# COIS 2240
# Software Design & Modelling

## Lecture 2

## Basics of Java Programming

**Taher Ghaleb**

TRENT UNIVERSITY

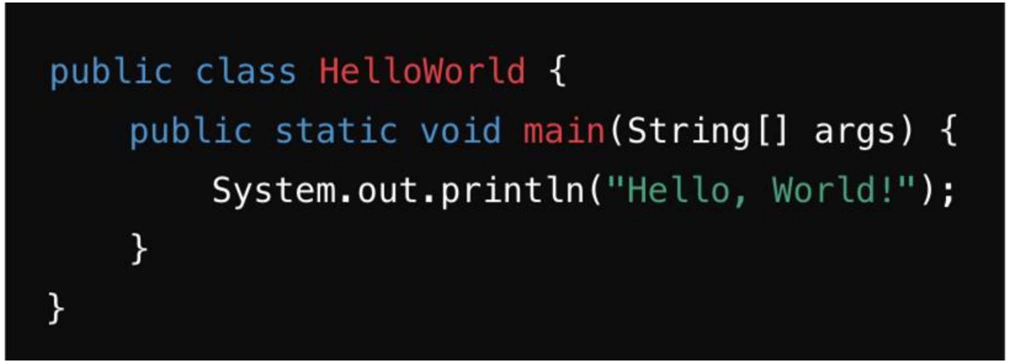
The content on the slides are collected from various sources (not the textbook)

## Introduction to Java

- General-purpose, object-oriented programming language
- Created in 1995 by Sun Microsystems (acquired by Oracle in 2010)
- Platform independent: write once, run anywhere
- Typed language
- Typically uses JDK (Java Development Kit) and IDEs like Eclipse and IntelliJ
- Uses JVM (Java Virtual Machine) to run compiled bytecode
- Automatic memory management via garbage collection

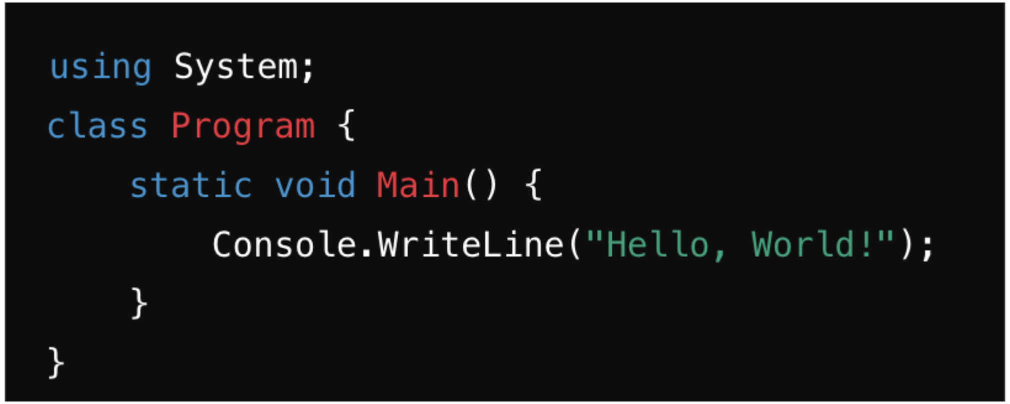
## Basic Syntax in Java

**Java**



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

**C#**



```csharp
using System;
class Program {
    static void Main() {
        Console.WriteLine("Hello, World!");
    }
}
```

## Data Types

**Primitive Data Types in Java:**

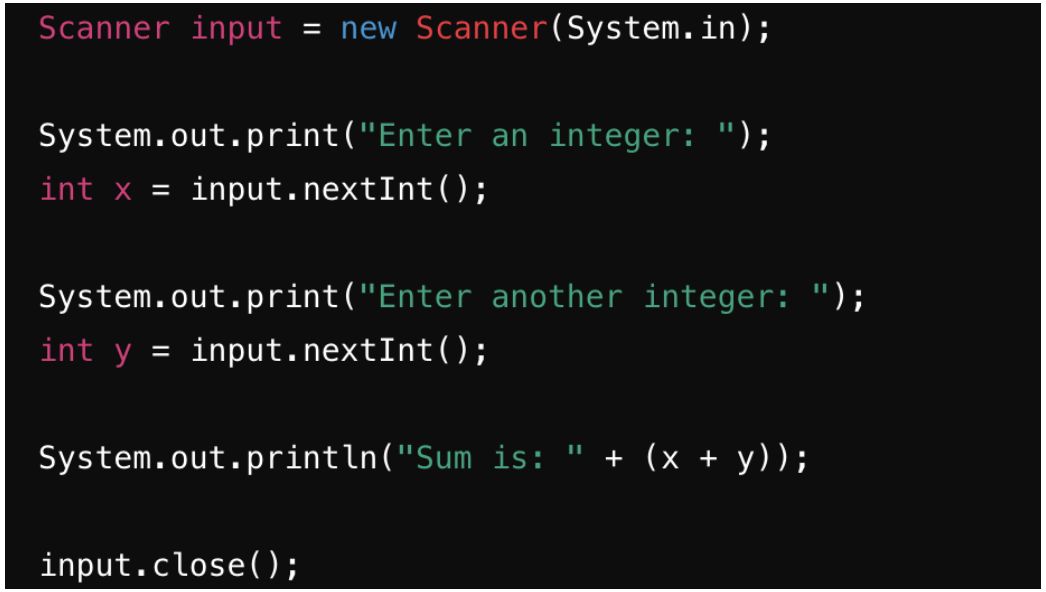
- *int*, *long*, *short*, *double*, *boolean*, *char*, *float*, *byte*
- Objects and arrays are reference types

**C#:**

- Similar primitive data types but with nullable types (*int?*, *double?*, etc.)
- *bool* instead of *boolean*

## Console Input/Output

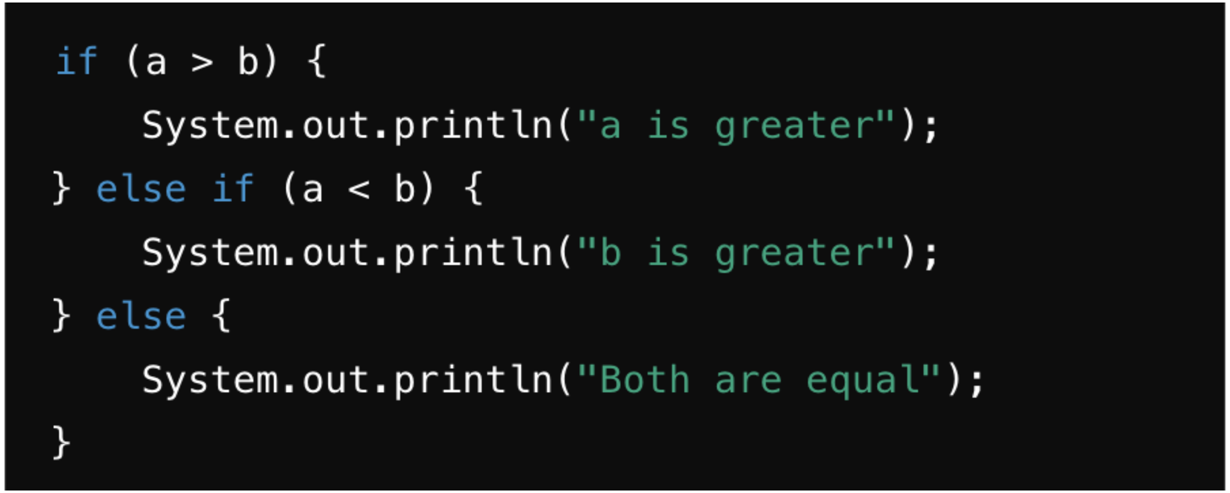
- Java provides mechanisms to read user input from the console and write output to the console window



```
Scanner input = new Scanner(System.in);

System.out.print("Enter an integer: ");
int x = input.nextInt();

System.out.print("Enter another integer: ");
int y = input.nextInt();

System.out.println("Sum is: " + (x + y));

input.close();
```

## Conditional Statements

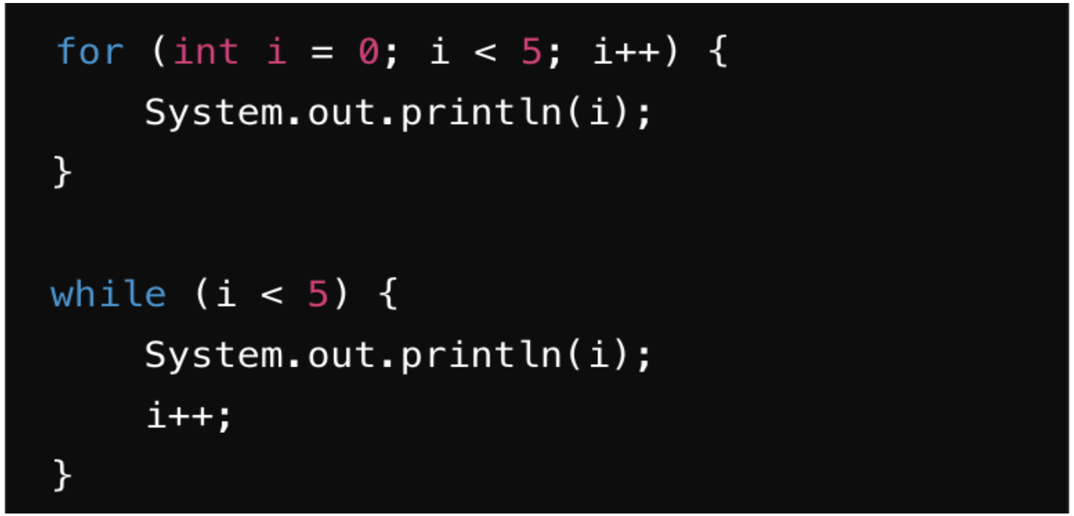
- Same *if-else* syntax in Java and C#
- Ternary operators ( *condition ? value1 : value2* ) are present in both



```
if (a > b) {
    System.out.println("a is greater");
} else if (a < b) {
    System.out.println("b is greater");
} else {
    System.out.println("Both are equal");
}
```

## Control Structures - Loops

- Same syntax for *for*, *while*, and *do-while* loops
- C# offers an explicit *foreach* keyword, but Java uses *for*



```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}

while (i < 5) {
    System.out.println(i);
    i++;
}
```

## Arrays

- Same syntax for declaring and using arrays
- Arrays in both Java and C# are fixed in size after initialization

```
int[] numbers = {1, 2, 3, 4, 5};
```

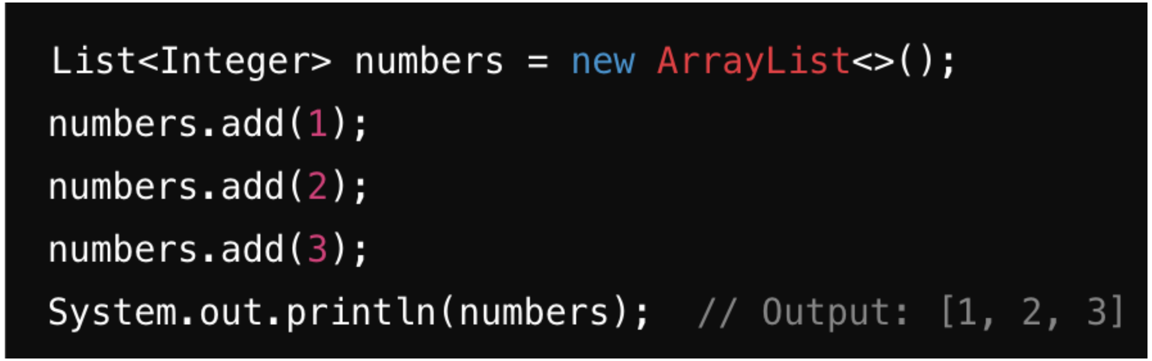
```
int[] numbers = new int[5];
```

```
int[] numbers = new int[]{1, 2, 3, 4, 5};
```

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

## Collections

- Both Java and C# have dynamic array-like collections



```
List<Integer> numbers = new ArrayList<>();
numbers.add(1);
numbers.add(2);
numbers.add(3);
System.out.println(numbers);  // Output: [1, 2, 3]
```

- *HashSet* (Unordered Collection, No Duplicates)
- HashMap (Key-Value Pair)
- LinkedList (Doubly-Linked List)

## Constants and Enums

- Constants in Java can be declared using the *final* keywords

```java
final int CONSTANT_VALUE = 100;
```

- Enum is a special type to define a set of constants (named values)

```java
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}
```

```java
Day today = Day.TUESDAY;
```

## Strings

- *String* in Java is a class, but in C# is a data type

```
String message = "Hello, World!";
String emptyString = new String();  // Creates an empty string
String fromChars = new String(new char[]{'H', 'e', 'l', 'l', 'o'});
```

- Length:

```
int length = message.length();
```

- Substring:

```
String sub = message.substring(7, 12);
```

- CharAt:

```
char letter = message.charAt(1);
```

- Contains:

```
boolean containsHello = message.contains("Hello");
```

## Exception Handling

- A mechanism to handle runtime errors and other exceptional conditions, preventing the program from crashing
- *try-catch-finally*

```
try {
    // Code that may throw an exception
} catch (ExceptionType e) {
    // Code to handle the exception
} finally {
    // Code that always executes
}
```

- *throw*

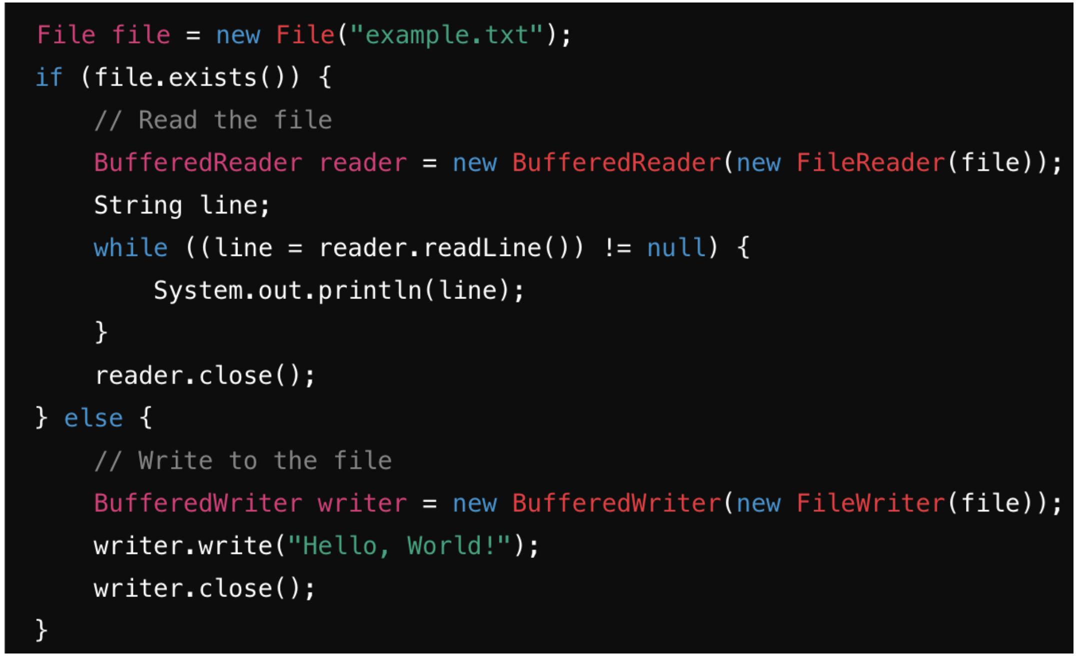
```
throw new ExceptionType("Error message");
```

- *throws*

```
public void methodName() throws ExceptionType {
    // Code that might throw an exception
}
```

# Files and I/O

- The *File* class is used to create, delete, and check file properties

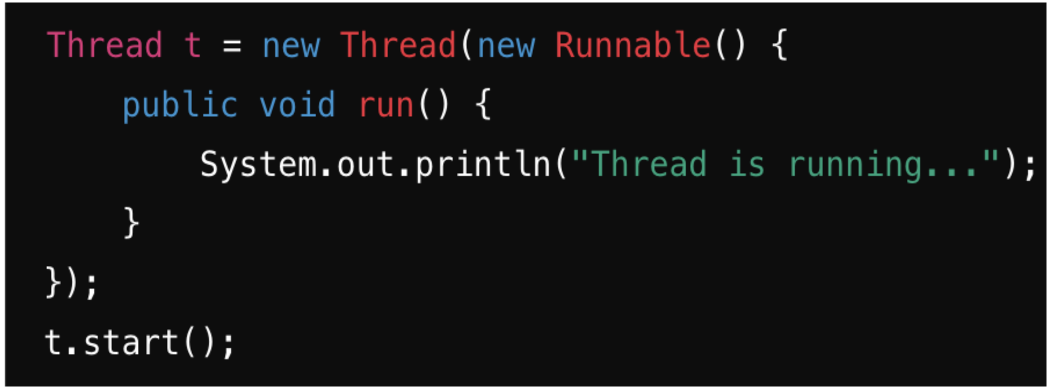


```java
File file = new File("example.txt");
if (file.exists()) {
    // Read the file
    BufferedReader reader = new BufferedReader(new FileReader(file));
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
    reader.close();
} else {
    // Write to the file
    BufferedWriter writer = new BufferedWriter(new FileWriter(file));
    writer.write("Hello, World!");
    writer.close();
}
```

- Files (in the *java.nio.file* package) provides a simple way

## Threads and Concurrency

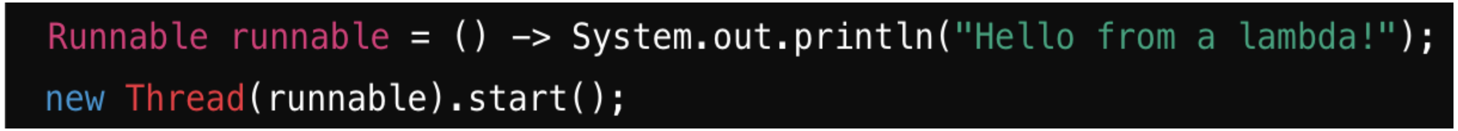
- A thread is a lightweight unit of execution within a process
- Allows programs to perform multiple tasks simultaneously (multithreading)
- Example: Web server handling multiple client requests at the same time



```java
Thread t = new Thread(new Runnable() {
    public void run() {
        System.out.println("Thread is running...");
    }
});
t.start();
```

## Lambda Expressions

- A feature introduced in Java 8 that allows writing concise and expressive code
- They enable you to pass behavior as parameters, write inline implementations, and reduce boilerplate code
- Syntax: `(parameters) -> expression`



```
Runnable runnable = () -> System.out.println("Hello from a lambda!");
new Thread(runnable).start();
```

## Generics

- Allow you to define classes, interfaces, and methods with type parameters
- E.g. You need a storage container that can hold various types of items, such as documents, tools, toys, etc.

```java
public class Box<T> {
    private T value;
    public void set(T value) { this.value = value; }
    public T get() { return value; }
}
```

```java
Box<String> stringBox = new Box<>();
Box<Integer> integerBox = new Box<>();
```

## GUI in Java

- **AWT** (Abstract Window Toolkit): the original Java GUI toolkit, providing basic components, layout managers, and event handling

- **Swing**: A more advanced and flexible GUI toolkit, offering a richer set of components and better support for look-and-feel customization

- **JavaFX**: The modern Java GUI framework designed for building rich client applications, with a more flexible and powerful UI toolkit

## Java Technologies

- **Web:** Servlets, JSP, and Spring Framework
- **Networking:** TCP/IP and UDP protocols using Sockets
- **URL:** For handling HTTP requests and responses
- **Databases:** JDBC (Java Database Connectivity), an API for connecting to and interacting with databases using SQL
- **Microservices:** Spring Boot and REST APIs
- **JUnit:** a testing framework for Java for creating automated tests
- **Apache Maven:** a build automation tool for Java projects
- **Android SDK** and **Android API**: for developing Android applications
- **And more..**