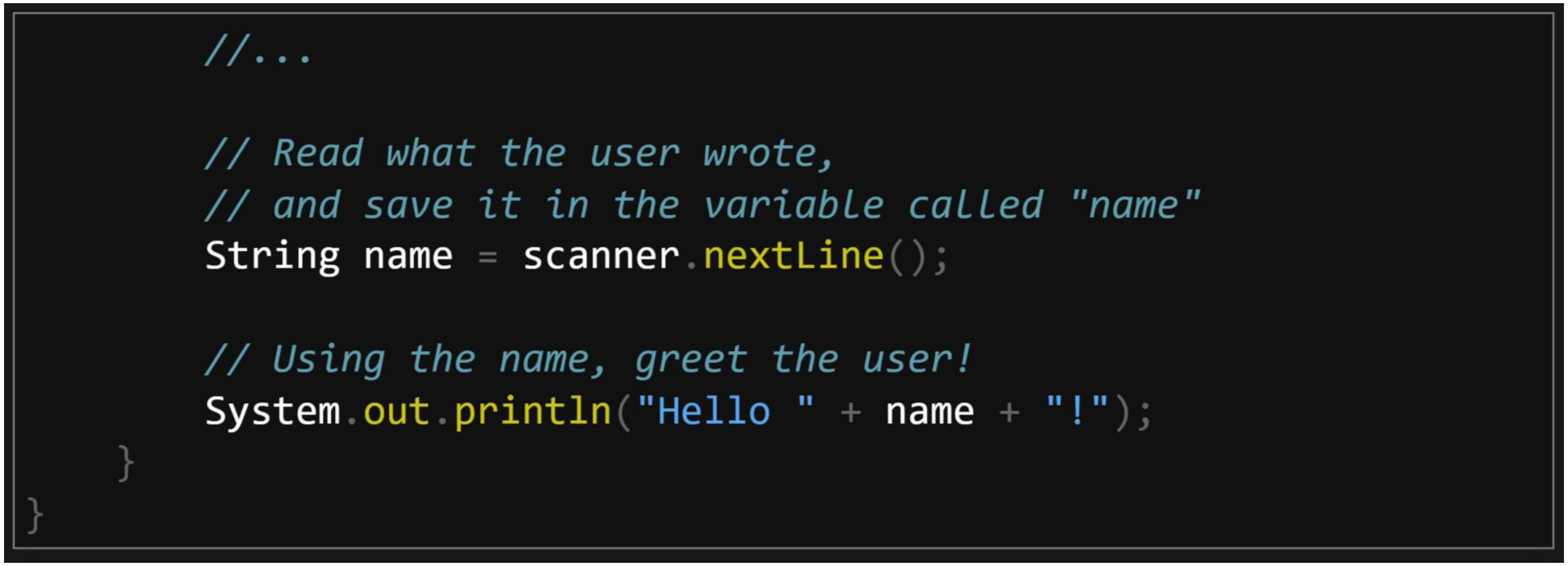
```java
// use somebody else's code, so we don't need to
// tell the computer how exactly to read input
import java.util.Scanner;

// same framework as before:
public class Talk {
    public static void main (String[] args) {
        // create a new variable of type Scanner
        // that reads from the console (System.in)
        Scanner scanner = new Scanner(System.in);

        // Ask the user about their name:
        System.out.println("Hi, what's your name?");
        //...
```

Let’s explain #2… (2/2)



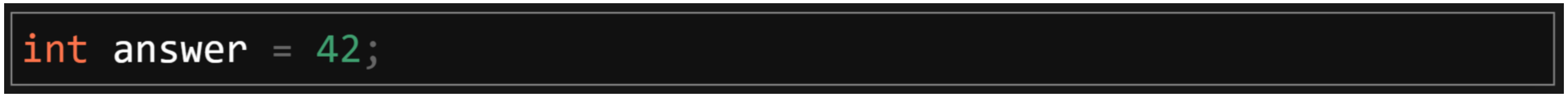
```
        //...

        // Read what the user wrote,
        // and save it in the variable called "name"
        String name = scanner.nextLine();

        // Using the name, greet the user!
        System.out.println("Hello " + name + "!");
    }
}
```

Let’s explain #2... (2/2)



```
        //...

        // Read what the user wrote,
        // and save it in the variable called "name"
        String name = scanner.nextLine();

        // Using the name, greet the user!
        System.out.println("Hello " + name + "!");
    }
}
```

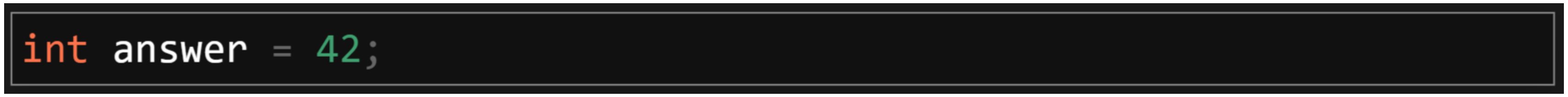
Java also knows numbers



```
int answer = 42;
```

- answer is a variable of type int
- type int (integer) stores whole numbers
  - like 7, 78482, -420

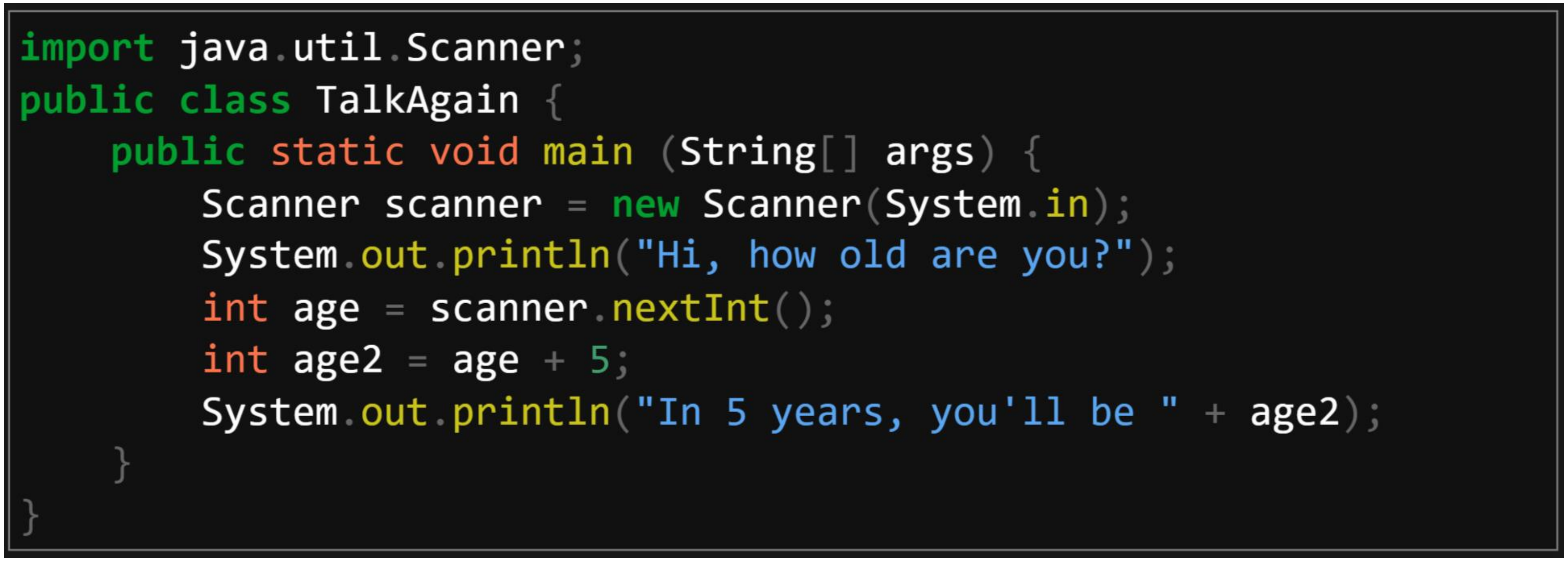
Java also knows numbers



```java
int answer = 42;
```

- answer is a variable of type int
- type int (integer) stores whole numbers
    - like 7, 78482, -420

We can also read numbers



```
import java.util.Scanner;
public class TalkAgain {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Hi, how old are you?");
        int age = scanner.nextInt();
        int age2 = age + 5;
        System.out.println("In 5 years, you'll be " + age2);
    }
}
```

Adding numbers works!

```
int num = 42 + 17;
int num2 = num + 7;
```

- it doesn’t matter if it’s the number itself or a variable containing a number
- some operators on numbers: +, -, *, /
- notice that an int divided by an int will still be an int
  - we’ll learn about floating point numbers soon

What have we learned?

- how to print text to console
- how to declare variables of type int, String
- how to read input from the console
- that operators like +, -, * and / exist

Apply your new-learned knowledge

- Let’s build a calculator!
- Suggestion on how to do that:
  - read one number
  - save it in a variable
  - read and save another number
  - add them
  - print the result

That’s it!

- Be encouraged to keep working on the calculator task :)
- Feel free to reach out
  - to send your results
  - to tell me about problems you ran into

Next lesson

- a few more types of variables
- control flow: if-statements, while-loops
- more practical examples!