

Lecture 0 – Introduction

SWS121: Secure Programming

Jihyeok Park



2024 Spring

- **Instructor:** Jihyeok Park (박지혁)
 - **Position:** Assistant Professor in CS, Korea University
 - **Expertise:** Programming Languages, Software Analysis
 - **Office hours:** 14:00–16:00, Tuesdays (appointment by e-mail)
 - **Office:** 609A, Science Library Bldg
 - **Email:** jihyeok_park@korea.ac.kr

- **Instructor:** Jihyeok Park (박지혁)
 - **Position:** Assistant Professor in CS, Korea University
 - **Expertise:** Programming Languages, Software Analysis
 - **Office hours:** 14:00–16:00, Tuesdays (appointment by e-mail)
 - **Office:** 609A, Science Library Bldg
 - **Email:** jihyeok_park@korea.ac.kr
- **Class:** SWS121: Secure Programming

- **Instructor:** Jihyeok Park (박지혁)
 - **Position:** Assistant Professor in CS, Korea University
 - **Expertise:** Programming Languages, Software Analysis
 - **Office hours:** 14:00–16:00, Tuesdays (appointment by e-mail)
 - **Office:** 609A, Science Library Bldg
 - **Email:** jihyeok_park@korea.ac.kr
- **Class:** SWS121: Secure Programming
- **Lectures:** 18:45–20:15, Mon. @ 304 Aegineung (애기능생활관)

- **Instructor:** Jihyeok Park (박지혁)
 - **Position:** Assistant Professor in CS, Korea University
 - **Expertise:** Programming Languages, Software Analysis
 - **Office hours:** 14:00–16:00, Tuesdays (appointment by e-mail)
 - **Office:** 609A, Science Library Bldg
 - **Email:** jihyeok_park@korea.ac.kr
- **Class:** SWS121: Secure Programming
- **Lectures:** 18:45–20:15, Mon. @ 304 Aegineung (애기능생활관)
- **Homepage:** <https://plrg.korea.ac.kr/courses/sws121/>

Week	Date	Contents
1	03/04	Introduction
2	03/11	Basics
3	03/18	Testing and Documentation
4	03/25	Classes, Traits, and Objects
5	04/01	First-Class Functions
6	04/08	Packaging and Imports
7	04/15	Collections
8	04/22	Midterm Exam Week (No Class)
9	04/29	Pattern Matching
10	05/06	For Comprehensions
11	05/13	Polymorphism
12	05/20	Lazy Evaluation
13	05/27	Variances
14	06/03	Contextual Abstraction
15	06/10	Course Review
16	06/17	Final Exam Week (No Class)

- **Homework Assignments: 90%**
 - **3 Programming Assignments:**
 - Homework 1: 30% (due on April 15)
 - Homework 2: 30% (due on May 20)
 - Homework 3: 30% (due on June 17)
 - Submit your homework on **Blackboard**.
 - You can utilize or refer to any other materials (e.g., ChatGPT), but you **MUST** write your **OWN** solution.
 - **Cheating is strictly prohibited. Cheating will get you an F.**

- **Homework Assignments: 90%**
 - **3 Programming Assignments:**
 - Homework 1: 30% (due on April 15)
 - Homework 2: 30% (due on May 20)
 - Homework 3: 30% (due on June 17)
 - Submit your homework on **Blackboard**.
 - You can utilize or refer to any other materials (e.g., ChatGPT), but you **MUST** write your **OWN** solution.
 - **Cheating is strictly prohibited. Cheating will get you an F.**
- **Attendance: 10%**
 - Please use **Blackboard** to attend the class **by yourself**.

- **Self-contained lecture notes.**

<https://plrg.korea.ac.kr/courses/sws121/>

- **Reference**

- **“Tour of Scala”**

docs.scala-lang.org/tour/tour-of-scala.html

- **“Scala 3 Book”**

docs.scala-lang.org/scala3/book/introduction.html

- **“Scala 3 Reference”**

docs.scala-lang.org/scala3/reference/index.html

Secure Programming is a coding practice that ensures the software is designed to be secure and free from vulnerabilities.

Secure Programming is a coding practice that ensures the software is designed to be secure and free from vulnerabilities.

- **Static type checking**
 - Using the type system to catch bugs

Secure Programming is a coding practice that ensures the software is designed to be secure and free from vulnerabilities.

- **Static type checking**
 - Using the type system to catch bugs
- **Test-driven development (TDD)**
 - Writing tests before writing the code

Secure Programming is a coding practice that ensures the software is designed to be secure and free from vulnerabilities.

- **Static type checking**
 - Using the type system to catch bugs
- **Test-driven development (TDD)**
 - Writing tests before writing the code
- **Documentation**
 - Writing clear and concise comments

Secure Programming is a coding practice that ensures the software is designed to be secure and free from vulnerabilities.

- **Static type checking**
 - Using the type system to catch bugs
- **Test-driven development (TDD)**
 - Writing tests before writing the code
- **Documentation**
 - Writing clear and concise comments
- **Encapsulation**
 - Hiding the implementation details

Secure Programming is a coding practice that ensures the software is designed to be secure and free from vulnerabilities.

- **Static type checking**
 - Using the type system to catch bugs
- **Test-driven development (TDD)**
 - Writing tests before writing the code
- **Documentation**
 - Writing clear and concise comments
- **Encapsulation**
 - Hiding the implementation details
- **Defensive programming**
 - Writing code to handle unexpected inputs



Scala stands for **Scalable Language**.



Scala stands for **Scalable Language**.

- A **more concise** version of Java with **advanced features**



Scala stands for **Scalable Language**.

- A **more concise** version of Java with **advanced features**
- A general-purpose programming language



Scala stands for **Scalable Language**.

- A **more concise** version of Java with **advanced features**
- A general-purpose programming language
- **Java Virtual Machine (JVM)**-based language



Scala stands for **Scalable Language**.

- A **more concise** version of Java with **advanced features**
- A general-purpose programming language
- **Java Virtual Machine (JVM)**-based language
- A **statically typed** language



Scala stands for **Scalable Language**.

- A **more concise** version of Java with **advanced features**
- A general-purpose programming language
- **Java Virtual Machine (JVM)**-based language
- A **statically typed** language
- A **object-oriented programming (OOP)** language



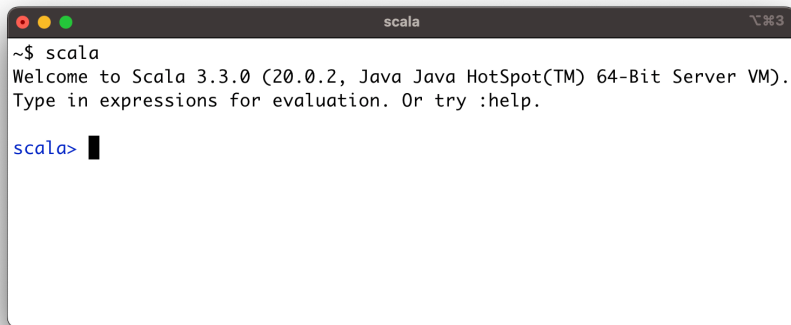
Scala stands for **Scalable Language**.

- A **more concise** version of Java with **advanced features**
- A general-purpose programming language
- **Java Virtual Machine (JVM)**-based language
- A **statically typed** language
- A **object-oriented programming (OOP)** language
- A **functional programming (FP)** language

Read-Eval-Print-Loop (REPL)

Please download and install them using the following links.

- **JDK >= 8** – www.oracle.com/java/technologies/downloads/
- **sbt** – www.scala-sbt.org/download.html
- **Scala REPL** – www.scala-lang.org/download/



A screenshot of a terminal window titled "scala" with a dark title bar and standard macOS window controls (red, yellow, green buttons). The terminal content shows the command `~$ scala` being executed, followed by the Scala REPL welcome message: "Welcome to Scala 3.3.0 (20.0.2, Java Java HotSpot(TM) 64-Bit Server VM). Type in expressions for evaluation. Or try :help." Below this, the prompt `scala>` is shown with a black cursor.

```
~$ scala
Welcome to Scala 3.3.0 (20.0.2, Java Java HotSpot(TM) 64-Bit Server VM).
Type in expressions for evaluation. Or try :help.

scala> █
```

- Basics

Jihyeok Park
jihyeok_park@korea.ac.kr
<https://plrg.korea.ac.kr>