COMP90041
Programming and Software Development

Week 1

Amani Abusafia
Trina Dey

THE UNIVERSITY OF MELBOURNE

The University of Melbourne acknowledges the Traditional Owners of the unceded land on which we work, learn and live: the Wurundjeri Woi-wurrung and Bunurong peoples (Burnley, Fishermans Bend, Parkville, Southbank and Werribee campuses), the Yorta Yorta Nation (Dookie and Shepparton campuses), and the Dja Dja Wurrung people (Creswick campus).

The University also acknowledges and is grateful to the Traditional Owners, Elders and Knowledge Holders of all Indigenous nations and clans who have been instrumental in our reconciliation journey.

We recognise the unique place held by Aboriginal and Torres Strait Islander peoples as the original owners and custodians of the lands and waterways across the Australian continent, with histories of continuous connection dating back more than 60,000 years. We also acknowledge their enduring cultural practices of caring for Country.

We pay respect to Elders past, present and future, and acknowledge the importance of Indigenous knowledge in the Academy. As a community of researchers, teachers, professional staff and students we are privileged to work and learn every day with Indigenous colleagues and partners.

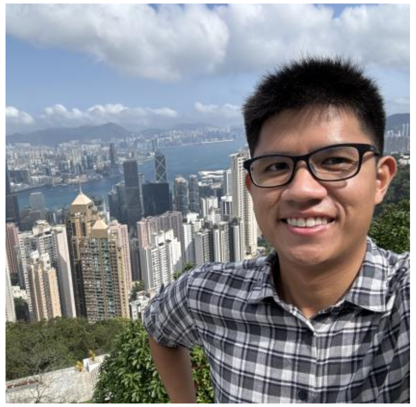
Topics:

- Teaching Staff
- Introduction to the Subject
- Assessments
- Academic Misconduct
- How to navigate Ed
- Subject Overview
- What's New?
- Getting Started With Java
- Setting up your Java Environment
- Data Types and Console I/O

Lecturers

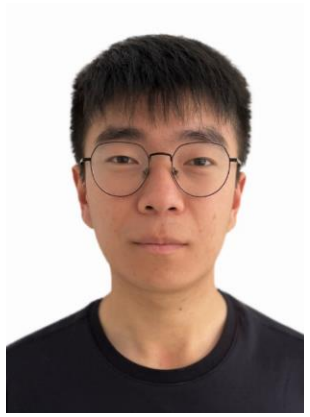
Lecturer & Subject Coordinator BSc Electronics (2006) , MS Comp Sci. (2009) , MS Data Sci. (2023) Worked in industry for 12+ years Java, Python, React, Cloud, ML.....	Lecturer BSc Comp Sci (2009), MS Comp Sci (2013), PhD in Eng (2024) Worked in Academia for 14+ years Java, Python, C++, SQL,...

Teaching Staff


Nat
(head tutor)

Niket

Sandhuni


Haoyu

Hening

Introduction to the Subject

- Programming Language used – **Java**. Why Java not Python?
  - Java is widely used in industry for programming.
  - Concepts of Object-Oriented Programming are not embedded within framework of Python but is supported via external libraries, making it entirely optional.

- Key objectives – Programming, Reading requirements, Testing, etc.
  - Handbook entry - https://handbook.unimelb.edu.au/2025/subjects/comp90041

- TextBooks
  - Walter Savitch, Absolute Java, 6th Edition, Addison Wesley.
  - SCHILDT, H. Java: The Complete Reference, 12th Edition: McGraw-Hill, 2022

Introduction to the Subject

- Weekly commitments -
  - Lectures – 2 hours (Friday, 9-11 am, Old Arts Building)
  - Tutorials – 2 hours (1 hour tutor demonstration, 1-hour self-practice in tutorial classrooms)
  - Workshops – self paced at home approx. 2-3 hours

- Subject Content
  - Canvas > Module Page
  - EdStem > Lessons

- Lectures takes place in person, but the live lecture sessions are automatically recorded in the classroom and is available in Canvas under > Lecture Capture.
- There may be some additional recording uploaded time to time in Lecture Capture. Announcements will be made
- Discussions are done on the EdStem Platform and not Canvas/LMS.

Assessments

Type	Week	Duration	Start Date	Due Date	%age
Practice Mid Sem Quiz	Week 4	30 min	28 Mar 25 17:00	11 Apr 25 08:00	0
Mid Sem Quiz	Week 6	30 min	11 Apr 25 09:00	11 Apr 25 09:30	10
Assignment 1	Week 6-8	3 weeks	07 Apr 25 09:00	28 Apr 25 09:00	25
Assignment 2	Week 10-12	3 weeks	12 May 25 09:00	30 May 25 23:59	25
Final Exam	During Exam Weeks	2 hour	TBD	TBD	40

The Quizzes & Exams will run online and on ***Lockdown Browser*** to ensure integrity.

Hurdle Requirements

- 3 hurdles in the subject.
  - a. atleast 50% (25 marks) of combined Mid sem quiz(10 marks) + Final Exam(40 marks). For eg - 4 marks in Mid Sem + 21 marks in Final Quiz = 25 marks combined or 7 marks in Mid-Sem + 18 marks in Final Quiz = 25 marks.
  - b. atleast 50% (25 marks) of combined Assignment 1 (25 marks) + Assignment 2(25 marks). For eg - 11 in A1 and 14 in A2 combines to 25 marks.
  - c. Overall 50% and above in the entire subject.
- See handbook for more information.

Extensions and Special Cons: Assignments

- **Short Extensions – upto 3 days**
  - Can be approved by Subject Coordinator
  - Can be rejected by Subject Coordinator if the reason is not appropriate
  - Complete the online declaration form - FEIT Short Extensions declaration form.

- **Long Extensions – 4 – 10 days**
  - Fill up the special consideration form with valid proofs

- **AAP**
  - Your AAP will define if you are pre-approved for certain number of days for extensions
  - Notify your subject Coordinator to get an extension

When to notify? Before Assessment Due date

See details in this module - here

Extensions and Special Cons: Exams

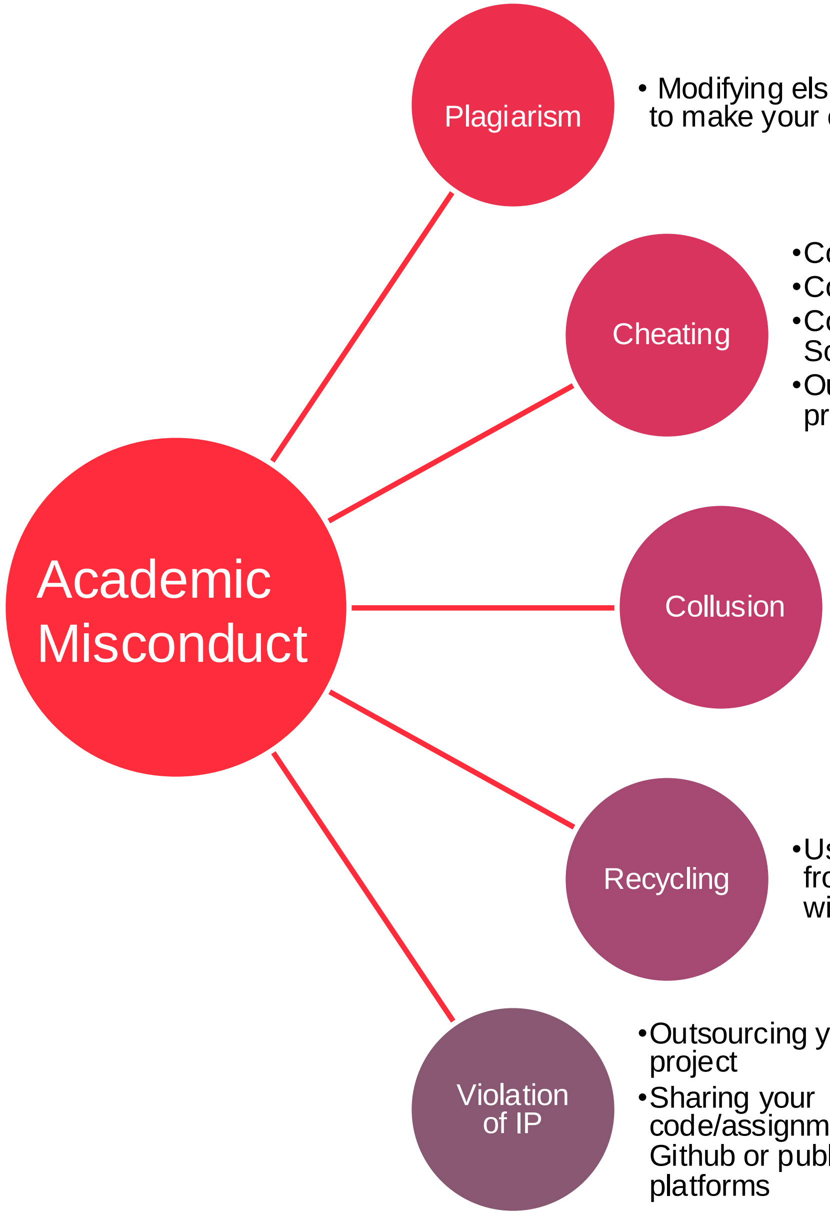
- **Mid Sem Quiz**
  - Fill the special con form with valid proof and **we will** arrange another time to take the in-person Quiz

- **Exam**
  - Fill the special con form with valid proof and **Centralised Exam team** will arrange a date

- **AAP -** Extra reading or writing time/ different room
  - Mid-Sem Quiz : – managed by us (Notify Subject Coordinator before)
  - Final Exam – managed by centralised exam team

When to notify/fill form? No later than upto 4 business days after the exam

Academic Misconduct



- Modifying else's code to make your own

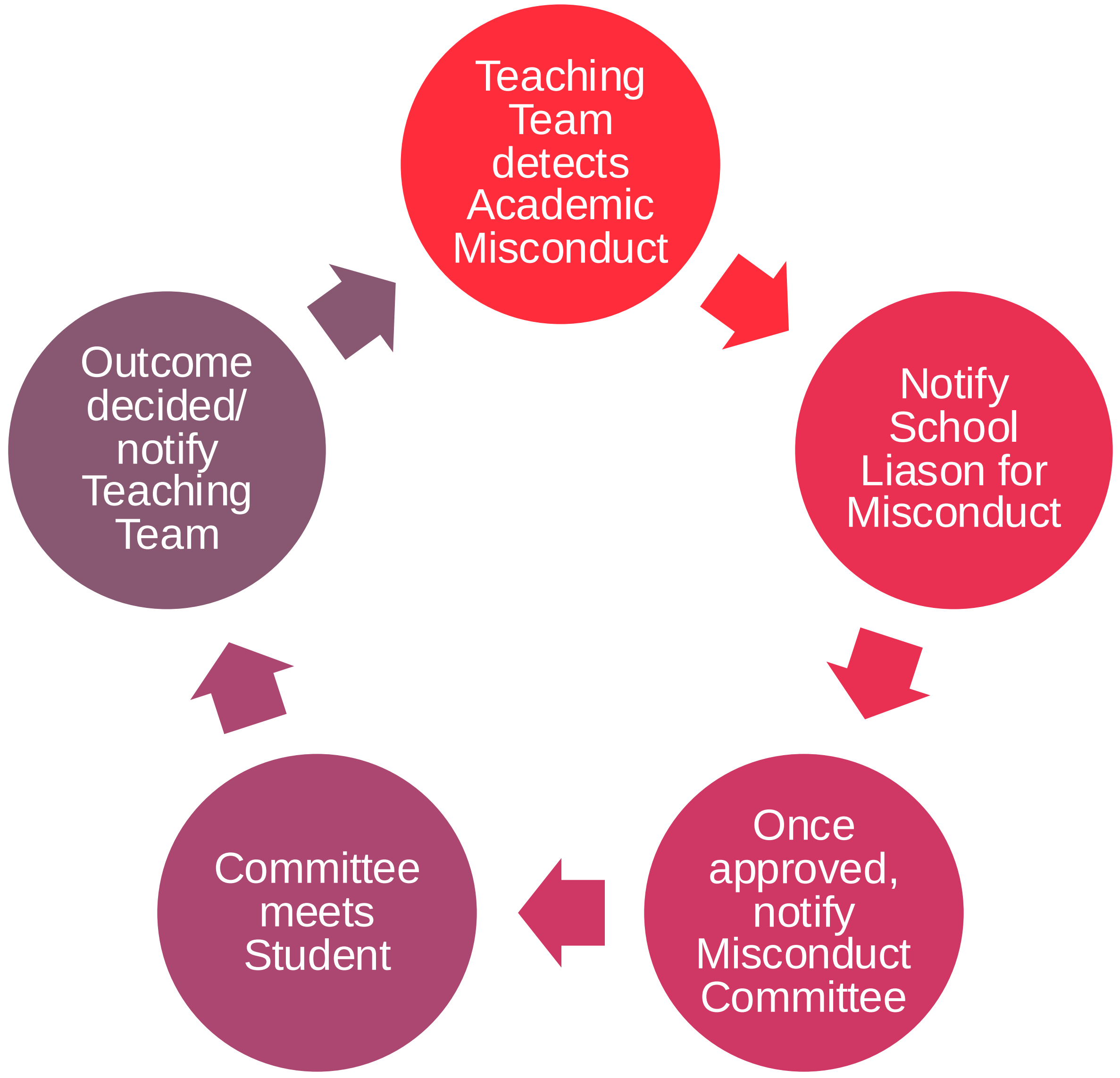
- Copying else's code
- Copying from internet
- Copying from AI Sources
- Outsourcing your project

- Two or more people in the class working together
- Getting help from outside the university

- Using your own work from elsewhere without notifying

- Outsourcing your project
- Sharing your code/assignment on Github or public platforms

Academic Misconduct


Teaching Team detects Academic Misconduct
Notify School Liason for Misconduct
Once approved, notify Misconduct Committee
Committee meets Student
Outcome decided/ notify Teaching Team

Communication Process

- Use Ed Discussions actively,
  - find (Ctrl + F) to look for similar queries
  - If not found, post new queries
  - Monitor the discussion board to learn something new
- Watch out for announcements from Canvas, Check your email inboxes
- Reach out to your tutors in your tutorials
- More help needed -
  - Attend extra consultations with teaching team every week, starts Week 3 or 4
  - Reach out to Amani or Trina via email for more clarifications

Be Professional! Use Formal Language with Teaching Team! Be Polite and Respectful!

What's new this Sem?

- 2 assignments instead of 3
  - 3 assignments worth 70% - too much for students
  - Assignments reduced to 2 – 25% each

- Lecture delivery style modified
  - No time to look at the lecture content before lectures
  - Go through the concepts in first half and live coding in the second half

- New advanced concepts introduced
  - Upkeep with new trends in Java such as Lambdas.
  - Some pre-requisite concepts like Threading used in subjects COMP90018

- **Exams -**
  - We will share practice exam with you somewhat similar to the final exam

Subject Overview

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
▪ Subject Overview ▪ I/O ▪ Data Types	▪ Operators ▪ Control Flow	▪ Classes	▪ Classes II	▪ Arrays	▪ OOPs	▪ OOP continued Good Friday
Easter Break	Week 8	Week 9	Week 10	Week 11	Week 12	SWOT VAC
Non-Teaching Period	▪ Java Interfaces	▪ Exception Handling	▪ File Handling	▪ Generics ▪ Collections	▪ Advanced Topics (Non-Examinable) ▪ Revision	**Non-Teaching Period**

Getting the Most Out of This Course

- **Be Proactive**
  - Pay attention to the learning outcomes in each lectuere
  - Regularly check your progress against course objectives
- **Manage Your Time**
  - Plan ahead for assignments and assessments
  - Break tasks into smaller steps to avoid last-minute stress
- **Engage with Others**
  - Collaborate with classmates—discuss ideas and challenges
- **Ask for Help Early**
  - If you're struggling, don't wait—reach out for support
  - Use available resources like Ed and consultations
- **Embrace the new concepts and enjoy the learning process!**

Getting to know you!

THE UNIVERSITY OF MELBOURNE

PollEv.com
/tdey

Getting Started – Overview

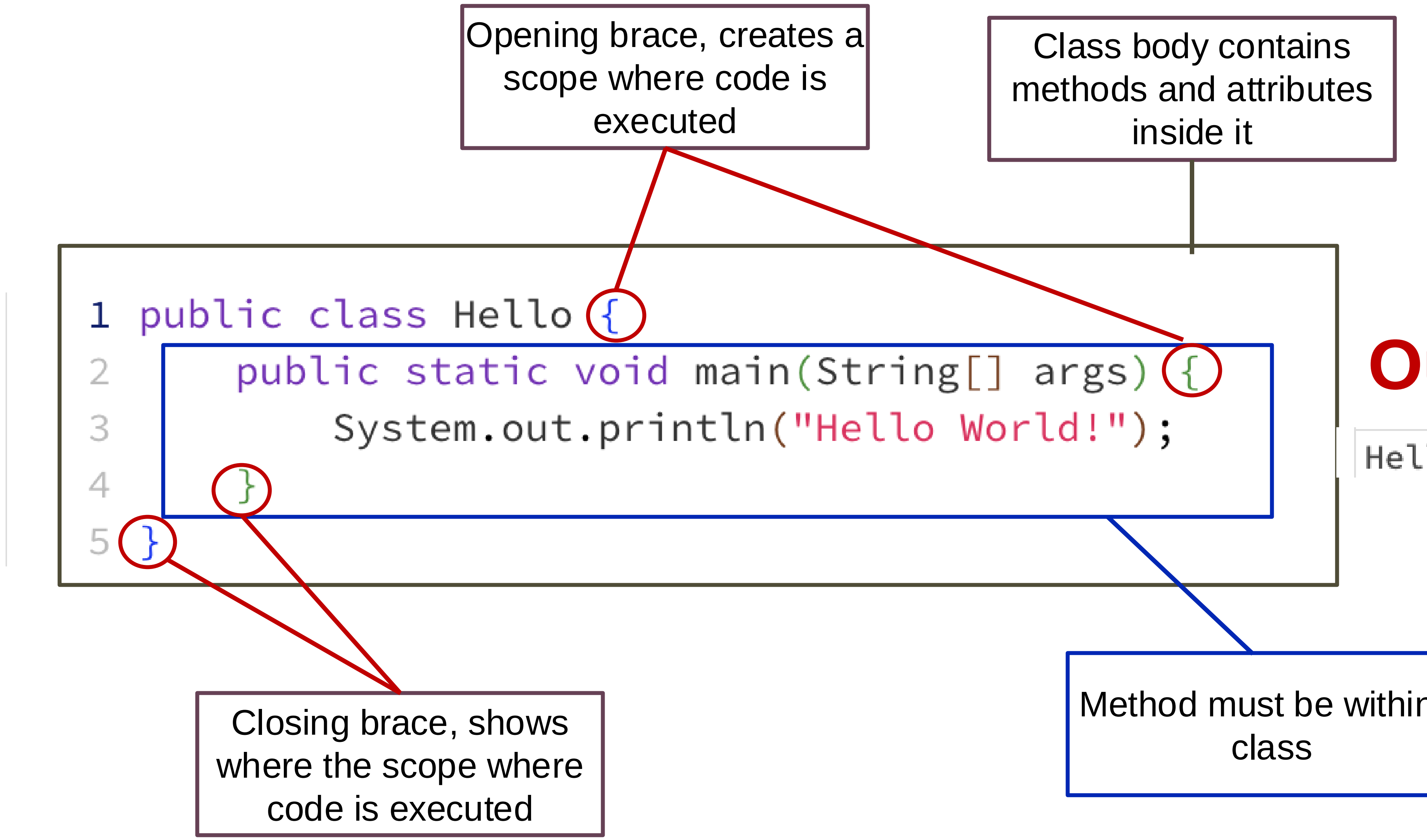
In this part of the lecture, you will learn about:

- **Java – History**
- **First Program**
- **Compilers and Interpreters**
- **Data Types**
- **I/O**
- **Operations**

Java – Brief History

- **James Gosling** at Sun Microsystems started Java in 1991
- Originally designed for **home appliances** – dishwashers, TV sets
- Original Motivation – write and compile code once, runs everywhere (every OS)!
- Java – High Level Language > Intermediate Byte Code > Machine Code
- **Oracle** took over Sun Microsystems
- Latest Java Version – Java 23

First Program – Hello World!



Opening brace, creates a scope where code is executed

Class body contains methods and attributes inside it

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

Closing brace, shows where the scope where code is executed

Method must be within a class

Output

```
Hello World!
```

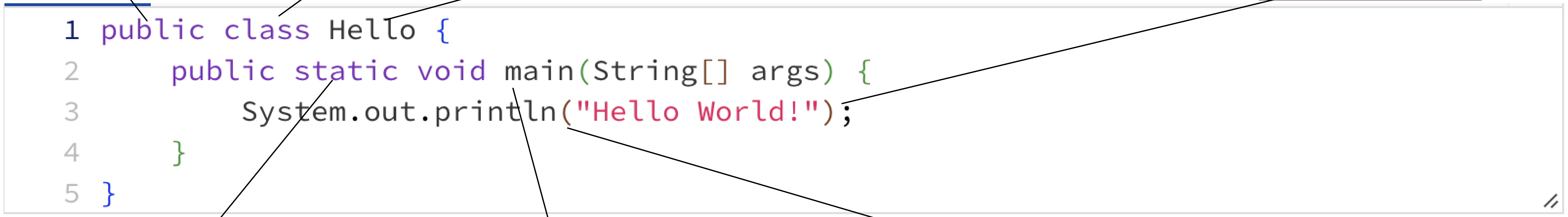
First Program – Hello World!

Public – accessible to all
Private – hidden from some

Everything goes in a class in Java

Every class has a name

Message we are trying to print



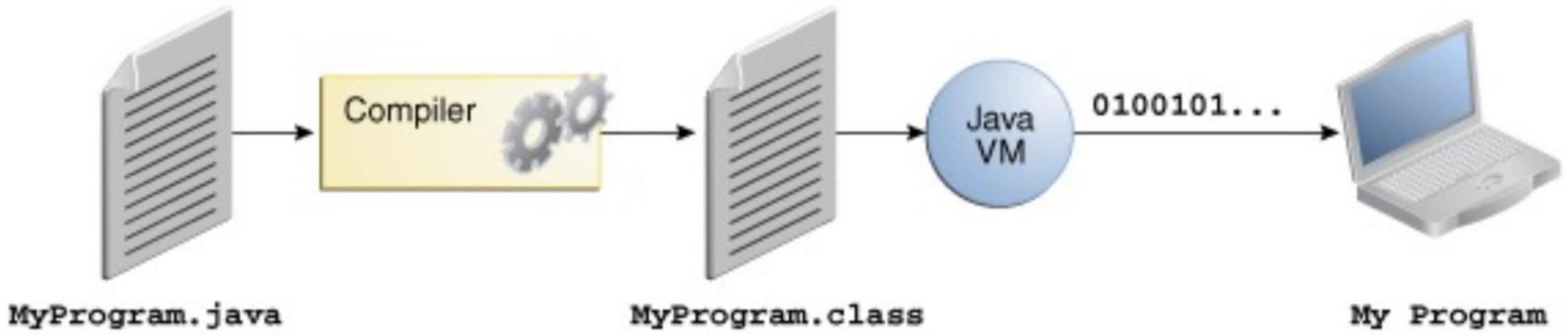
```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

We will discuss static more in Week 4

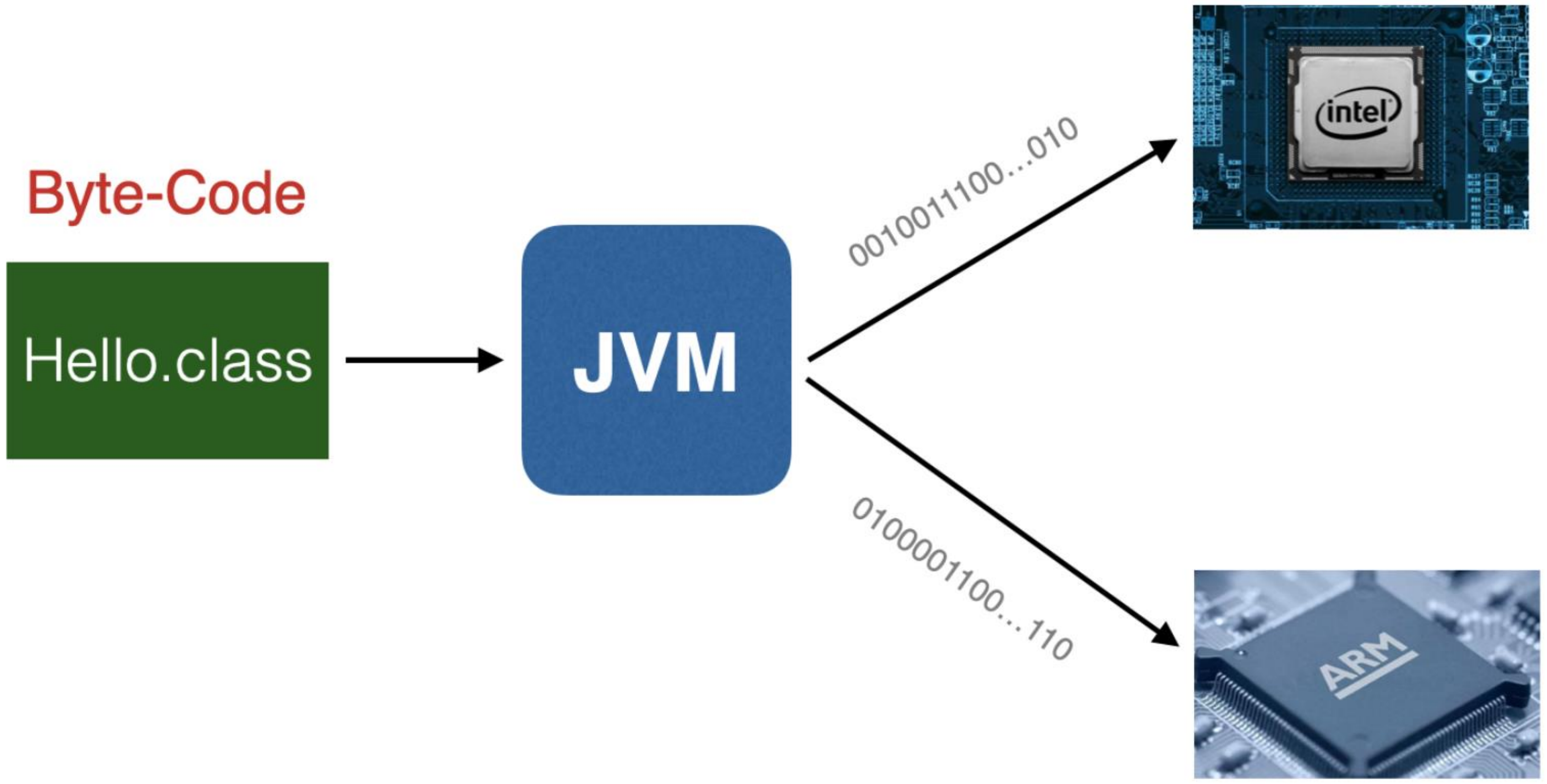
Main is the starting point for any program. Java compilers looks for a method named main

System – java library
Out – refers to standard output
Println – prints with newline at the end

Compilers & Interpreters



Compilers & Interpreters



Compilers & Interpreters

- Java compiler – javac
- JVM – Java Virtual Machine – helps execute bytecode. Two ways
  - Interpreter – interprets the bytecode to machine executable code
  - JIT – Just in Time Compiler – compiles the bytecode into executeable code on need basis
- JRE – Java Runtime Environment – System libraries like you used to print Input/Output
- JDK – Java Development Kit - Helps you to write or develop the programs. Provides series of system libraries.

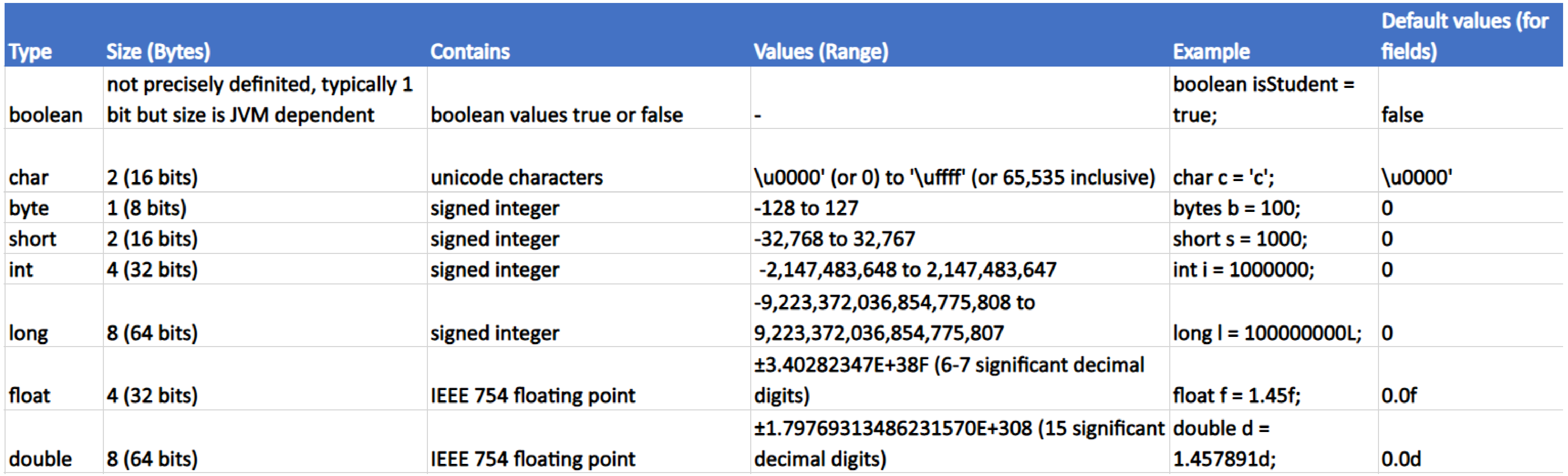
General Code Structure

```
class ClassName {
        public static void main (String [] arguments) {
         → Your code goes here ←
        }
}
```

Data Types

- **In general, a computer program consist of two parts: code and data.**
- **Code is the text of the program that determines what operations the program performs.**
- **Data is what the code operates on.**
  - Each datum (singular of data) has a **type**.
- **Java distinguishes between three groups of data types: primitive, class, and array.**

Data Types



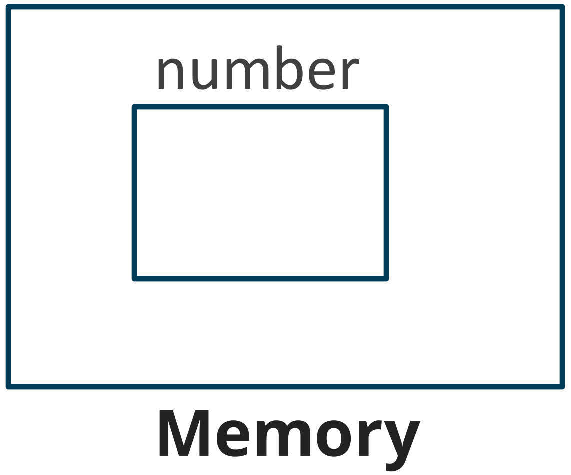
Type	Size (Bytes)	Contains	Values (Range)	Example	Default values (for fields)
boolean	not precisely definited, typically 1 bit but size is JVM dependent	boolean values true or false	-	boolean isStudent = true;	false
char	2 (16 bits)	unicode characters	\u0000' (or 0) to '\uffff' (or 65,535 inclusive)	char c = 'c';	\u0000'
byte	1 (8 bits)	signed integer	-128 to 127	bytes b = 100;	0
short	2 (16 bits)	signed integer	-32,768 to 32,767	short s = 1000;	0
int	4 (32 bits)	signed integer	-2,147,483,648 to 2,147,483,647	int i = 1000000;	0
long	8 (64 bits)	signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	long l = 100000000L;	0
float	4 (32 bits)	IEEE 754 floating point	±3.40282347E+38F (6-7 significant decimal digits)	float f = 1.45f;	0.0f
double	8 (64 bits)	IEEE 754 floating point	±1.79769313486231570E+308 (15 significant decimal digits)	double d = 1.457891d;	0.0d

Variables

- Variables are named locations that store data
- FOMAT:

```
type variableName;
 int number;
```

- Variables can have different values at different times.
- In Java, variable names begin with a letter, followed by letters, digits, and underscores (_).
- Best practice: make them descriptive, but not too long (clear abbreviations are okay)
- Each variable must be declared before being used, specifying its type.



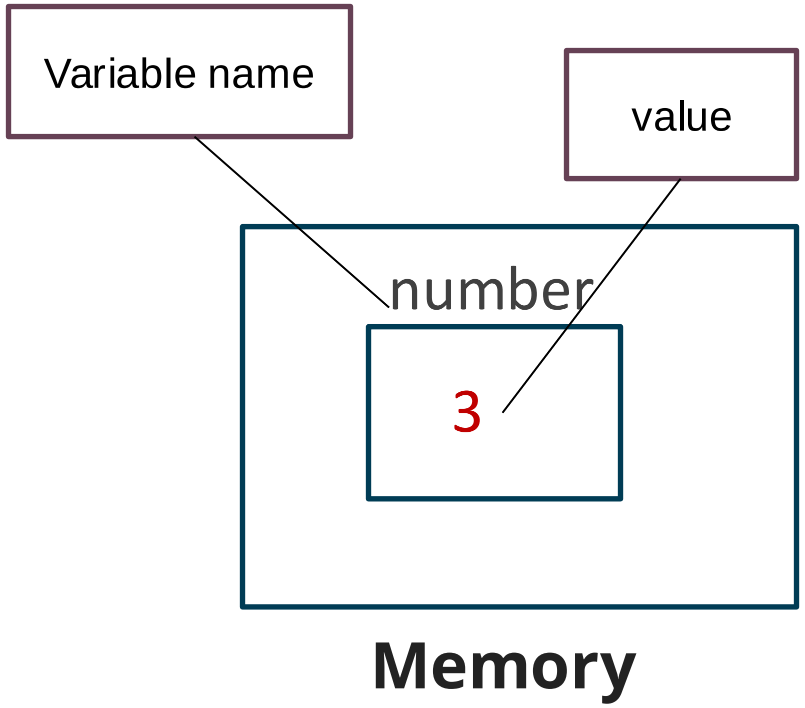
Variables

- **Variables must be assigned a value before being used.**
- **This can be done in two ways:**

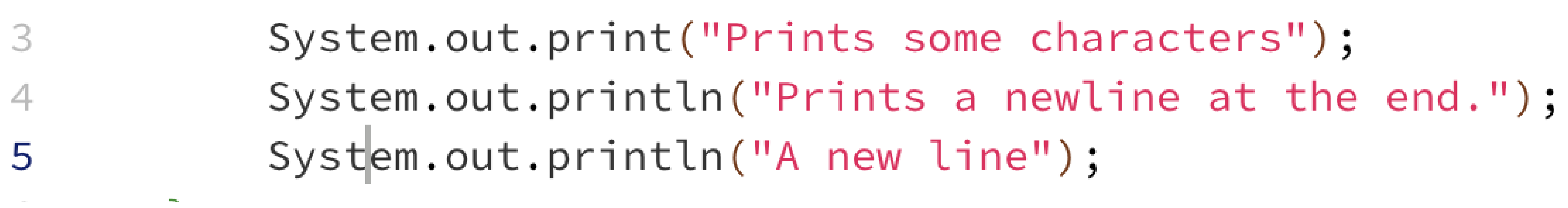
```
int number;
number = 3;
```

- **OR Declaration and assignment can also happen in a single statement:**

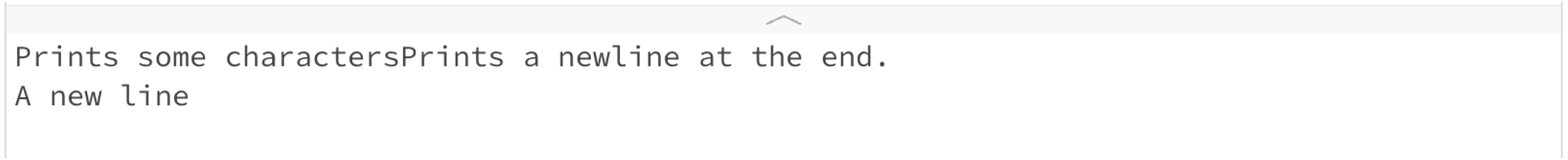
```
int number = 3;
```



I/O: Standard Output



```java
		System.out.print("Prints some characters");
		System.out.println("Prints a newline at the end.");
		System.out.println("A new line");
```

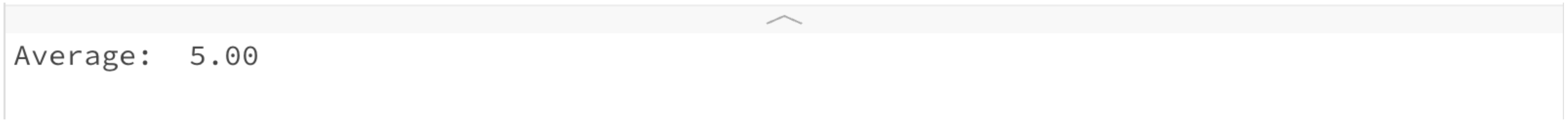


```
Prints some charactersPrints a newline at the end.
A new line
```

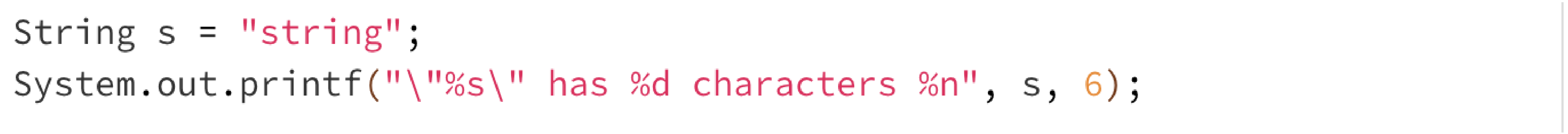
I/O: Formatted Output

```
double average = 5.0;
System.out.printf("Average: %5.2f", average);
```

output

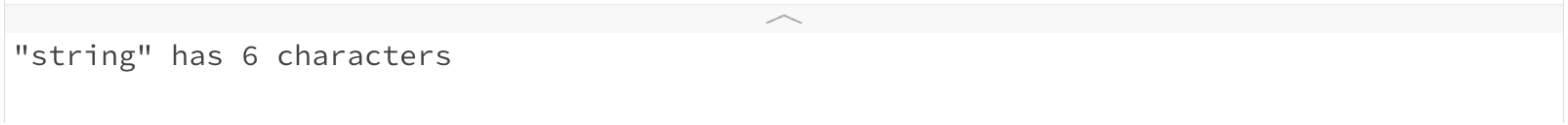


```
Average:  5.00
```



```
String s = "string";
System.out.printf("\"%s\" has %d characters %n", s, 6);
```

output



```
"string" has 6 characters
```

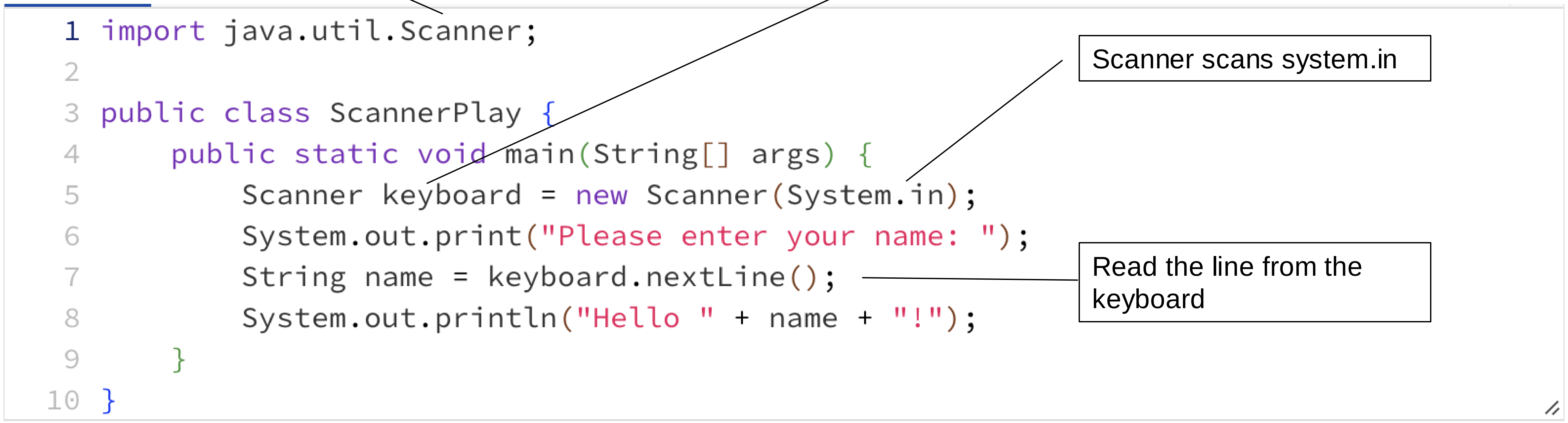
I/O: Standard Input

Include the scanner library using import statement

Write the first line defining the scanner named as keyboard.

Scanner scans system.in

Read the line from the keyboard



```java
import java.util.Scanner;

public class ScannerPlay {
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Please enter your name: ");
        String name = keyboard.nextLine();
        System.out.println("Hello " + name + "!");
    }
}
```

I/O: Command Line Input

What's that?

```java
public class HelloStranger {
    public static void main(String[] args) {
        System.out.println("Hello " + args[0] + "!");
    }
}
```

Why 0? Counting starts from 0 in most programming languages

```
$ javac Hello.java
```

```
$ java HelloStranger Java
Hello Java!
```

The input from command line

Output

Live Coding

Java on TextPad

Demo!

Java using IDE

Demo!

Java on EdStem

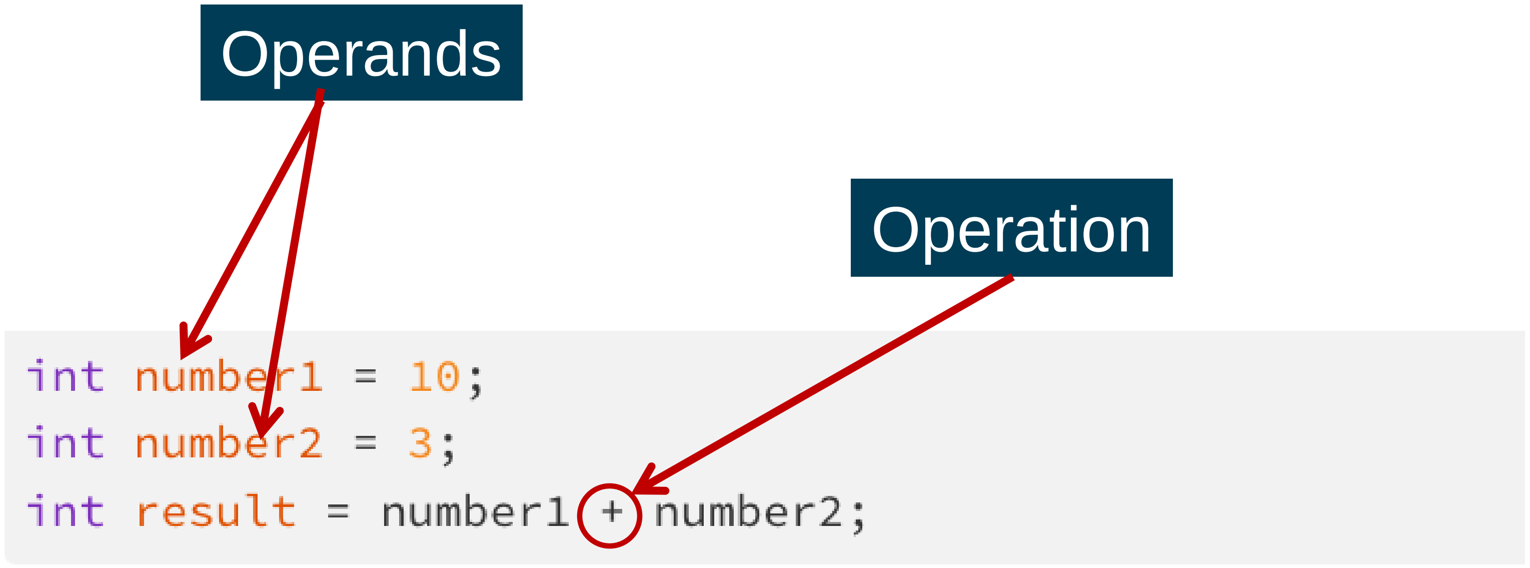
Demo!

Number Types

- Each type has certain operations that apply to it.
- Common ***operations (also called operators)*** for primitive number types are:
  - Addition (+) and Subtraction (-)
  - Multiplication (*) and Division (/) as well as Modulo (%), i.e., the remainder

- The numbers we apply the operation to are called the ***operands***.
- The **data type of the result** is the same as the type of the operands.

Number Types



```java
int number1 = 10;
int number2 = 3;
int result = number1 + number2;
```

Operations are used to construct ***expressions***, which have values that can be assigned or used as operands:

```java
int answer=(2+4)*7;
```

Integer vs Float Division

Integer Division

```
System.out.println(5/2);
```

The Output will be:

```
2
```

Float Division

```
System.out.println(5.0/2);
```

```
2.5
```

Comparison Operations

- The following **comparison operations** also work for number types:

```
<  : less than
<= : less than or equal to
>  : greater than
>= : greater than or equal to
== : equal to
!= : not equal to
```

- Comparisons always **return** a **boolean (true / false)** value

```
System.out.println(5!=4);
```

```
Output is True
```

Operations for Booleans

- You can construct **logic statement** in Java using:

```
&& (AND) : is true if both operands are true
|| (OR) : is true if either operand is true
```

- Both of these are so-called 'short-circuit' operations, *i.e.*, the second **operand** is only evaluated if necessary.
  - If the first argument of && is false, or the first argument of || is true, then the second one is not necessary because it cannot change the value of the expression.
- Then, there is also **negation**:

```
! (NOT) : is true if its operand is false
```

Comparison Operations

```
System.out.println((5!=4) && (5>3));
```

The Output will be:

```
true
```

Increments and Decrements

- Increments and decrements are a **special expressions**
- There are **two** types of increments and decrements:

pre

- The **pre-increment** increments a number by **1** **before** returning the value.

++y

- The **pre-decrement** - decrements a number by **1** **before** returning it:

-- y

post

- The **post-increment** increments a number by **1** **after** returning the value.

y++

- The **post-decrement** - decrements a number]by **1** **after** returning it:

y--

Increments and Decrements

```
int x = 5;
```

pre

```
System.out.println(++x);
```

```
Output is 6
```

post

```
System.out.println(x++);
```

```
Output is 5
```

- Pre/post increment/decrements can also be used as statements rather than expressions:

```
++x;          OR          x++;
```

Additional Reading Resources

- WALTER, S. Absolute Java, Global Edition. [Harlow]: Pearson, 2016. (Chapter 1, 2 and 3)
- SCHILDT, H. Java: The Complete Reference, 12th Edition: McGraw-Hill, 2022 (Chapter 4)
- Language Basics (accessible on 14-02-2024) Oracle’s Java Documentation. Available at: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html

See you next time!