Lecture 0 – Introduction

SWS121: Secure Programming

Jihyeok Park

2024 Spring

Course Information

- **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

Course Information

- **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

Course Information

- **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)

Course Information

- **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/

Schedule

PLRG

Weak	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)

Grading

- **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.

Grading

PLRG

- **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**.

Course Materials

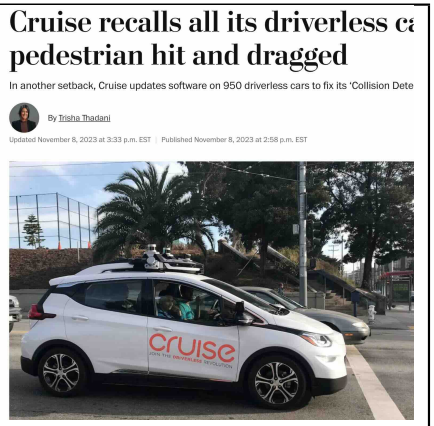
- **Self-contained lecture notes.**

  `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`

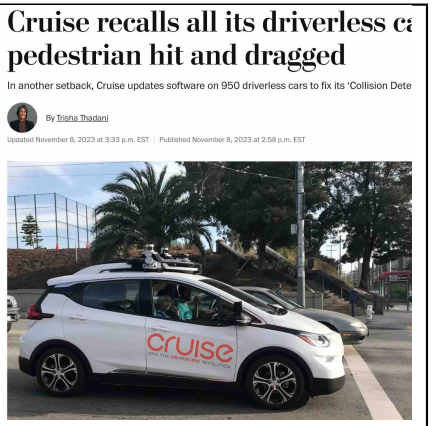
Errors in Safety-Critical Software

Unexpected faults in **safety-critical software** cause serious problems:

			
Rocket	**Financial**	**Airport**	**Auto. Vehicle**
(1996)	(2012)	(2020)	(2023)

Errors in Safety-Critical Software

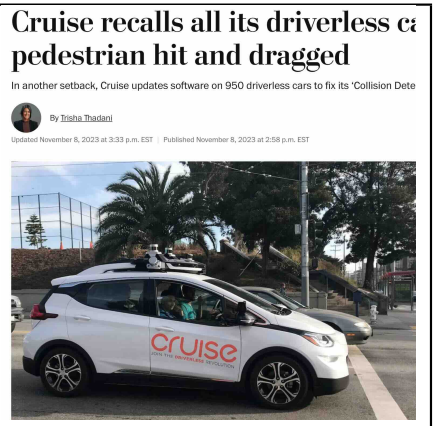
Unexpected faults in **safety-critical software** cause serious problems:

			
Rocket	**Financial**	**Airport**	**Auto. Vehicle**
(1996)	(2012)	(2020)	(2023)

Then, how can we **prevent** such software faults?

Errors in Safety-Critical Software

Unexpected faults in **safety-critical software** cause serious problems:

			
Rocket	**Financial**	**Airport**	**Auto. Vehicle**
(1996)	(2012)	(2020)	(2023)

Then, how can we **prevent** such software faults?

Let's learn **secure programming** to write **safe** and **reliable** software with **Scala**.

Secure Programming

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

Secure Programming

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

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

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

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

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

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

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

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

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

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

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

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

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

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

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

What is Scala?

PLRG

Scala stands for **Sca**lable **La**nguage.

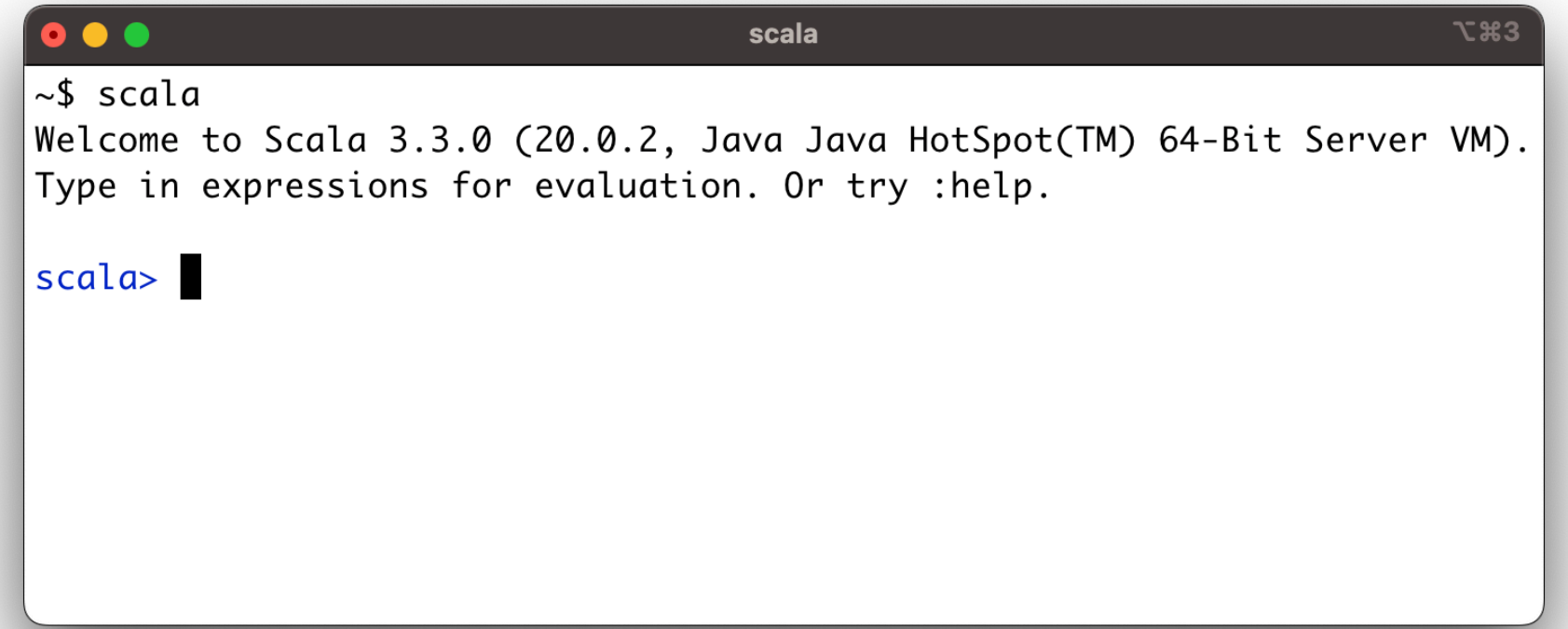
- A **more concise** version of Java with **advanced features**
- A general-purpose programming language
- **Java Virtual Machine (JVM)**-based language
- A **statically typed** language
- An **object-oriented programming (OOP)** language
- A **functional programming (FP)** language

Read-Eval-Print-Loop (REPL)

PLRG

Please download and install them using the following links.

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



Next Lecture

- Basics

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