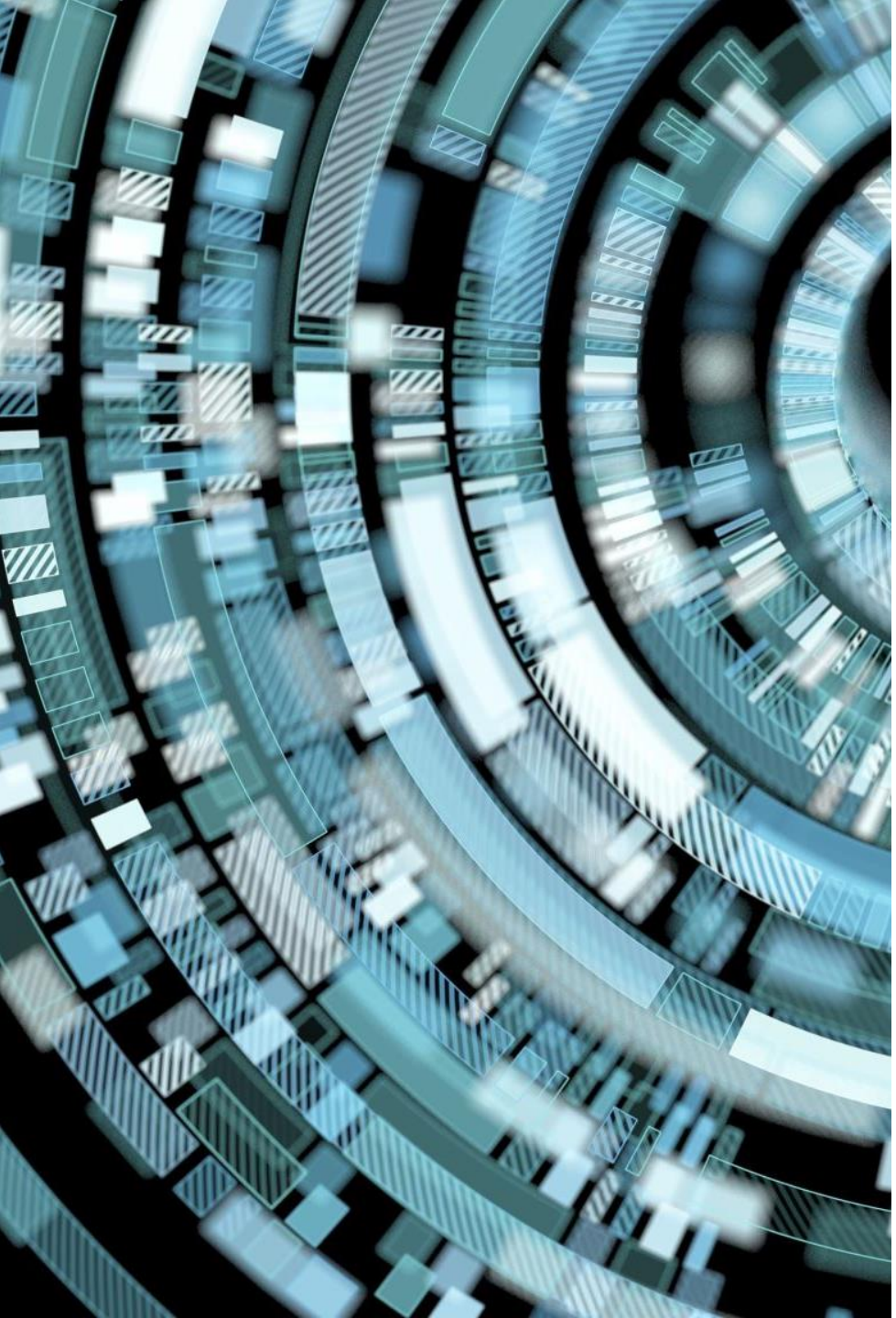
Computer System Design & Application

计算机系统设计与应用A

陶伊达 （TAO Yida）
taoyd@sustech.edu.cn



Lecture 15

- Project Demo
- Grading Policy
- Course Review

Project Demo

- 伍福临、刘家宝
- 钟志源、李华炫
- 陶毅诚、秦尧

Grading Policy

	Score	Description
Assignments	25%	2 assignments Assignment 1: release at week 4 and due at week 7 Assignment 2: release at week 8 and due at week 11
Project	20%	Released around week 9 Team: Preferably 2 people +1 for submitting the final project at week 15 +1 (max) for presenting at week 16 lecture
Labs	15%	Attendance Lab practices (+0.1 points for submitting lab practice onsite, max +1)
Feedback	4%	Submitting feedback at the end of each lab
Quiz	6%	Quizzes, exercises, participation during lectures
Final Exam	30%	Close-book (Two pieces of A4 cheat sheets allowed) No electronic device

Disclaimer

Course content that does not appear in the slides may still be tested in the final exam

Topics covered

Applications

- Data analytics and visualization
- C/S applications (e.g., FTP)
- Text scraping and processing
- Web applications & REST services

Principles

- OOP, AOP
- Functional programming
- Design principles
- Reusable software
- JVM

Utilities

- Generic collections
- Lambdas & Stream
- Exception handling
- Files & I/O
- Annotations
- Reflection
- JUnit Testing
- Logging

Functionalities

- GUI & JavaFX
- Networking
- Multithreading
- Web development
- Web services

Topics covered

Applications

- Data analytics and visualization
- C/S applications (e.g., FTP)
- Text scraping and processing
- Web applications & REST services

Principles

- OOP, AOP
- Functional programming
- Design principles
- Reusable software
- JVM

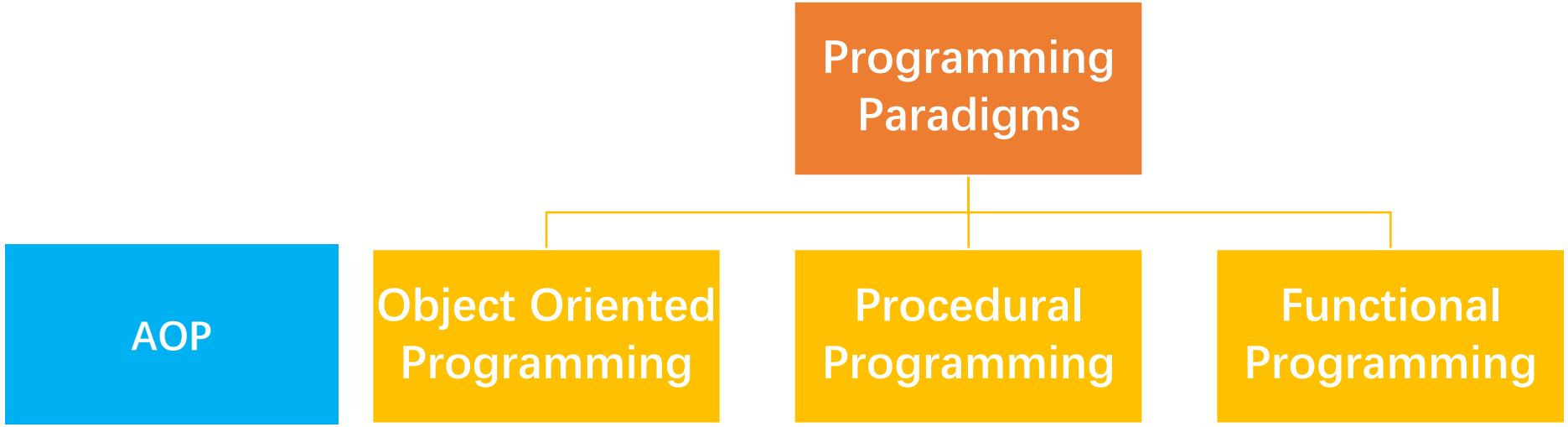
Utilities

- Generic collections
- Lambdas & Stream
- Exception handling
- Files & I/O
- Annotations
- Reflection
- JUnit Testing
- Logging

Functionalities

- GUI & JavaFX
- Networking
- Multithreading
- Web development
- Web services

Programming Paradigms



- Business logics
- Cross-cutting concerns

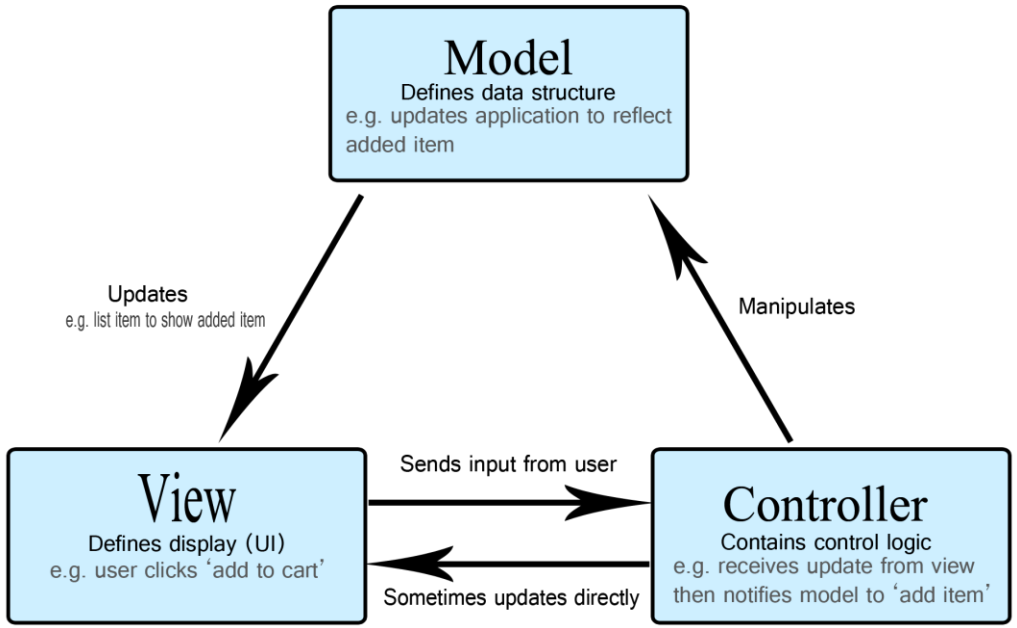
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

- First-class functions
- Higher-order functions
- No side effects

Design Principles

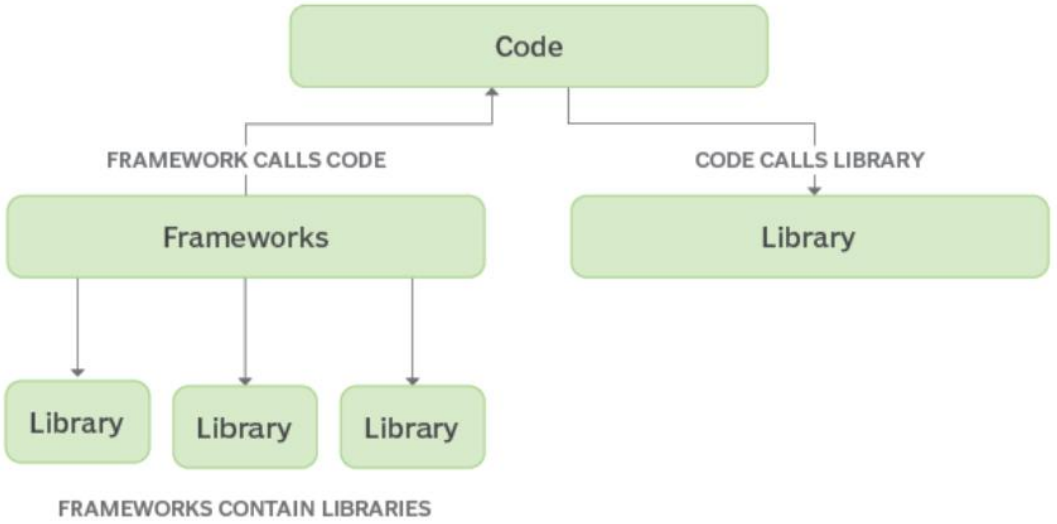
Software Design Principles

- High Cohesion (高内聚)
- Low Coupling (低耦合)
- Information Hiding (信息隐藏)

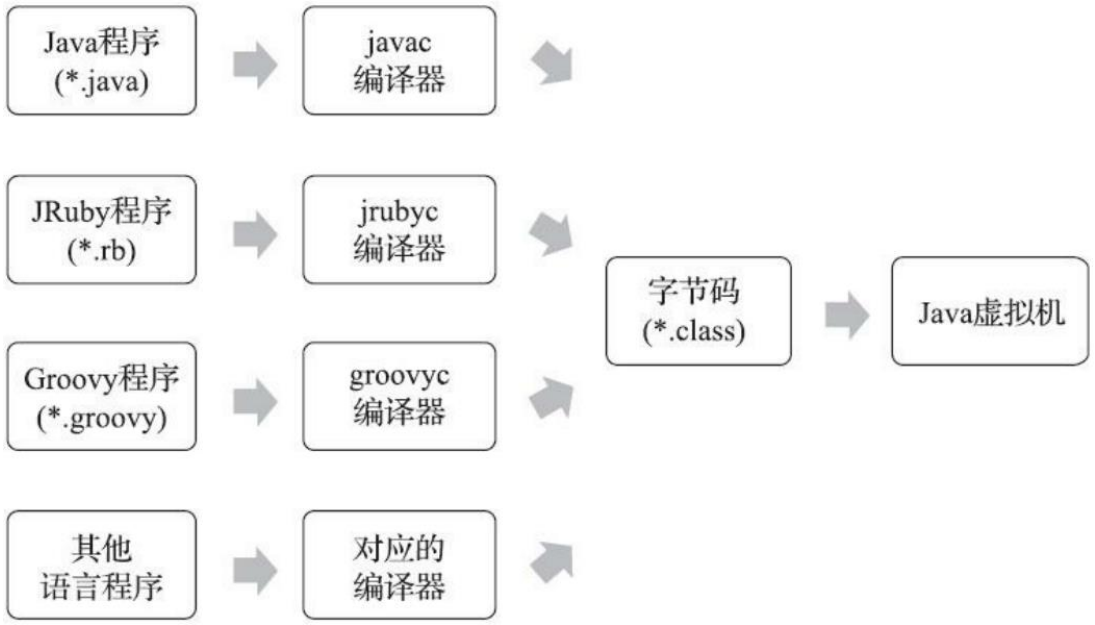


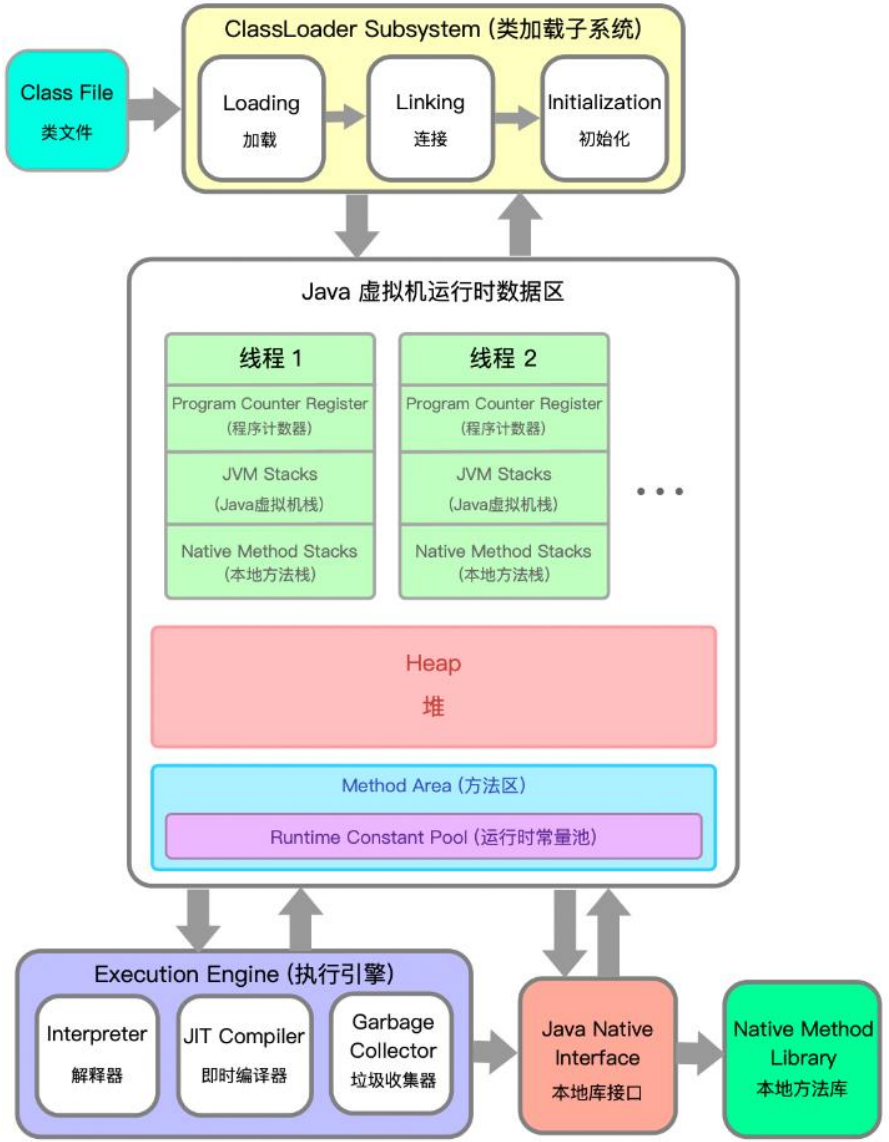
Reusable Software

- Inversion of Control (IoC, 控制反转): a principle in SE which transfers the control of objects or portions of a program to a container or framework
- Traditionally, our custom code makes calls to a library; In contrast, IoC enables a framework to take control of the flow of a program and make calls to our custom code.
- To use a framework, you need to insert your behavior into various places in the framework either by subclassing or by plugging in your own classes. The framework's code then calls your code at these points.



Bytecode & JVM





Topics covered

Applications

- Data analytics and visualization
- C/S applications (e.g., FTP)
- Text scraping and processing
- Web applications & REST services

Principles

- OOP, AOP
- Functional programming
- Design principles
- Reusable software
- JVM

Utilities

- Generic collections
- Lambdas & Stream
- Exception handling
- Files & I/O
- Annotations
- Reflection
- JUnit Testing
- Logging

Functionalities

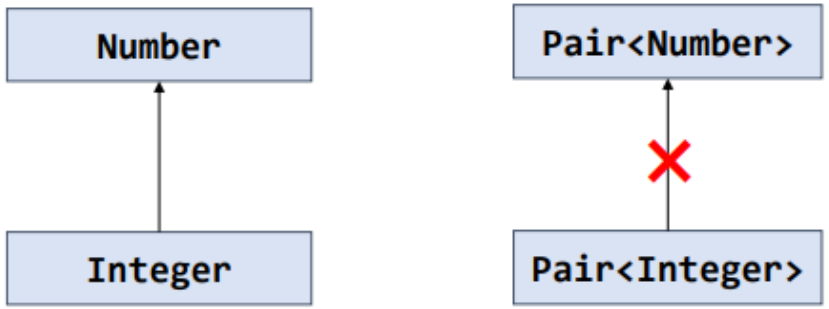
- GUI & JavaFX
- Networking
- Multithreading
- Web development
- Web services

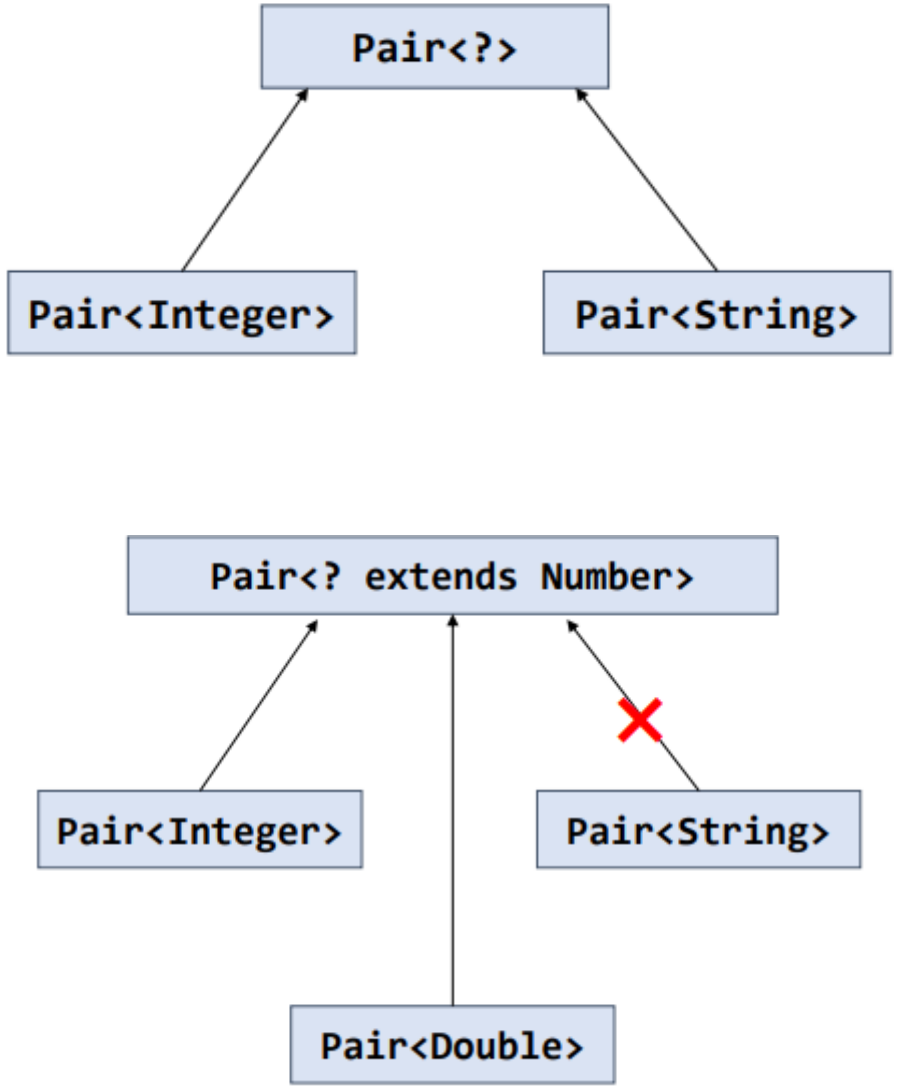
Generics

- Motivation
- Syntax & Usages
  - Generic Classes
  - Generic Interfaces
  - Generic Methods
- Type erasure

Generics

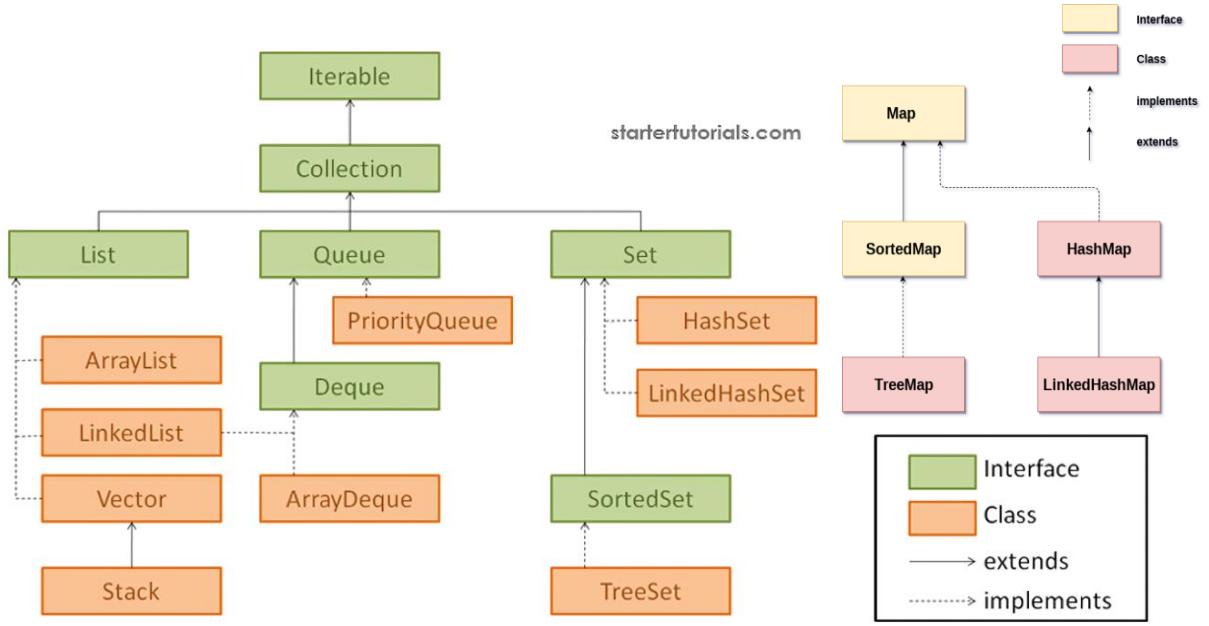
- Inheritance Rules
- Bounded Type Variables
- Wildcards





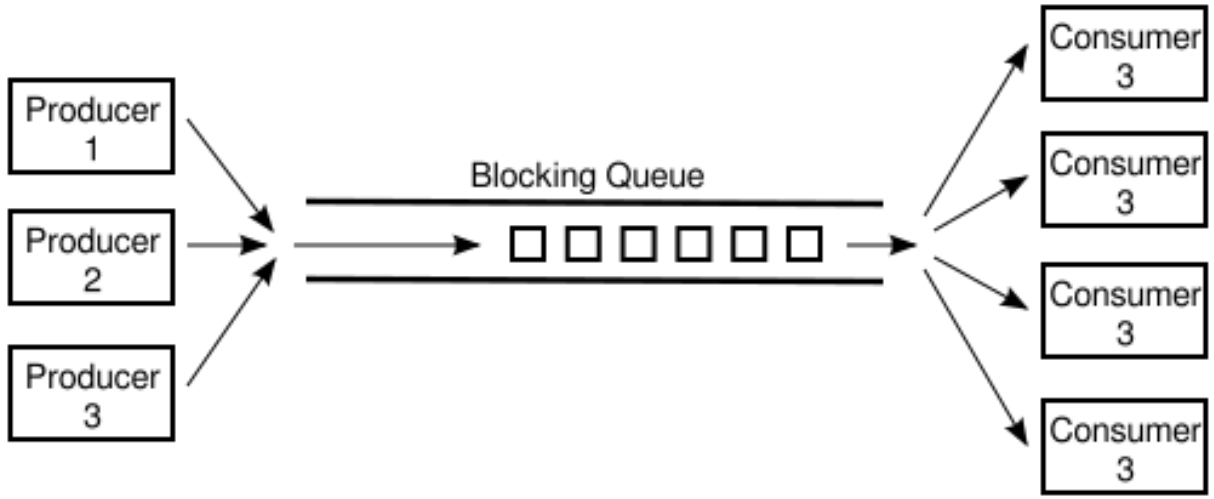
Collections

- Class hierarchy
- The Iterable<T> and Iterator<T> interfaces
- Commonly used collection implementations & characteristics
- Comparisons between different implementations



Collections

- Non thread-safe collections
- Thread-safe collections
  - Copy-on-Write collections
  - Compare-and-Swap collections (CAS)
  - Collections using Lock

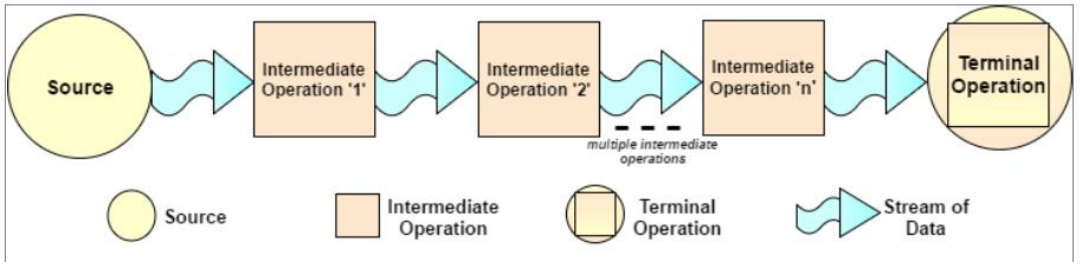


Lambda Expressions

- Lambda syntax
- Type inference
- Method references
  - Static method
  - Instance method (Bound)
  - Instance method (Unbound)
  - Constructor
- Common functional interfaces

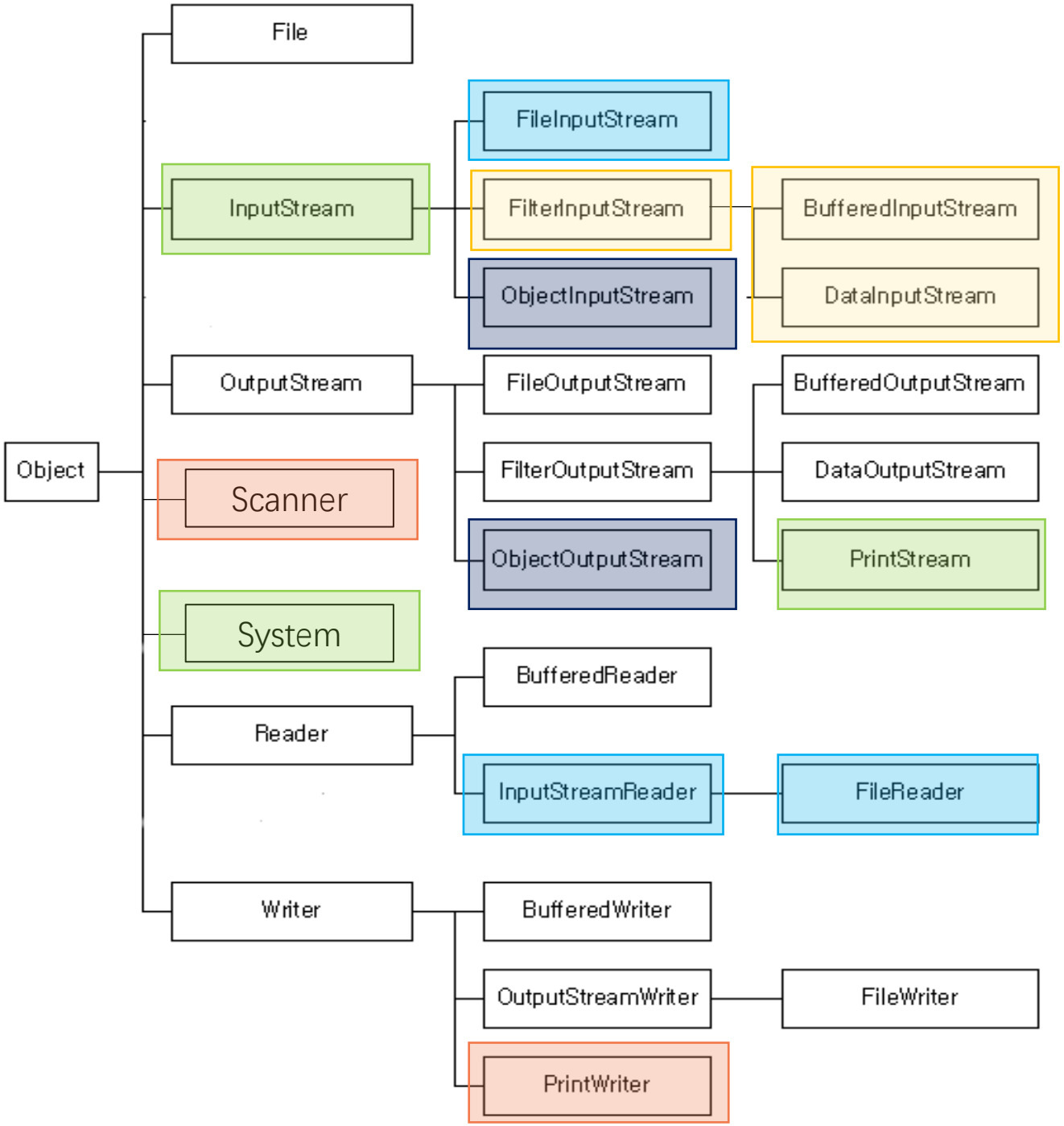
Stream API

- How to create a stream
- Common intermediate operations
- Common terminal operations
- Write a stream pipeline/chain
- The Optional<T> class



Exception Handling

- Exception class hierarchy
- Checked vs unchecked exceptions
- Syntax & Control flow
  - Try-catch, finally, throw



I/O

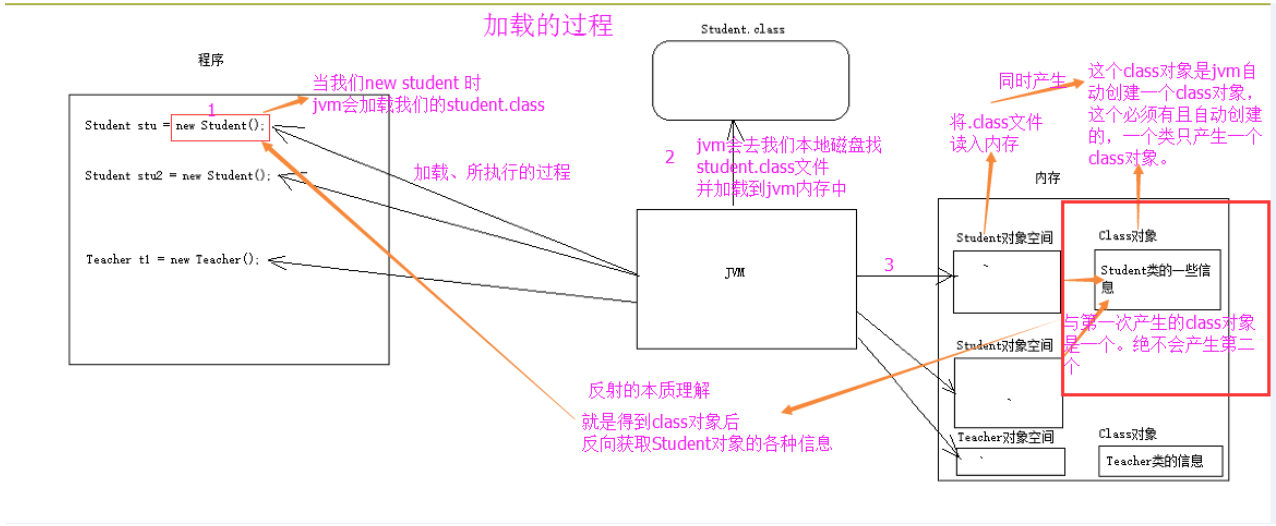
- Class hierarchy
- Byte streams vs. character streams & conversions
- Commonly used classes and methods
- Basic knowledge of character encoding & encoding schemes
- I/O from command line

Files

- Serialization & deserialization
- Absolute path & relative path
- Dot notations
- Basic operations for files and directories (e.g., traverse a directory recursively)

Reflection

- JVM class loading process
- Getting the `Class` object
- Examining fields and methods of a class
- Instantiating a class
- Invoking a method of an object



Annotations

- Built-in annotations
  - `@Deprecated`
  - `@Override`
  - `@SuppressWarnings`
  - `@FunctionalInterface`
  - `@SafeVarargs`
- Meta-annotations
  - `@Target`
  - `@Retention`
  - `@Documented`
  - `@Inherited`
  - `@Repeatable`

JUnit Testing

- Test classes and test methods
- Lifecycle methods
- Test instance lifecycle
- Assertions & assumptions

Logging

- Requirements for logging
- Design of Apache Log4j
- Different log levels and meanings

Topics covered

Applications

- Data analytics and visualization
- C/S applications (e.g., FTP)
- Text scraping and processing
- Web applications & REST services

Principles

- OOP, AOP
- Functional programming
- Design patterns
- Reusable software

Utilities

- Generic collections
- Lambdas & Stream
- Exception handling
- Files & I/O
- Annotations
- Reflection
- JUnit Testing
- Logging

Functionalities

- GUI & JavaFX
- Networking
- Multithreading
- Web development
- Web services

JavaFX

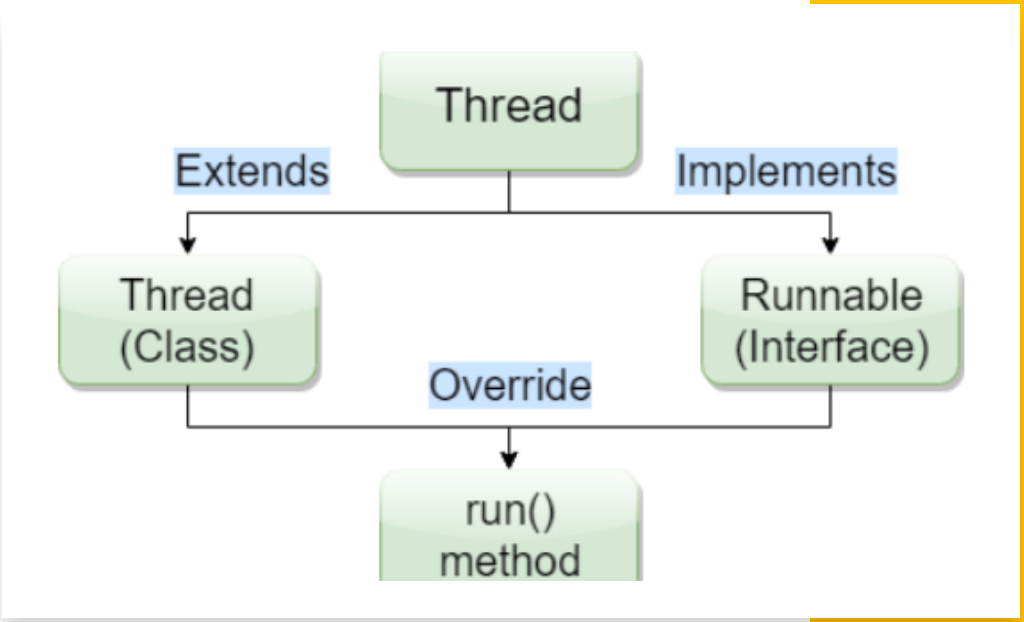
- Basic concepts
  - Stage, scene, scene graph, node
  - Panes, controls, charts

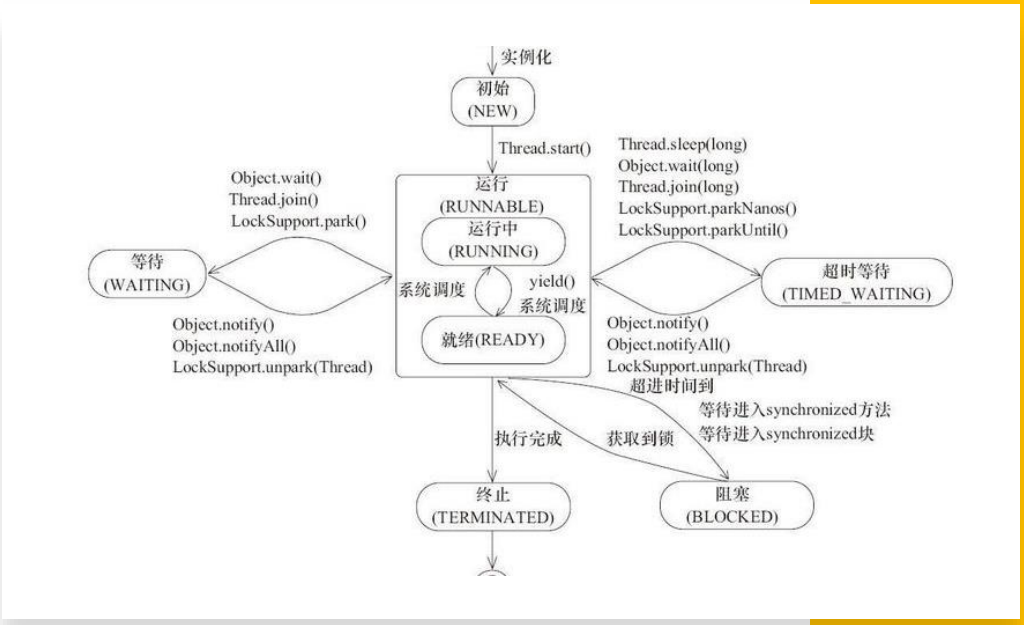
Networking

- Socket concepts
- Protocols
- Using socket to implement basic client & server
- Reading from and writing to a socket (sending requests & getting responses)

Concurrency

- Creating & Starting Threads
- Thread States





Concurrency

- Thread Safety
- Synchronization
  - The synchronized, volatile keyword
  - The Concurrency API (ReentrantLock, Condition)
- Task execution & Thread pools

Java EE

- Java EE multitiered model
- Servlet & Containers
  - HttpServlet
  - Servlet lifecycle
- Common Java EE specifications and corresponding implementations

Java EE

- JDBC
- ORM
- JPA & Hibernate

The Spring Framework

- IoC & Dependency Injection (typical annotations)
- Spring AOP concepts and terminology
- Spring MVC basic workflow

Spring Boot

- Basic concepts (convention over configuration)
- Workflow architecture
- Building a MVC web application
- Building a RESTful web service

- Q&A
- Teaching Evaluation

方法及步骤

1. 网页端：登录教务系统：https://tis.sustech.edu.cn/ -业务办理-评教任务-2023秋季学期学生评价任务。系统按课程类型设置评价任务（理论类、实验实践类、体育类、艺术类），如页面上有多个评价任务，请逐一进入并提交评价。

2. 微信端：通过微信进入“南方科技大学”微信企业号--教学质量管理平台，在“我的任务-待评”中填写并提交本学期所选课程的所有听课评价。

操作指南请扫描下方二维码获取：

Thank You & Good Luck!