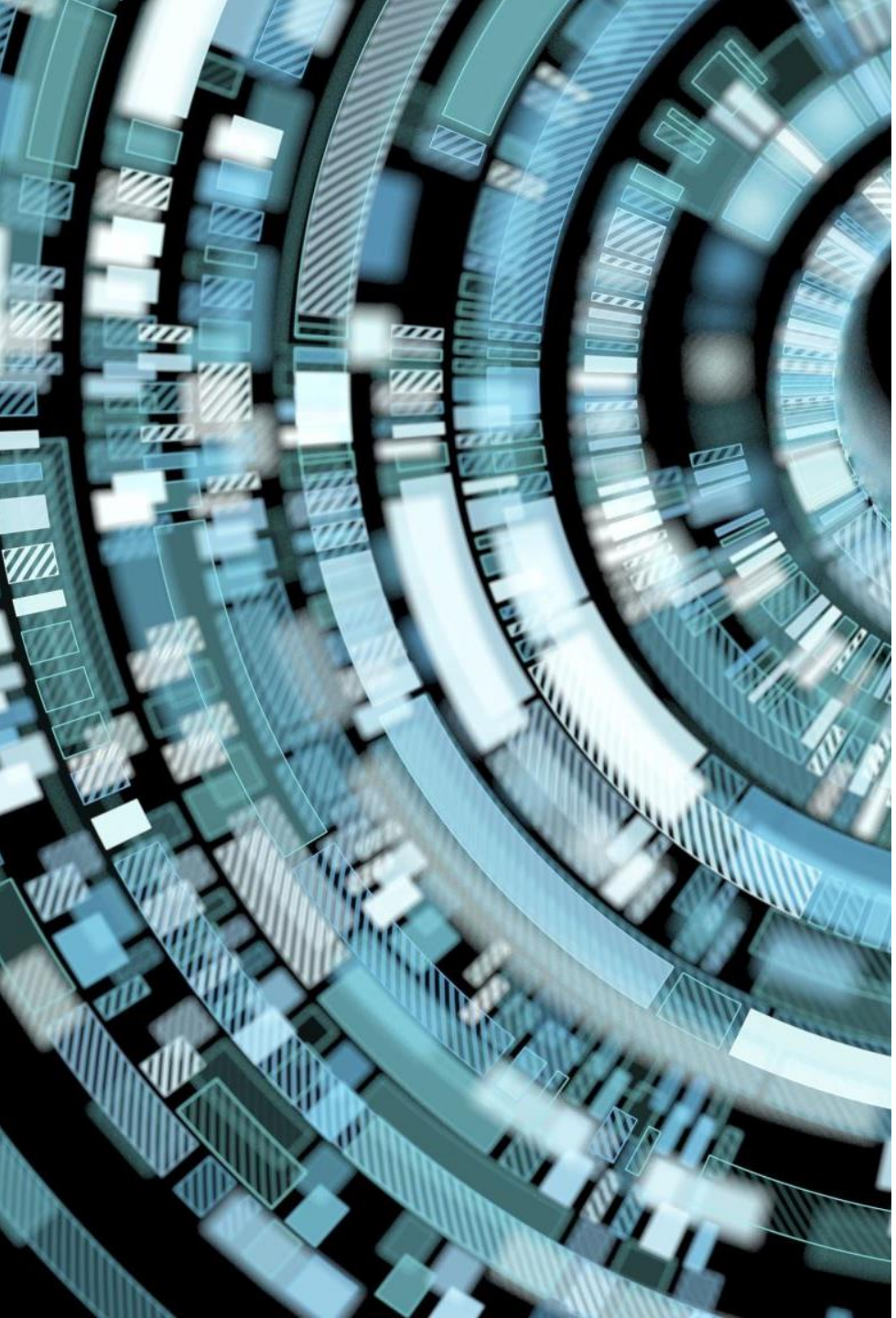
Computer System Design & Application

计算机系统设计与应用A

陶伊达 （TAO Yida）
taoyd@sustech.edu.cn



Lecture 15

- Grading Policy
- Course Review

Grading Policy

	Score	Description
Labs	15%	Attendance Lab practices (+0.1 points for submitting lab practice onsite, max +1)
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 no later than week 8 Team: Preferably 2 people +1 for submitting the final project at week 15 +2 (max) for presenting at week 16 lecture
Standardization	4%	Version control (git) Coding styles / coding convention
Quiz	6%	Quizzes during lectures
Final Exam	30%	Close-book (Two pieces of A4 cheat sheets allowed, handwritten) No electronic device

Topics covered

Applications

- Data analytics and visualization
- Text scraping and processing
- Web applications

Principles

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

Utilities

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

Functionalities

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

Topics covered

Applications

- Data analytics and visualization
- Text scraping and processing
- Web applications

Principles

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

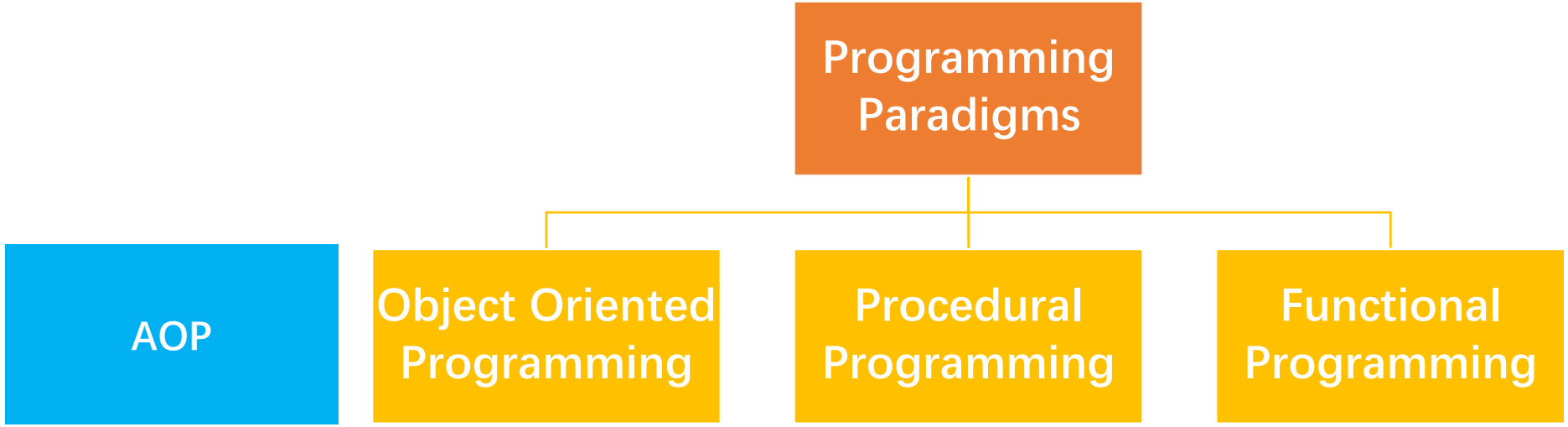
Utilities

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

Functionalities

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

Programming Paradigms



- Business logics
- Cross-cutting concerns

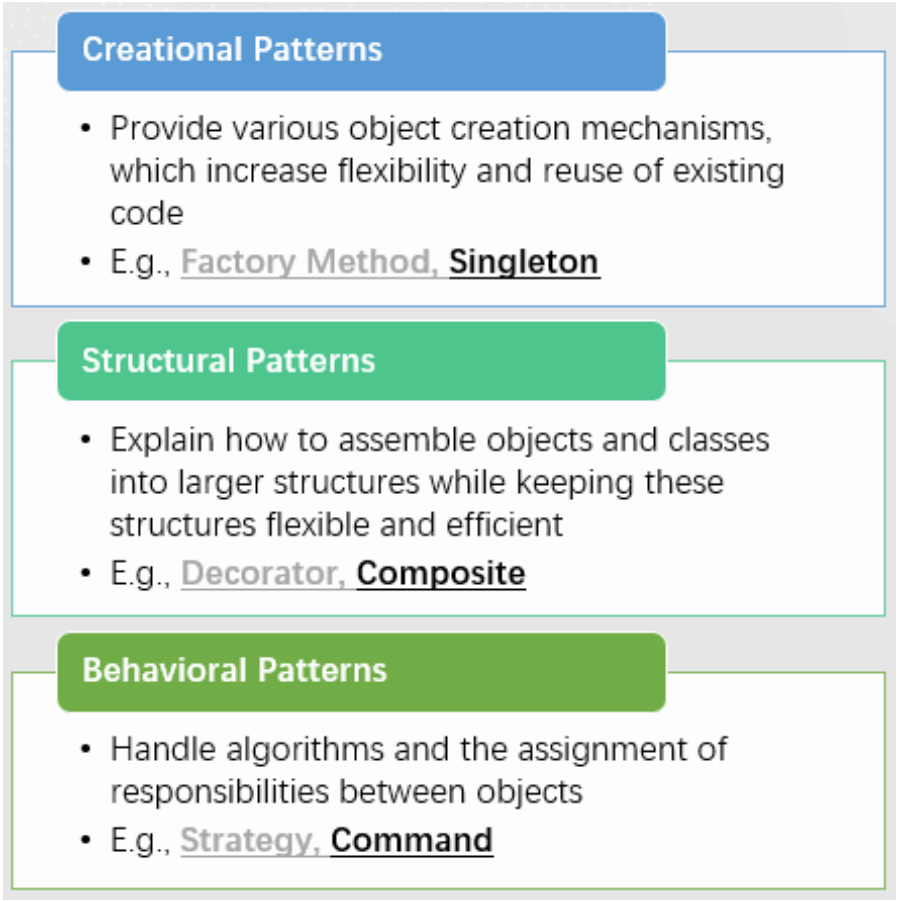
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

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

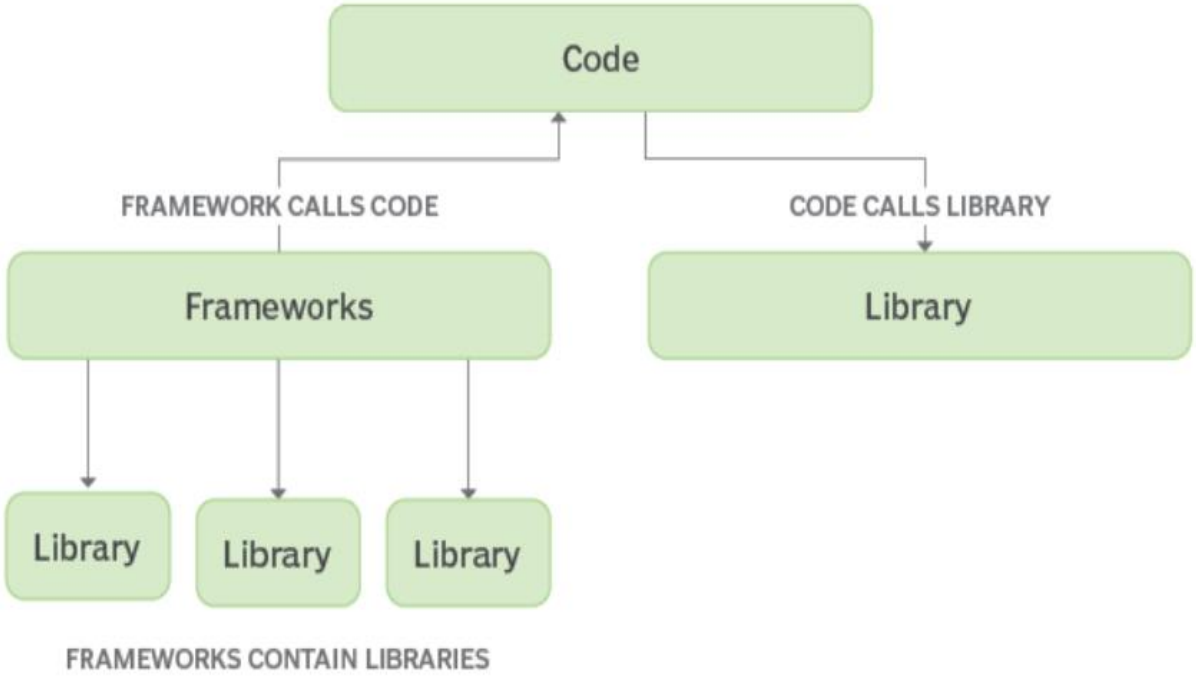
Software Design Principles & Design Patterns

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.

Topics covered

Applications

- Data analytics and visualization
- Text scraping and processing
- Web applications

Principles

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

Utilities

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

Functionalities

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

Generics

- Motivation
- Syntax & Usages
  - Generic Classes
  - Generic Interfaces
  - Generic Methods
- Inheritance Rules
- Bounded Type Variables & Wildcards
- Type erasure

Collections

- Commonly used collections & characteristics
- Comparisons between similar collections
- The Iterator<T> and Comparator<T> interfaces
- Concurrent collections
  - Copy-on-Write collections
  - Compare-and-Swap collections (CAS)
  - Collections using Lock

HashSet	LinkedHashSet	TreeSet
HashSet internally uses HashMap to store its elements.	LinkedHashSet internally uses LinkedHashMap to store its elements.	TreeSet internally uses TreeMap to store its elements.
HashSet doesn't maintain any order of elements.	LinkedHashSet maintain insertion order of elements.	TreeSet maintains default natural sorting order.
HashSet gives better performance than LinkedHashSet and TreeSet.	The performance of LinkedHashSet is between HashSet and TreeSet.	The TreeSet gives less performance than HashSet and LinkedHashSet.
HashSet allow maximum one null element.	LinkedHashSet also allow maximum one null element.	The TreeSet doesn't allow even single element.

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 & Files

- I/O Streams
  - Class hierarchy
  - Byte streams vs. character streams & conversions
  - Commonly used classes and methods
  - Basic knowledge of character encoding
- Files
  - Serialization
  - Path
  - Basic operations for files and directories

Annotations

- Built-in annotations
- Meta-annotations
- Declaring custom annotations

Reflection

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

JUnit Testing

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

Topics covered

Applications

- Data analytics and visualization
- Text scraping and processing
- Web applications

Principles

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

Utilities

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

Functionalities

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

JavaFX

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

Networking

- Socket concepts
- Using socket to implement basic client & server
- Reading from and writing to a socket

Multithreading

- Creating & Starting Threads
- Thread States
- Thread Safety & Synchronization
- Concurrent Collections

Java EE

- Java EE multitiered model
- Servlet & Containers
- JDBC, ORM, JPA

Spring

- The Spring Framework
  - IoC & Dependency Injection
  - Spring AOP
  - Spring MVC
- Spring Boot
  - Workflow
  - Building a MVC web application
  - Building a RESTful web service

Thank You!

Take Care and
Good Luck!