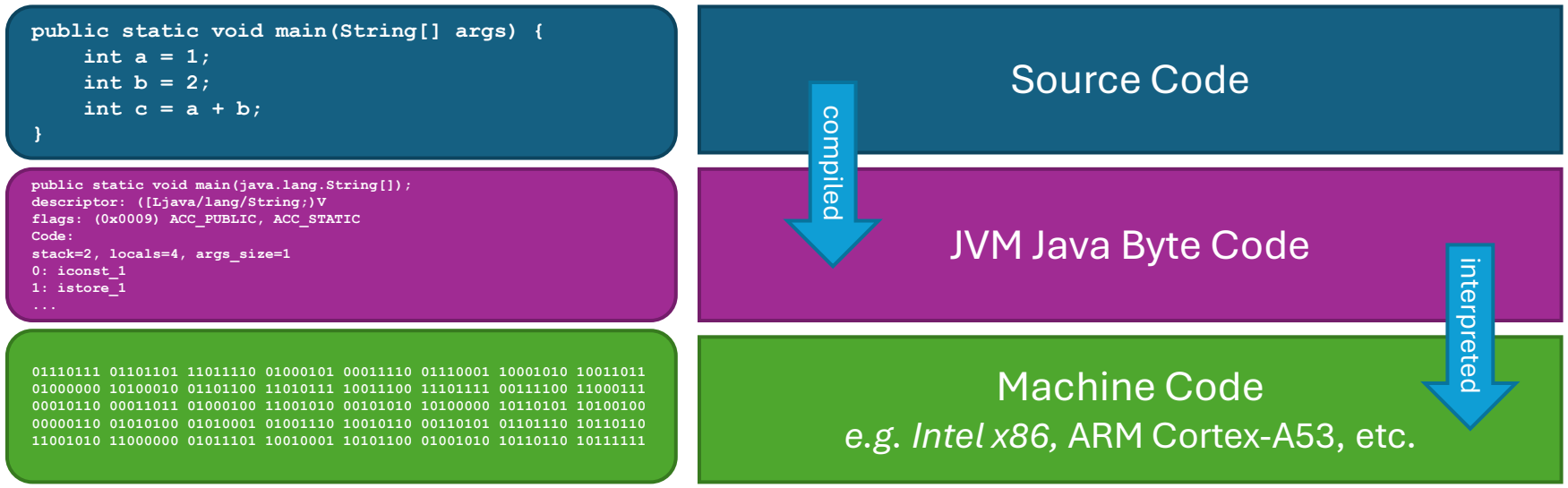
Java Programming
Object-oriented Programming

Fundamentals of Java Programming

What is a program?

- A computer program is nothing more than a precise set of instructions that the computer must follow:
  - ***Begin***
    Do X
    Do Y
    Do Z
    End
- The computer will follow those instructions exactly – it can only do what you tell it to do

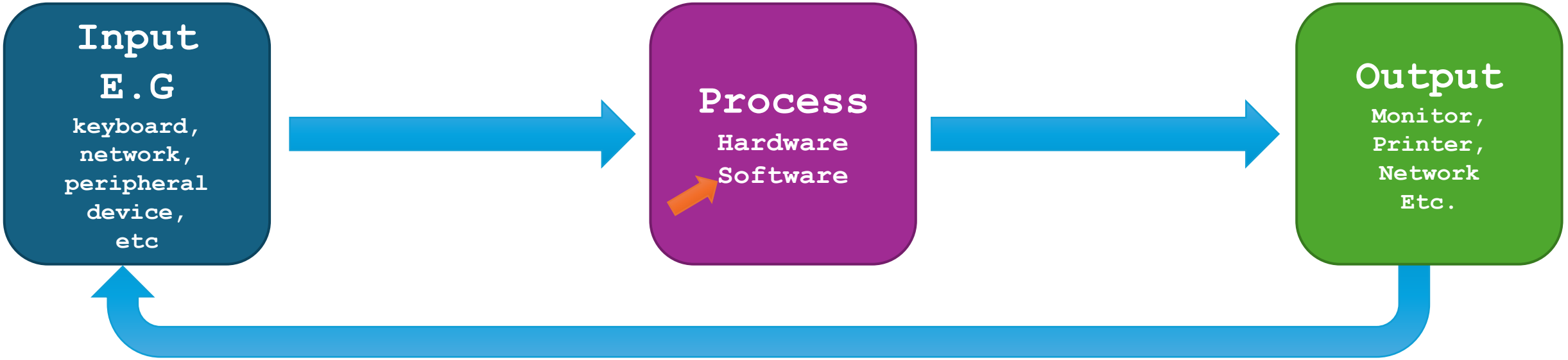
Java and the Java Virtual Machine



Notes

1. Java and Bytecode example source: https://dzone.com/articles/introduction-to-java-bytecode
2. The binary code is not real machine code version of the Java byte code, it is just for illustrative purposes only

Input – Process - Output



Simple Java Program #1

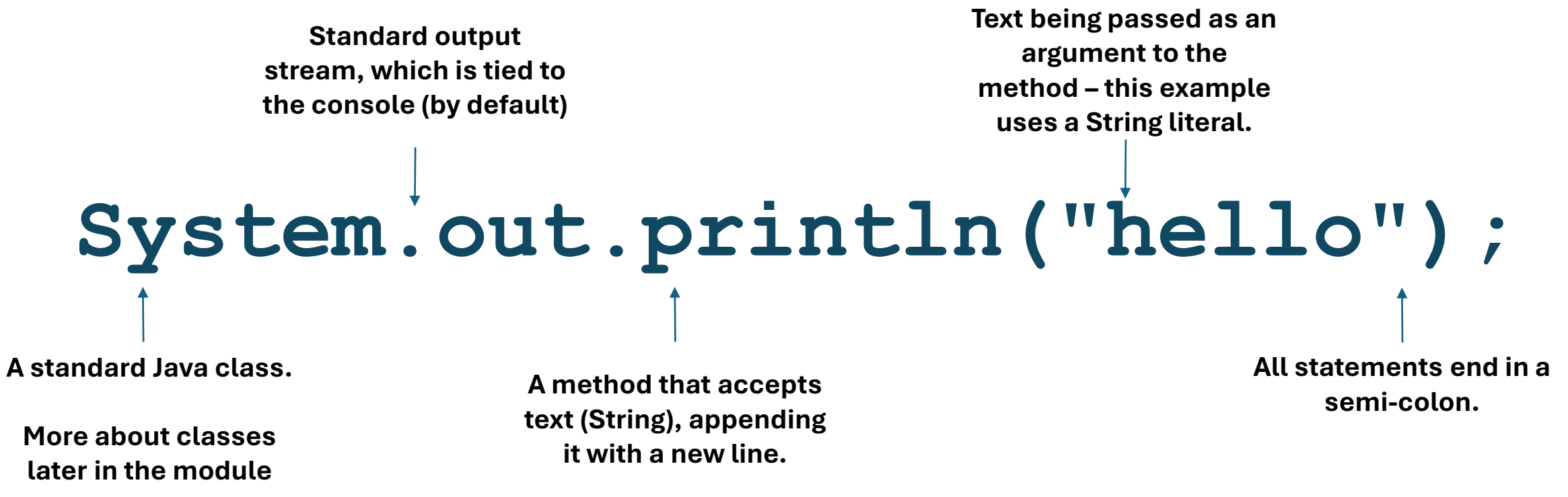
- Demonstrates:
  - Netbeans IDE
  - The *main* method as the program's entry/exit point
  - Basic Java syntax
  - A Java *statement*
  - Outputting text to the standard output stream (STDOUT)

Simple Java Program #1

```
public class HelloWorld {

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

Displaying output to the user

- Our programs will have human beings as receiver of any results
- Therefore, the programs must output the results somewhere
  - This will be the 'console', a text-based interface
  - We use **`System.out.println()`** to display information


Standard output stream, which is tied to the console (by default)
Text being passed as an argument to the method – this example uses a String literal.
System.out.println("hello");
A standard Java class.
More about classes later in the module
A method that accepts text (String), appending it with a new line.
All statements end in a semi-colon.

Output & Error streams

- **`System.out`**
  - Output Stream
  - A stream opened by the operating system for the application that data is sent to in a serial manner, i.e. one character at a time.
  - File Descriptor 1 (POSIX standard)
  - Typically, the output stream's end point is a terminal
- **`System.err`**
  - Error Stream
  - Also opened by the operating system for the application.
  - File Descriptor 2 (POSIX standard)
  - Typically, the error stream's end point is a terminal
- Platform dependent, for example Windows and Linux implementations are different
  - However, Java provides a consistent cross-platform interface to it.

Variables, constants, and literal values

- Programs process data
- ***Variables*** are named objects that store data that is expected to change, for example an incrementing counter.
- ***Constants*** are named objects that store data that will not change, for example the number of miles in a full marathon
- Data that can be manipulated are known as ***variables***
  - **mutable**
- Data that can't be manipulated are known as ***constants***
  - **Immutable**
- Literal values things like `1`, `"Hello"`, `3.14` and are by definition constant – you can't reassign 1 as 99!
  - They are typically used when assigning a value to a variable or named constant

Primitive data types – Statically Typed

- Java is a **statically typed** language – data can't change its type once it has been defined

- When you declare a variable you also explicitly state its type

- `int`; `short`; `long`; `float`; `double`; `byte`; `boolean`; `char` are examples of **primitive types**

Primitive types in Java

- Numerical data/real numbers
  - Integer (whole number)
  - Fractional (with a decimal point)
- An integer type in Java is `int`
  - Also, `short` and `long`
- A fractional type in Java is `double`
  - Also, `float`
- Other primitive types are
  `byte`, `char`, `boolean`

Restricted operations on type

- Java is a **statically typed** language – checks what you're allowed to do to the data
- The type provides information to the Java compiler to help check that legal operations are being performed on the data
  - Legal operation: For example, two numbers can be mathematically divided or multiplied:
    1 / 2 ; 9.6 * 4.89 ; …
  - Illegal operation: Text and a number can't be mathematically divided or multiplied:
    "Cow" / 3.14 ; "Aadvark" * 900

Simple Java Program #2

- Demonstrates:
  - Netbeans IDE
  - The ***main*** method as the program's entry/exit point
  - Basic Java syntax
  - Java ***statements***
  - Outputting text to the standard output stream (STDOUT)
  - Declaring ***variables*** of ***primitive types*** to store values
  - Compute results by using ***arithmetic operators*** and the ***assignment operator***
  - Repeatedly performing the same set of statements using ***iteration*** (loop)

Simple Java Program #2

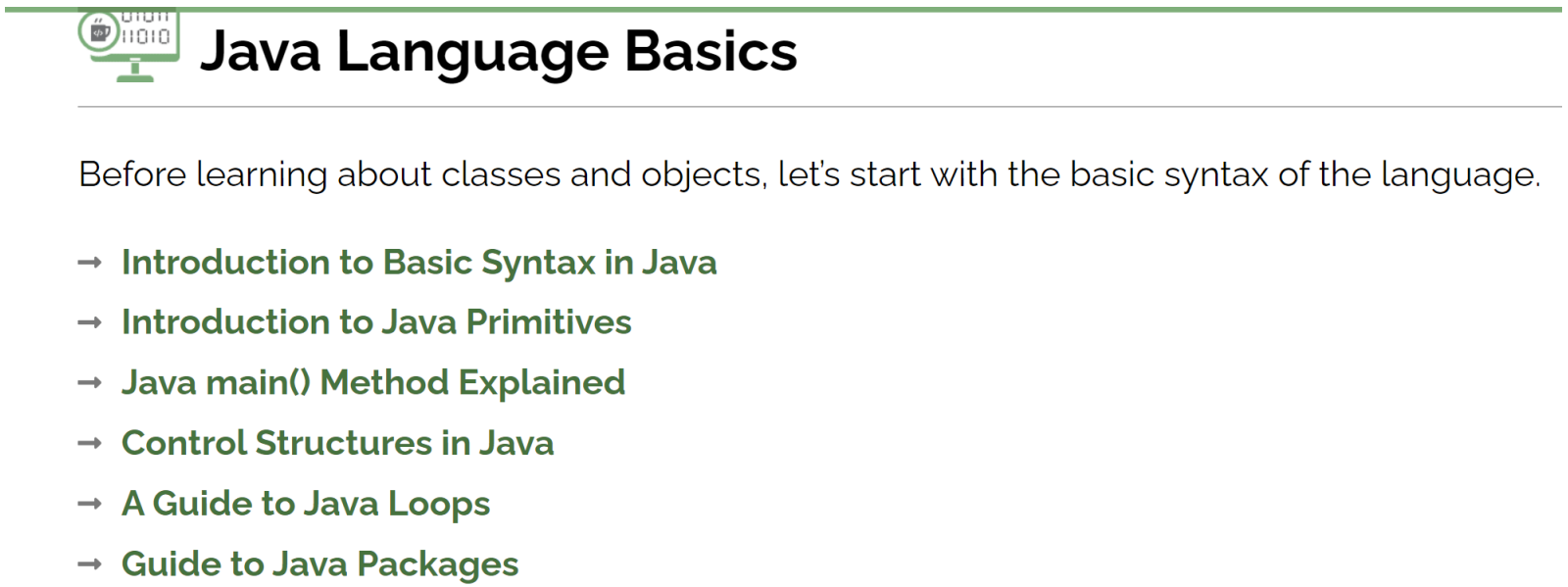
```
public class HelloWorldWithLoop {

    public static void main(String[] args) {

        String message = "Hello, World from NetBeans with a loop!";

        System.out.println(message);

        int a = 10;

        int b = 5;

        for (int i = 1; i <= 5; i++) {

            System.out.println("Iteration " + i + ":");

            int sum = a + b;

            System.out.println("The sum of a and b is: " + sum);

            System.out.println("This is iteration number " + i + "\n");

        }

    }

}
```

Simple Java Program #3

```
import java.util.Scanner;

public class HelloWorldWithScanner {

  public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      double num1, num2;

       // User input for the first number

       System.out.print("Enter the first number: ");

       num1 = scanner.nextDouble();

      // User input for the second number

      System.out.print("Enter the second number: ");

       num2 = scanner.nextDouble();

     }

}
```

Lab Tasks

- Download and have a quick tour into NetBeans IDE (10 minutes): Java Quick Start Tutorial (apache.org)
- Get Started with Java by completing the following tutorial – the 1st 6 sections (30 minutes): Get Started with Java | Baeldung



Java Language Basics

Before learning about classes and objects, let's start with the basic syntax of the language.

- Introduction to Basic Syntax in Java
- Introduction to Java Primitives
- Java main() Method Explained
- Control Structures in Java
- A Guide to Java Loops
- Guide to Java Packages

Case Study

- Develop a simple static calculator app using java