# Object-Otiented

Chen Yuheng

# Software Engineering

systematic, disciplined, qualifiable approach to development, operation, maintainance of software

- making things
- useful
- fashioning objects
- always learning
- tractable

**PROS**

- must perfect
- not by ourselves
- tedious
- always testing
- outdating

**CONS**

# RCS: Basics & Git

RCS: Revision Control Software

Revision: also known as version,
a state of a piece of info at a specific time
as a result of some change to it

- track history and evolution
- easier to collaborate
- recover from mistakes
- multiple versions support

Repository: database of the history being tracked

Initializationg: `git init`

Stage: prepare files to commit, ~~otherwise ignore~~

Commit: saving current state to history

Ignore: .gitignore: add the file name

diff: see changes

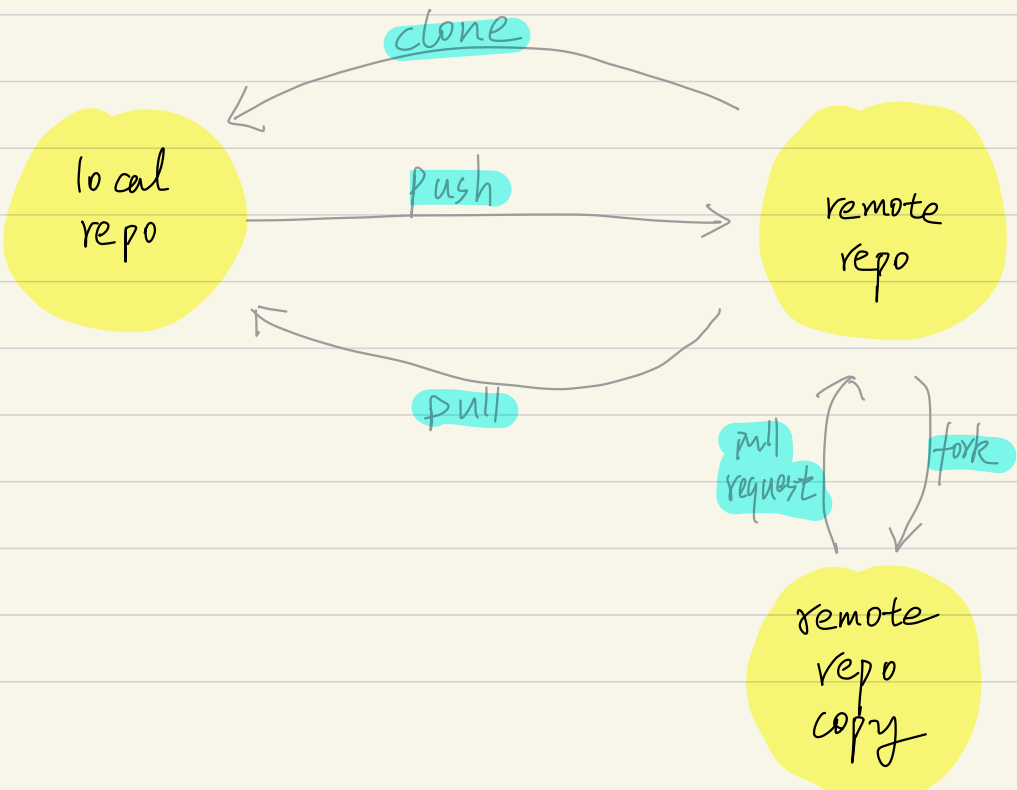
checkout: restore a historical state 查看历史状态.

HEAD label for current check out commit

tag: `git tag -a v1.0`

stash: 存放，暂存某些更改于 stash 中，然后可以撤销当前版本的更改做别的事

# REMOTE COMMUNICATION



# IDEs Integrated Development Environment

- source code editor
- compiler / interpreter
- debugger
- other tools

# OOP object-Oriented Programming 面向对象

Programming language: → datas
→ operations on datas

| Programming Paradigm | Examples |
| --- | --- |
| OO | Java, Python, JS |
| Procedural | C |
| Functional | F#, Haskel, Scala |
| Logic | Prolog |

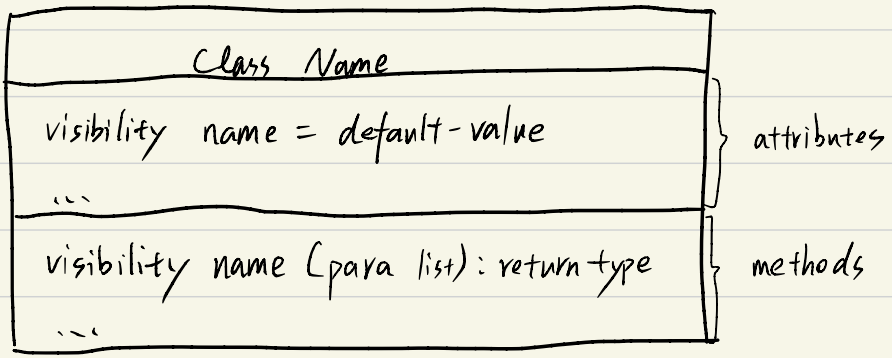
- OOP views the world as a network of interacting objects
- Every object has an interface & an implementation
- Objects interact by sending messages

A class contains instructions to creating an object
Object is abstraction from the real world

Encapsulation protects implementation from unintended access
- package to a self-contained unit
- hiding informations

# UML: Basic Structure Diagram

## Class Diagram

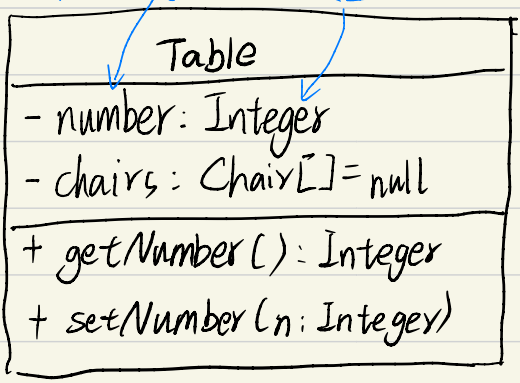



visibility:

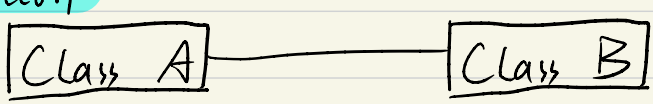
- \+ public
- \- private
- # protected
- ~ package private

An example

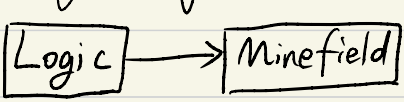
先写名字，再写类型



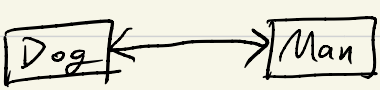

## Association



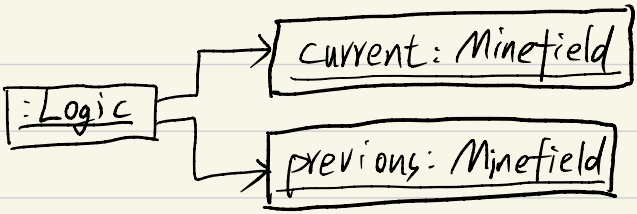

## Navigability: awareness




逻辑知道雷区

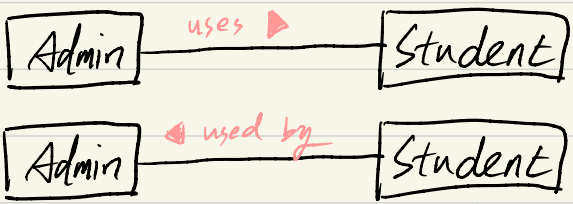



人与狗相互知道

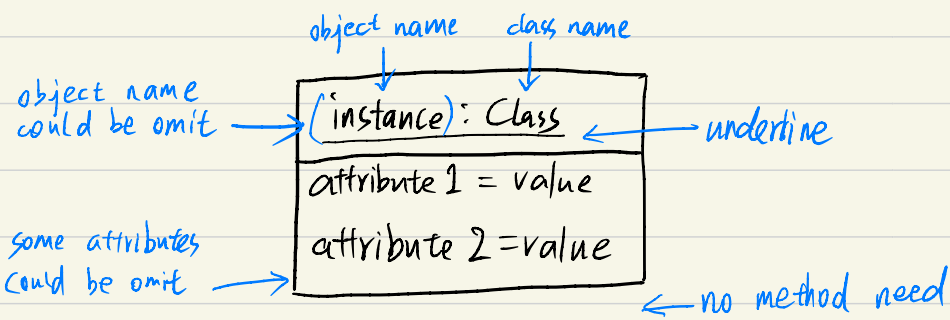



Not only for classes
also for objects

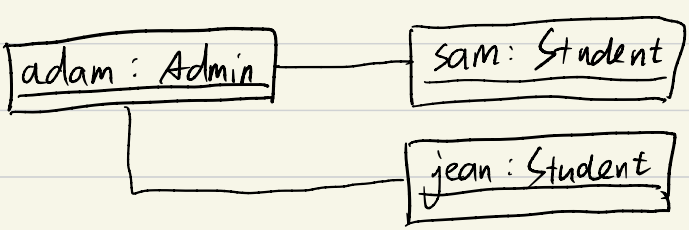
## Labels 注释用




## Object Diagram




## Associations



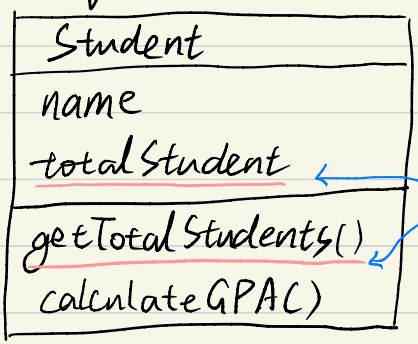
# Java

## Class-Level Members

Also: static ...

- class-level attributes ...
- class-level methods ...

In class diagram, underline them: (also example)




- use static final to define constant

## Enumerations

enum

## Varargs

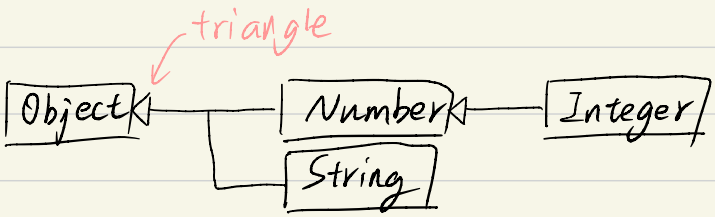
methodName ( Type ... name)

# OOP: Inheritance 繼承

Base Class : parent class, super class

↑

Derived class : child class, sub class, extended class




"is a" relationships

Multiple inheritance is available in C++, Python, but not supported by Java

# Exception

- checked exceptions
- errors
- runtime exceptions

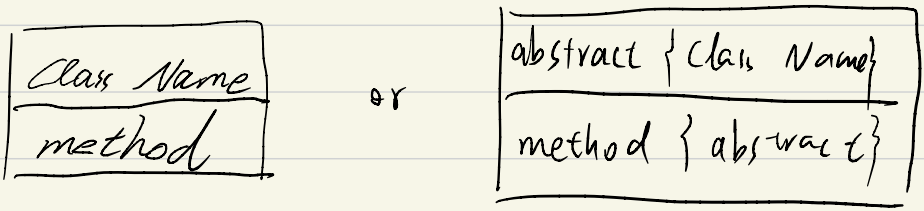
} unchecked exceptions

# Logging

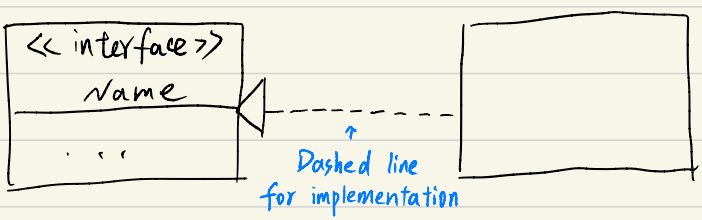
# Assertion

# Polymophism 多态

## Abstract Class Diagram




## Interfaces Diagram




An Interface can:
- used as a type
- inherited (multiple)
- contain constant and static method

# Generics 泛型

`public class Name<T1, T2 ...> {...}`

By convention, single, Uppercase.

like:
- E: element
- K: Key
- N: number
- T: type
- V: value
- S, U, V: 2nd, 3rd, 4th types.

# Collection

Interfaces of collection:

- Collection
- Set (unique)
- List (not unique)
- Queue
- Map

ArrayList

HashMap

# RCS : Workflow

## Branching

Some simple operations .....

## Merge Conflicts

Simply solve it by modifying the conflict file, it might cause some problems

## Pull Request

## Workflows

RCS can be done centralized (CRCS) or distributed (DRCS)

- Feature Branch flow
- Centralized flow

# Regression Test

## Software testing

- Dynamic
- Finite
- Selected
- Expected

## Regression Test

Test after every small steps

# Requirement

functional　non-functional

## Analysis

- Prioritizing analysis
- Requirement quality

## Gathering

- Brainstorming
- Product surveys
- Observation
- User surveys
- Interviews
- Focus groups

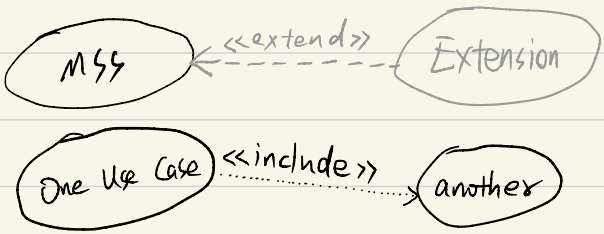
## Specifying

User stories 用户故事: short description from the perspective of User

Use case 用例: a sequence of description about operation

↑ needs an actor　The exercise could be reviewed

Main Success Senario (MSS) v.s. Extentions




- Feature list
- Prose
- Prototype
- Glossary
- Supplymentary Requirement

# Documentation

**Javadoc**: tool for generating API documentation

```
/**
 * ......
 *@XXX  ......
 */
```

**Markdown**: very easy

**AsciiDoc**: similar to MD, more complex

# Developer Testing

Testing done by the developers themselves

finding bugs earlier makes sense

## Test Automation

JUnit

# Continuos Integration

integration: combining parts to a whole 集成

# Integration Approaches

late and one-time, early and frequent

Big-bang integration, incremental integration

# Code Quality

## Readability

- Avoid Long Method
- Avoid Deep Nesting
- Avoid Complicated Expressions
- Avoid Magic Numbers
- Make the Code Obvious
- Structure Code Logically
- Don't Trip Up Readers
- KISS
- Avoid Premature Optimizations
- Single Level of Abstraction Per method
- Make the Happy Path Prominent

## Unsafe Practices

- Use the Default Branch
- Don't Recycle Variables
- Avoid Empty Catch Blocks
- Delete Dead Code
- Minimize Variable Scope
- Minimize Code Duplication

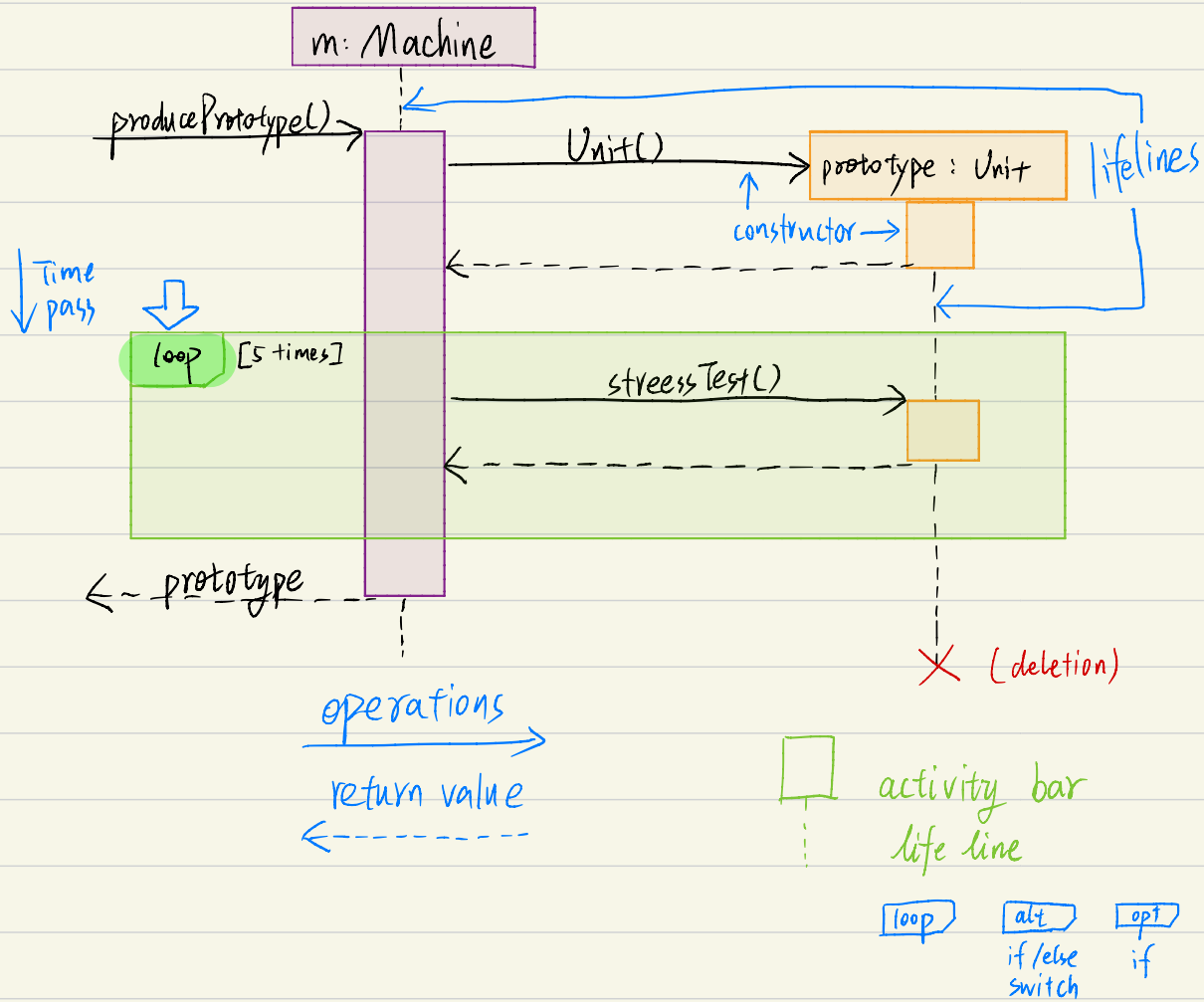
## Code Comments

- No Repeating Obvious
- Write to the Readers
- Explain What, Why, not How

## Naming

- Nouns for Things, Verbs for Actions
- Standard Words
- Meaningful
- Appropriate Length
- Avoid Misleading

# UML: Sequence Diagram



# Architecture

Design of software

## Styles

- n-Tier
- Client-Server
- Event-Driven

## API

Application Programming Interface

Which is useful when developing a large system

# Basic Design Approach

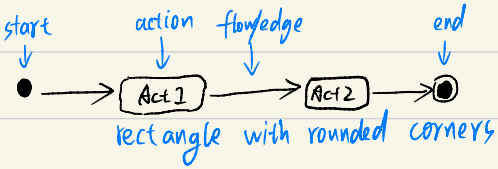
- Top-down
- Bottom-up
- Mix

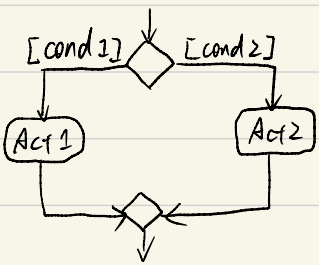
flesh out 充实

Agile Design
always updating, not declined up front

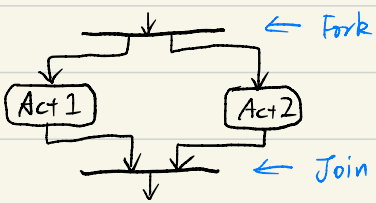
# UML: Activity Diagram

describe workflow




方括号
圆角矩形




rake notation (#) indicate a separate diagram

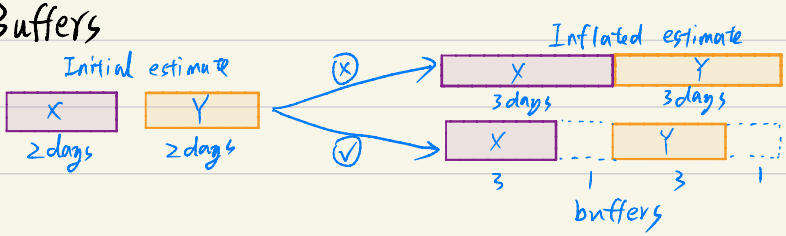
swimlane diagram: partitioned diagram

# Scheduling and Tracking

## Milestones

"each intermediate product release is a milestone"

## Buffers




Issue Tracker: useful tools, integrated in Github

## Work Breakdown Structure (WBS)

TaskID Task Estimated effort Prerequisite Task .....

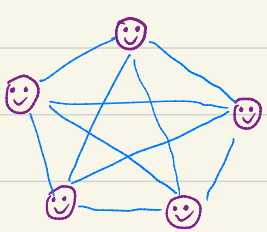
↓

man hour / day / month

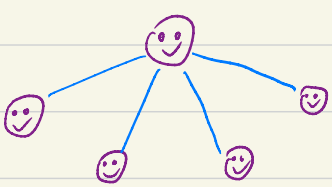
GANTT Charts

PERT Charts

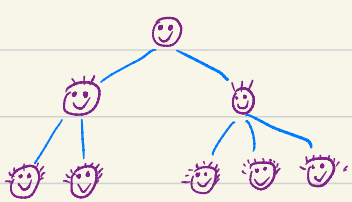
## Team Structure



egoless team



chief-programmer



strict-hierarchy

# Models

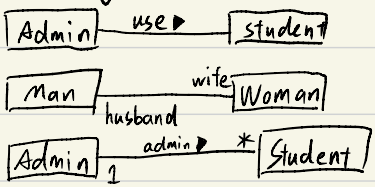
Abstraction

usage: To analyze / communicate with stakeholders / as a blueprint

UML class diagram describe the structure of an OOP solution

# Structure Diagram

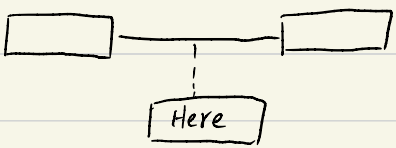
- class diagram 
  underlines denote class-level (static) attribute or variables
- associations: connection between objects / also between classes
  - labels — label ▸
  - roles
  - multiplicity




- Object diagram

curly braces { }

association class



# Design Principle

Abstraction • Data Abstraction • Control Abstraction • others

Coupling 耦合 a measure of the degree of dependence

high coupling → harder in maintenance, integration, testing and reuse

x is coupled to y if a change to y can potentially (not always) need a change in x

Cohesion 內聚 a measure of how strongly-related a component are

low cohesion: low understandability, maintainability, reusability

keep: same concept, close time, same data structure together.

SRP: Single Responsibility Principle

A class should have one, and only one, reason to change.

OCP: Open-Closed Principle

A module should be open for extension but closed for modification

SoC: Separation of Concerns Principle

- It reduce functional overlaps
- lead to 高內聚, 低耦合

# Types of Testing

## Unit Test

Stubs can isolate the SUT from its dependencies

Dependency injection is a way to implement stubs

## Integration Testing

testing whether different parts of the software work together as expected

## System Testing

typecally done by a testing team (QA team)

including non-functional requirements

## Acceptance Testing

aka. user acceptance testing (UAT)

## Alpha / Beta Testing

alpha testing: by user, under controlled condition set by development team

beta testing: by selected subset of target user, in real environment

open beta release: almost done release

# Developer Document

- for developer-as-user : API, tutorial ...
- for developer-as-maintainer

  Tutorial, How-to, explanation, reference

Should comprehensible, not only comprehensive
可理解的 全面的

Top-down is better to explain

# Design Principle

Substitutability : subclass can substitude superclass
Dynamic Binding : overridden methods
Static Binding : overloaded methods

} achieve polymorphism

Liskov Substitution Principle (LSP):
Derived classes must be substitutable for their base classes
Subclasses should not have more restrictive conditions

Law of Demeter (LoD): Principle of least knowledge

---

Interface Segregation Principle (ISP):
No client should be forced to depend on methods it does not use

Dependency Inversion Principle (DIP):
High-level module should not depend on low-level modules, both should depends on abstraction. Abstraction not depend on detail, detail depend on abstraction.

SOLID Principles: SRP, OCP, LSP, ISP, DIP

Brook's Law : Add people to a late project makes it later

# Design Patterns

elegant reusable solution to a commonly recurring problem with a given context in SE

must have: context, problem, solution

Singleton Pattern

Facade Pattern

Command Pattern

Abstraction Occurrence Pattern

Model View Controller (MVC)

# Test Case Design

Scripted testing, Exploratory testing

positive test cases, negative test cases

Black box, white box, gray box

## Equivalence Partitioning (EP)

- avoid redundant test case in one partition; ensure to cover all partitions

## Boundary Value Analysis (BVA)

typically choose 3 values around the boundary

# Quality Assurance

- Testability
- Test Coverage
  - function / method coverage
  - statement coverage
  - decision / branch coverage
  - condition coverage
  - path coverage
  - entry/exit coverage
- Test-driven developement (TDD)
- QA = validation + verification
  - validation → requirements
  - verification → implementations
- Code Review:
  - pair-programming
  - PR review
  - formal inspections
- Static Analysis

# Combining Multiple Test Input

testing
- all combination: effective but not efficient
- at least once
- all pairs
- random

Heuristic:
- each valid input at least once in a positive test case
- no more than one invalid input in a test case
- mix above

# SDLC Process Model

SDLC: Software Development Life Cycle:

- requirements, analysis, design, implementation, testing

## Sequential Models (waterfall model)

linear process, for not varing requirements (not often)

## Iterative Models

Several iteration of SDLC. BF or DF or mix

## Agile Models

- Scrum
- eXtreme Programming (XP)

# Reuse

- framework
- library
- platform