



## Chapter 4: Reusability-Oriented Software Construction Approaches

### 4.1 Metrics, Morphology and External Observations of Reusability

### 可复用性的度量、形态与外部表现



April 10, 2020

# Objective of this lecture

- **Advantages and disadvantages of software reuse**
- **Construction for/with reuse**
- **Characteristics of generic reusable components**
- **Methods of developing portable application systems**

第3章介绍了软件构造的核心理论（ADT）与技术（OOP），其核心是保证代码质量、提高代码安全性。

本章面向一个重要的外部质量指标：可复用性——如何构造出可在不同应用中重复使用的软件模块/API？

4-1节探讨可复用的软件应该“长什么样”，下一节学习“如何构造”

# Outline

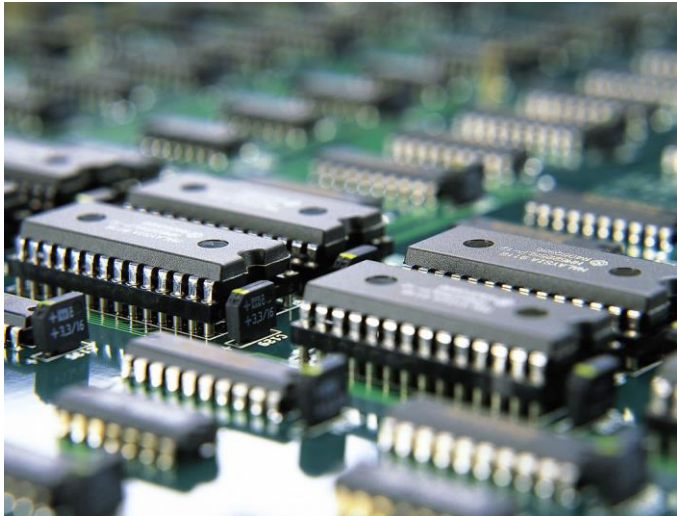
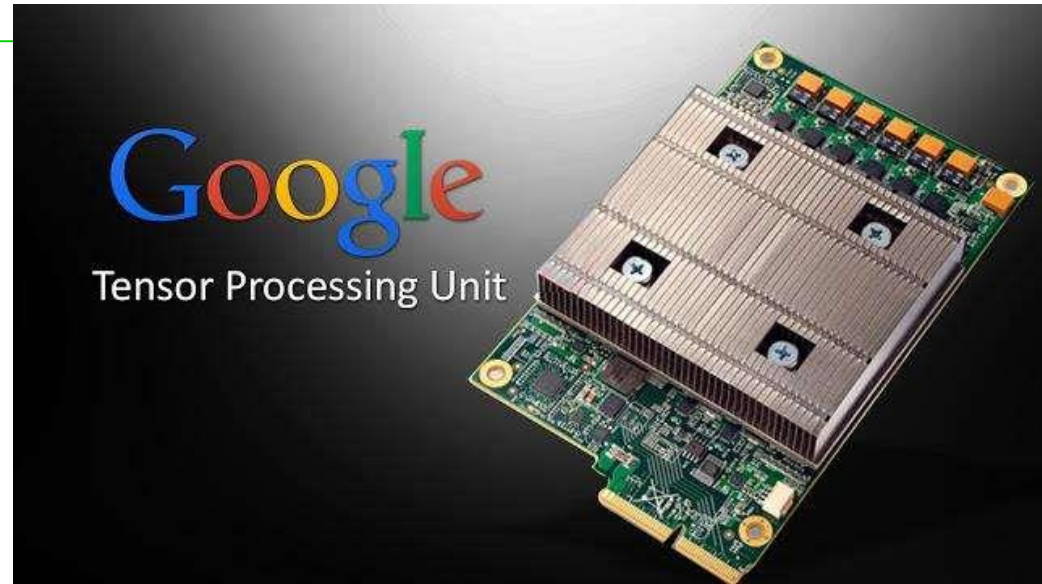
- **What is software reuse?**
- **How to measure “reusability”?**
- **Levels and morphology of reusable components**
  - Source code level reuse 源代码级别的复用
  - Module-level reuse: class/interface 模块级别的复用：类/抽象类/接口
  - Library-level: API/package 库级别的复用：API/包
  - System-level reuse: framework 系统级别的复用：框架
- **External observations of reusability**
  - Type Variation
  - Routine Grouping
  - Implementation Variation
  - Representation Independence
  - Factoring Out Common Behaviors
- **Summary**



# 1 What is Software Reuse?



# Hardware is reused inherently

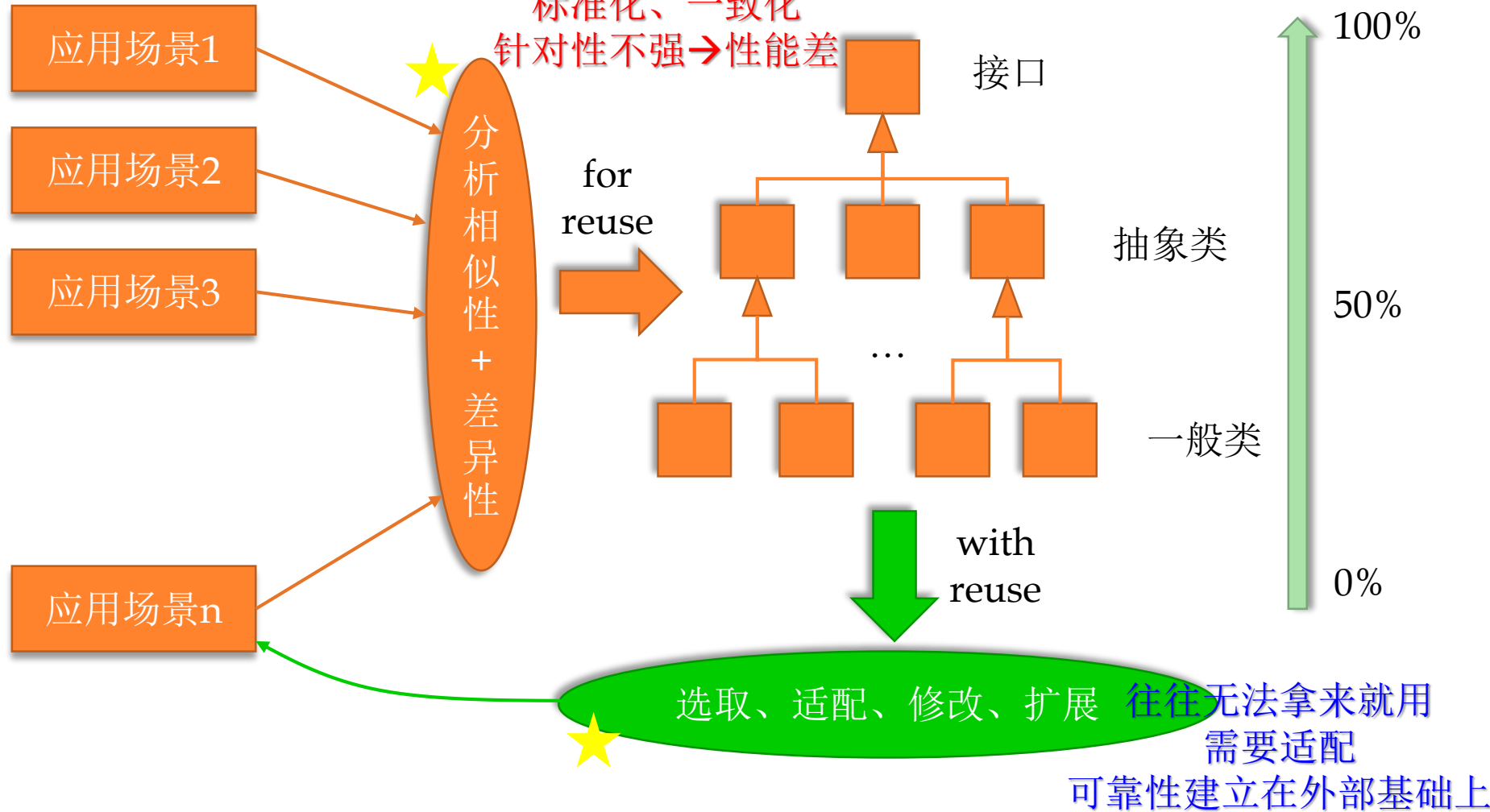


# Software reuse

- **Software reuse** is the process of implementing or updating software systems using existing software components.
- **Two perspectives of software reuse**
  - Creation: creating reusable resources in a systematic way (**programming for reuse** 面向复用编程：开发出可复用的软件)
  - Use: reusing resources as building blocks for creating new systems (**programming with reuse** 基于复用编程：利用已有的可复用软件搭建应用系统)
- **Why reuse?**
  - “The drive to create reusable rather than transitory artifacts has aesthetic and intellectual as well as **economic motivations** and is part of man’s desire for immortality.
  - It distinguishes man from other creatures and civilized from primitive societies” (Wegner, 1989).

# Programming for/with reuse

很大的适应性  
降低成本和开发时间  
充分的测试→高可靠  
标准化、一致化  
针对性不强→性能差





# Recall Lab2

- 在Lab2中：你开发了一个基于泛型的抽象接口**Graph<L>**，定义了支持图结构的ADT
- 针对该ADT，用两种不同的Rep，开发了两个不同的实现**ConcreteVertexGraph<L>**和**ConcreteEdgeGraph<L>**

Programming for reuse  
面向复用编程：开发出可复用的软件

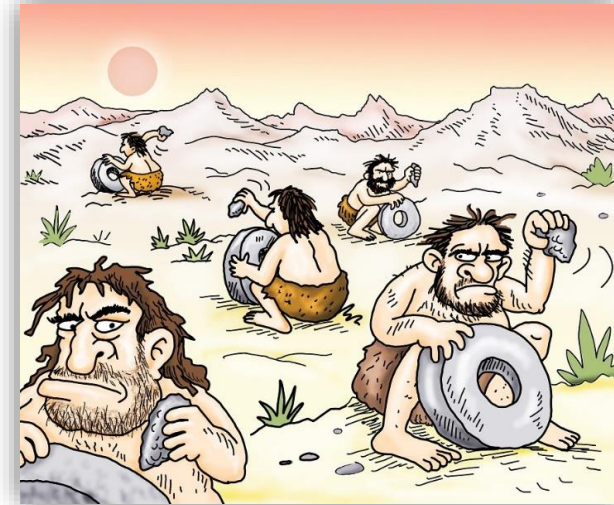
- 进而，你利用该ADT及其两个实现，完成了两个应用的开发：
  - Poetic Walks
  - Friendship Social Network

Programming with reuse  
基于复用编程：利用已有的可复用软件搭建应用系统



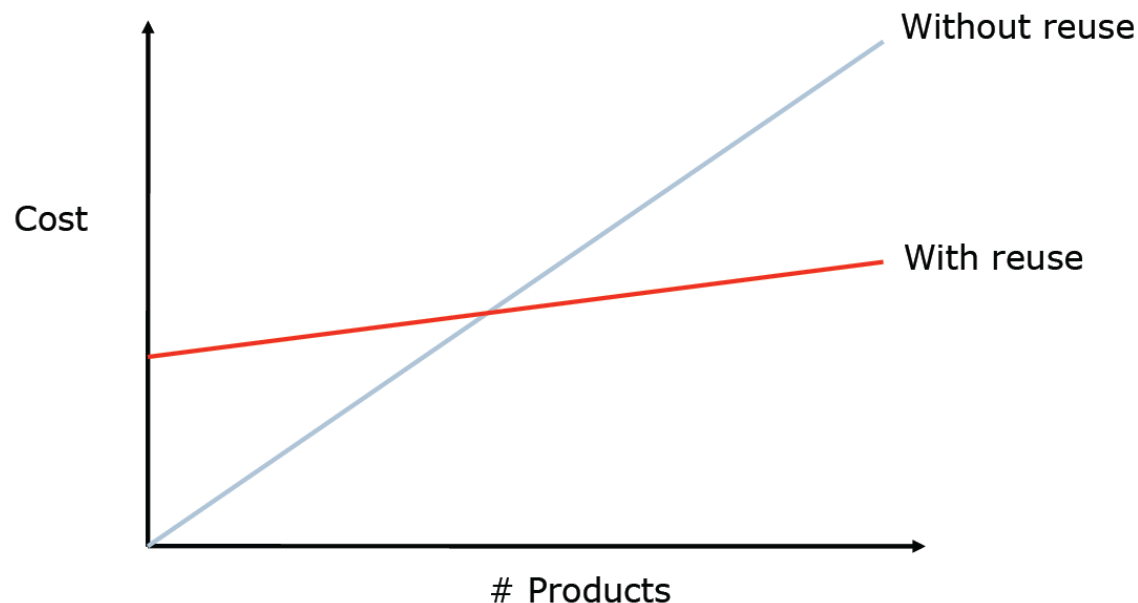
# Why reuse?

- **Reuse is cost-effective and with timeliness 降低成本和开发时间**
  - Increases software productivity by shortening software production cycle time (software developed faster and with fewer people)
  - Does not waste resources to needlessly "reinvent-the-wheel"
  - Reduces cost in maintenance (better quality, more reliable and efficient software can be produced)
- **Reuse produces reliable software 经过充分测试, 可靠、稳定**
  - Reusing functionality that has been around for a while and is debugged is a foundation for building on stable subsystems
- **Reuse yields standardization 标准化, 在不同应用中保持一致**
  - Reuse of GUI libraries produces common look-and-feel in applications.
  - Consistency with regular, coherent design.



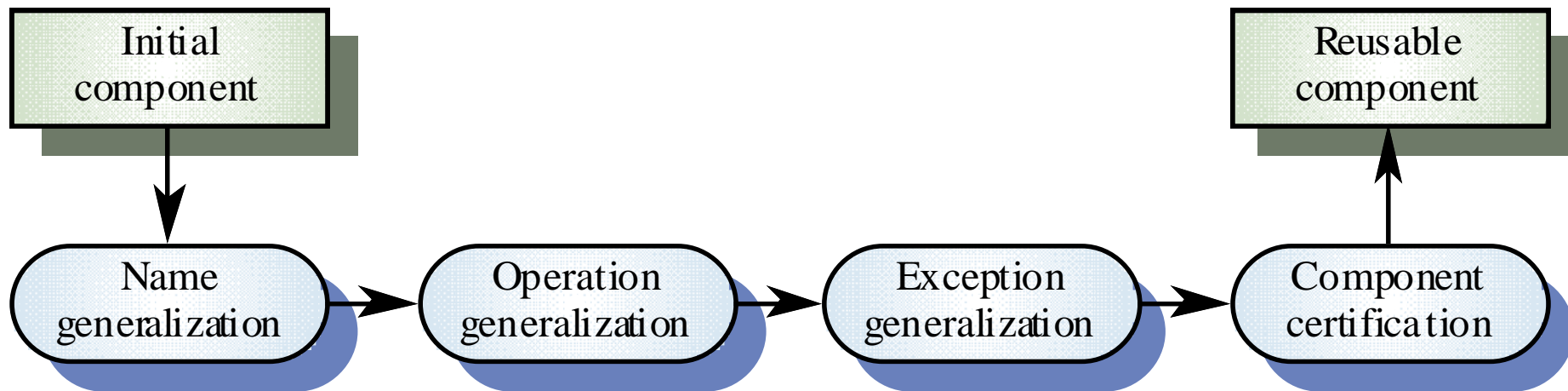
# Reuse costs

- **Reusable components** should be designed and built in a *clearly* defined, *open* way, with *concise* interface specifications, *understandable* documentation, and an eye towards *future use*. 做到这些，需要代价
- **Reuse is costly:** it involves spans organizational, technical, and process changes, as well as the cost of tools to support those changes, and the cost of training people on the new tools and changes. 不仅 program for reuse 代价高，program with reuse 代价也高



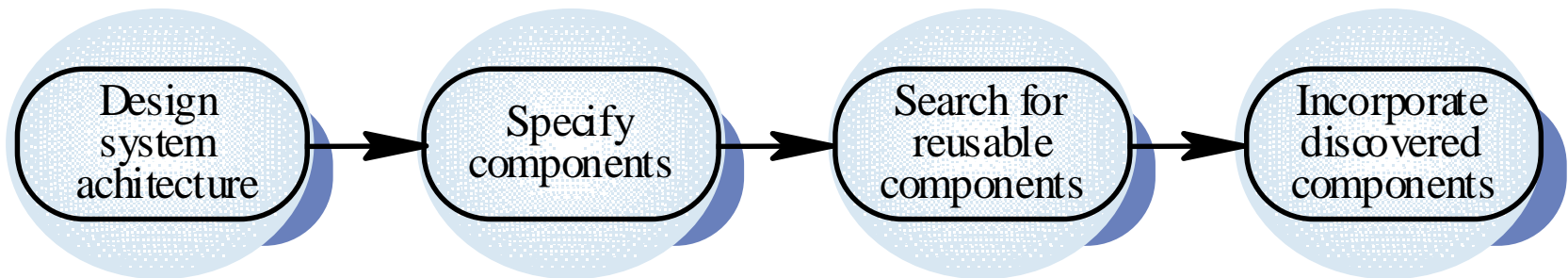
# Development for reuse: 开发可复用的软件

- The development cost of reusable components is higher than the cost of specific equivalents. This extra reusability enhancement cost should be an organization rather than a project cost. **开发成本高于一般软件的成本：要有足够高的适应性**
- Generic components may be less space-efficient and may have longer execution times than their specific equivalents. **性能差些：针对更普适场景，缺少足够的针对性**



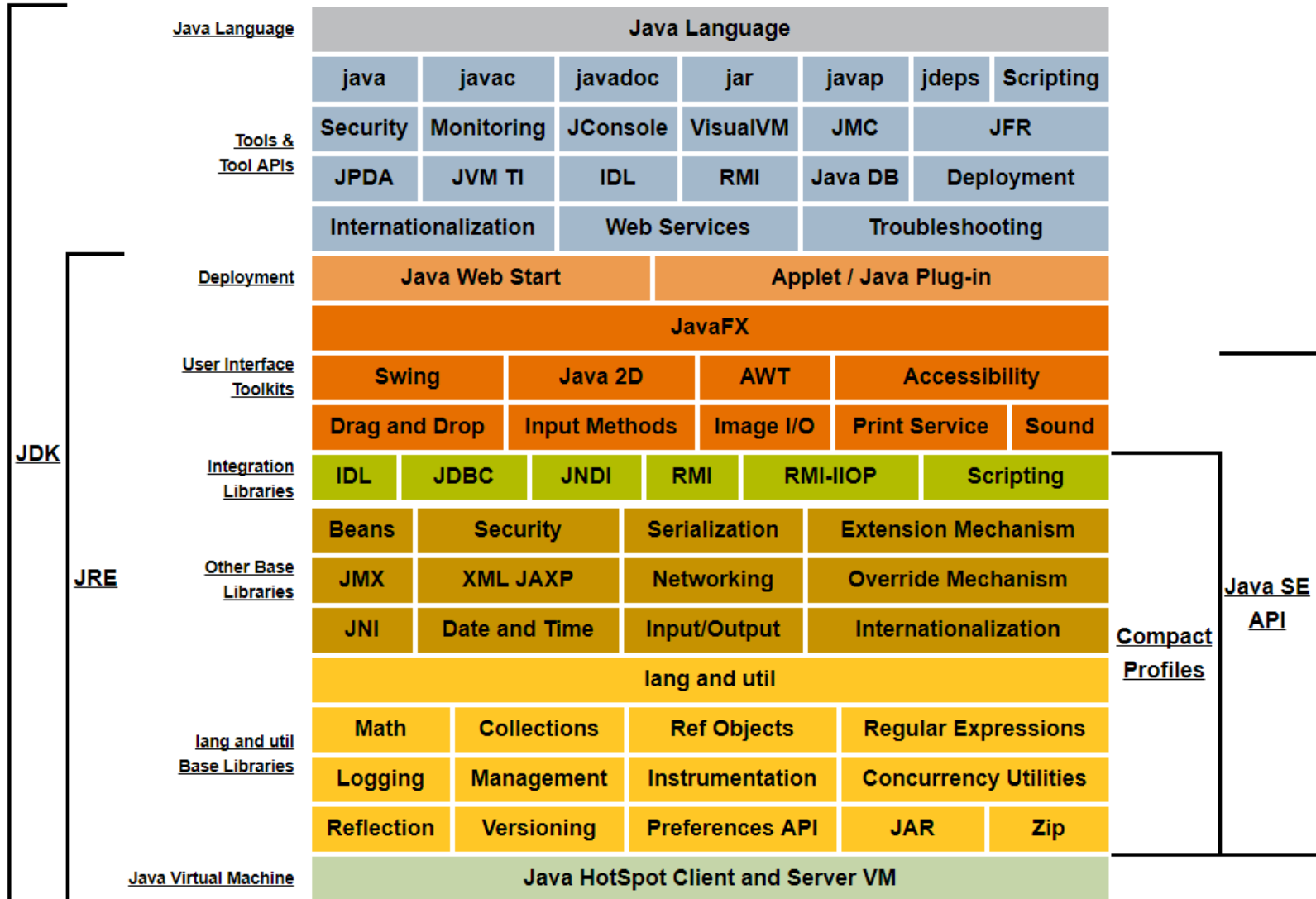
# Development with reuse: 使用已有软件进行开发

- Component management tools, such as repositories, for architectures, designs, documentation, and code must be developed and maintained. 可复用软件库，对其进行有效的管理



- A key issue: **adaptation** 往往无法拿来就用，需要适配
  - **Extra** functionality may have to be added to a component. When this has been **added**, the new component may be made available for reuse.
  - **Unneeded** functionality may be **removed** from a component to improve its performance or reduce its space requirements
  - The implementation of some component operations may have to be **modified**.

# Reusable Libraries and APIs in JDK



# Rich third-party libraries and APIs in Java



## Central Repository

<http://central.maven.org/maven2/>

URL	<a href="http://central.maven.org/maven2/">http://central.maven.org/maven2/</a>
Jars	2,467,181 indexed jars

### Published Jars by Year

2018	31,819
2017	709,421
2016	539,245
2015	362,410
2014	229,885
2013	170,787
2012	126,711
2011	106,285
2010	86,191
2009	49,537
2008	25,685
2007	12,478
2006	7,563
2005	9,067
2004	1

Home » [log4j](#) » [log4j](#) » 1.2.17



## Apache Log4j » 1.2.17

Apache Log4j 1.2

### License

Apache 2.0

### Categories

Logging Frameworks

### Organization

Apache Software Foundation

### HomePage

<http://logging.apache.org/log4j/1.2/>

### Date

(May 26, 2012)

### Files

[pom \(21 KB\)](#) [bundle \(478 KB\)](#) [View All](#)

### Repositories

[Central](#) [Apache Releases](#) [Redhat GA](#)  
[Sonatype Releases](#) [Spring Plugins](#)

### Used By

11,493 artifacts

[Maven](#)

[Gradle](#)

[SBT](#)

[Ivy](#)

[Grape](#)

[Leiningen](#)

[Buildr](#)

```
<!-- https://mvnrepository.com/artifact/log4j/log4j -->
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>
```

☒ Include comment with link to declaration



## 2 How to measure “reusability”?





# Measure resuability

- **How frequently** can a software asset be reused in different application scenarios? 复用的机会会有多频繁？复用的场合有多少？
  - The more chance an asset is used, the higher reusability it has.
  - Write once, reuse multiple times.
- **How much** are paid for reusing this asset? 复用的代价有多大？
  - Cost to buy the asset and other mandatory libraries 搜索、获取
  - Cost for adapting and extending it 适配、扩展
  - Cost for instantiating it 实例化
  - Cost for changing other parts of the system that interact with it 与软件其他部分的互连的难度

# Reusability

- Reusability implies some explicit management of build, packaging, distribution, installation, configuration, deployment, maintenance and upgrade issues.
- A software asset with high reusability should:
  - Brief (small size) and Simple (low complexity) 小、简单
  - Portable and Standard Compliance 与标准兼容
  - Adaptable and Flexible 灵活可变
  - Extensibility 可扩展
  - Generic and Parameterization 泛型、参数化
  - Modularity 模块化
  - Localization of volatile (changeable) design assumptions 变化的局部性
  - Stability under changing requirements 稳定
  - Rich documentation 丰富的文档和帮助



# 3 Levels and morphology of reusable components



# Levels of Reuse

- **A reusable component may be code** 最主要的复用是在代码层面
  - Most prevalent: what most programmers relate with reuse
- **But benefits result from a broader and higher-level view of what can be reused.** 但软件构造过程中的任何实体都可能被复用
  - Requirements 需求
  - Design and specifications 设计/规约spec
  - Data 数据
  - Test cases 测试用例
  - Documentation 文档

# Types of Code Reuse

- **White box reuse 白盒复用：源代码可见，可修改和扩展**
  - Reuse of code when code itself is available. Usually requires some kind of modification or adaptation 复制已有代码当正在开发的系统，进行修改
  - **Pro:** You can customize the module to fit the specific situation, this allows reuse in more situations 可定制化程度高
  - **Con:** You now own the customized result, so it adds to your code complexity. You requires intrinsic knowledge on component internals. 对其修改增加了软件的复杂度，且需要对其内部充分的了解
- **Black box reuse 黑盒复用：源代码不可见，不能修改**
  - Reuse in the form of combining existing code by providing some “glue”, but without having to change the code itself - usually because you do not have access to the code 只能通过API接口来使用，无法修改代码
  - **Pro:** Simplicity and Cleanliness 简单，清晰
  - **Con:** Many times it is just not possible 适应性差些



# (1) Source code reuse



# Reusing Code – Lowest Level

- **Copy/paste parts/all into your program**
- **Maintenance problem**
  - Need to correct code in multiple places
  - Too much code to work with (lots of versions)
- **High risk of error during process**
- **May require knowledge about how the used software works**
- **Requires access to source code**
- **相关研究1：如何从互联网上快速找到需要的代码片段？**
- **反向研究：如何从源代码中检测出克隆代码(clone code)？**



# Example code search: [greppcode.com](http://greppcode.com)

The screenshot displays the greppcode.com website interface. At the top, there is a navigation bar with links for Stack Trace Search, Eclipse, IntelliJ, Contact, FAQ, and social media icons. The main search area features a large input field and a 'Search' button. Below the search bar, the results are categorized into 'Types', 'Projects', and 'Methods'. The 'Types' tab is selected, showing a list of search results for the term 'hashset'. On the left side, there is a sidebar with 'All Repositories' and 'All Kinds' sections. The 'All Repositories' section lists various repositories like JDK, Maven-Central, Cloudera, Hyracks, Java.net, JBoss, NetBeans, Pentaho, PrimeFaces, SpringSource, Eclipse-4.4.2, Eclipse-4.4.1, Eclipse-4.4.0, Eclipse-4.3.1, Eclipse-4.3, Eclipse-4.2.2, Eclipse-4.2, Eclipse-3.7.2, Eclipse-3.6.2, Eclipse-Virgo, EclipseLink, GrepCode, and GrepCode-Eclipse. The 'All Kinds' section lists Class, Interface, Enum, and Annotation. The search results for 'hashset' are as follows:

- java.util.HashSet** - This class implements the Set interface, backed by a hash table (actually a HashMap instance)
  - gwt-user: 2.7.0, 2.7.0-rc1, 2.7.0-beta1, 2.6.1, 2.6.1-rc2, 2.6.1-rc1, 2.6.0, 2.6.0-rc4, 2.5.1, 2.5.0, 2.4.0, 2.3.0, 2.2.0, 2.1.1, 2.1.0, 2.0.4, ...
  - emul: 0.15, 0.14, 0.13, 0.12, 0.11, 0.10, 0.9, 0.8.1, 0.8, 0.7.2, 0.7, 0.6, 0.5, 0.4
  - gwt-servlet: 2.7.0, 2.7.0-rc1, 2.6.1, 2.6.1-rc2, 2.6.1-rc1, 2.6.0, 2.6.0-rc4, 2.6.0-rc3, 2.5.1, 2.5.0, 2.4.0, 2.3.0, 2.2.0, 2.1.1, 2.1.0, 2.0.4, ...
  - org.apache.servicemix.bundles.gwt-user: 2.6.0.1, 2.4.0.1
- com.thaiopensource.validate.mns.Hashset**
  - jing: 20091111
  - jing: 20120724.0.0, 20091111.0.0
  - wicketstuff-jing: 1.11, 1.10, 1.9, 1.8, 1.7, 1.6
  - jing: 20150629VNU, 20130806VNU
- com.thaiopensource.validate.nrl.Hashset**
  - jing: 20091111
  - jing: 20120724.0.0, 20091111.0.0
  - wicketstuff-jing: 1.11, 1.10, 1.9, 1.8, 1.7, 1.6
  - jing: 20030619
- com.thaiopensource.validate.nvdl.Hashset** - Utility class, stores a set of objects
  - jing: 20091111

# GitHub code search: [github.com/search](https://github.com/search)

The screenshot displays the GitHub Advanced Search interface. At the top, the navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. The search bar contains the query 'hashmap language:Java' and a Search button. Below the search bar, the results page shows a summary of 13,052,763 available code results. On the left sidebar, there are filters for Repositories (974), Code (13M), Commits (291K), Issues (28K), Wikis (8K), and Users (1). Below these are language filters for Java (13,052,763), C++ (669,400), HTML (620,079), Smali (489,293), JavaScript (245,495), XML (176,562), Scala (151,977), Java Server Pages (143,091), PHP (104,555), and C (77,620). The main content area shows two code snippets from the repository 'leijiangping/history'. The first snippet is ' UserDao.java' showing a package declaration, imports, and a loop that puts elements into a HashMap. The second snippet is ' RedisCaptchaStore.java' showing a return statement and an empty method that initializes a new HashMap and calls deleteAll().

Advanced search

hashmap language:Java

Search

hashmap language:Java

Pull requests Issues Marketplace Explore

Showing 13,052,763 available code results ⓘ

Sort: Recently indexed ▾

Repositories 974

Code 13M

Commits 291K

Issues 28K

Wikis 8K

Users 1

Languages

Java	13,052,763
C++	669,400
HTML	620,079
Smali	489,293
JavaScript	245,495
XML	176,562
Scala	151,977
Java Server Pages	143,091
PHP	104,555
C	77,620

leijiangping/history – UserDao.java

Showing the top two matches Last indexed 2 minutes ago

```
1 package com.systoon.integration.user.dao;
2
3 import java.util.Collection;
4 import java.util.HashMap;
5
6 ...
30     for (int i = 0; i < ids.length; i++) {
31         map = new HashMap<String, Object>();
32         map.put("disabled", disable ? 1 : 0);
33         map.put("id", ids[i]);
```

leijiangping/history – RedisCaptchaStore.java

Showing the top match Last indexed 2 minutes ago

```
98         return client.hkeys(CaptchaRedisKey);
99     }
100 });
101 return ret;
102 }
103
104 @Override
105 public void empty() {
106     // this.store = new HashMap();
107     deleteAll();
```

Advanced search Cheat sheet

# Searchcode: [searchcode.com](https://searchcode.com)

The screenshot shows the Searchcode website interface. At the top, there's a search bar with the text 'regionmatches' and a green 'search' button. To the right of the search bar are links for 'lopers', 'Updates', and 'searchcode server'. Below the search bar is a banner for Google Cloud. The main content area shows 'About 598 results: "regionmatches"'. On the left side, there's a sidebar with 'Page 1 of 30', navigation buttons for 'Previous' and 'Next', and a featured advertisement for Monday.com. Below the ad, there are filters for 'Filter Results' (Remove, Apply), 'Sources' (Github, 598), and 'Languages' (Java, 598). The main content area displays two code snippets. The first snippet is from 'RegionMatches.java in ManagedRuntimeInitiative' and shows a Java class with a method 'regionMatches' that throws a RuntimeException. The second snippet is from 'OldStringTest.java in android\_libcore' and shows test methods for 'regionMatches'.

searchcode

regionmatches

search

lopers

Updates

searchcode server


Google Cloud — Use GCP products for free up to non-expiring usage limits. AD

About 598 results: "regionmatches"

Page 1 of 30

◀ Previous

Next ▶

  
Project tracking,  
teamwork & client  
reporting like you've  
never seen before.  
ads via Carbon

Filter Results

Remove

Apply

Sources

☒ Github 598

Languages

Filter Languages

☒ Java 598

Filter Results

RegionMatches.java in ManagedRuntimeInitiative [git://github.com/GregBowyer/ManagedRuntimeInitiative.git](https://github.com/GregBowyer/ManagedRuntimeInitiative.git) | 41 lines | Java Show 3 matches

```

26. * @bug 4016509
27. * @summary test regionMatches corner case
30.
31. public class RegionMatches {
36.
37.     if (!s1.regionMatches(0,s2,0,Integer.MIN_VALUE))
38.         throw new RuntimeException("Integer overflow in RegionMatches");
39. }

```

OldStringTest.java in android\_libcore [https://github.com/PAmoto/android\\_libcore.git](https://github.com/PAmoto/android_libcore.git) | 525 lines | Java

```

141.
142. public void test_regionMatchesILjava_lang_StringII() {
143.     assertFalse("Returned true for negative offset.", hw1.regionMatches(-1,
144.         hw2, 2, 5));
145.     assertFalse("Returned true for negative offset.", hw1.regionMatches(2,
147.         assertFalse("Returned true for toffset+len is greater than the length.",
148.             hw1.regionMatches(5, hw2, 2, 6));
149.     assertFalse("Returned true for ooffset+len is greater than the length.",
150.         hw1.regionMatches(2, hw2, 5, 6));
152.
153. public void test_regionMatchesZILjava_lang_StringII() {
155.
156.     assertFalse("Returned true for negative offset.", hw1.regionMatches(true,
157.         -1, hw2, 2, 5));

```



## (2) Module-level reuse: class/interface



Inheritance

Use

Composition/aggregation

Delegation/association

# Reusing classes

- **A class is an atomic unit of code reuse**
  - Source code not necessary, class file or **jar/zip**
  - Just need to include in the **classpath**
  - Can use **javap** tool to get a class's public method headers
- **Documentation very important (Java API)**
- **Encapsulation helps reuse**
- **Less code to manage**
- **Versioning, backwards-compatibility still problem**
- **Need to package related classes together -- Static Linking**

# Approaches of reusing a class: inheritance继承

- **Java provides a way of code reuse named **Inheritance****
  - Classes extend the properties/behavior of existing classes
  - In addition, they might **override** existing behavior
- **No need to put dummy methods that just forward or delegate work**
- **Captures the real world better**
- **Usually need to design inheritance hierarchy before implementation**
- **Cannot cancel out properties or methods, so must be careful not to overdo it**

# Approaches of reusing a class: delegation 委托

- **Delegation** is simply when one object relies on another object for some subset of its functionality (one entity passing something to another entity)
  - e.g. the Sorter is delegating functionality to some Comparator
- **Judicious delegation enables code reuse**
  - Sorter can be reused with arbitrary sort orders
  - Comparators can be reused with arbitrary client code that needs to compare integers
- **Explicit delegation:** passing the sending object to the receiving object
- **Implicit delegation:** by the member lookup rules of the language
- **Delegation** can be described as a low level mechanism for sharing code and data between entities.



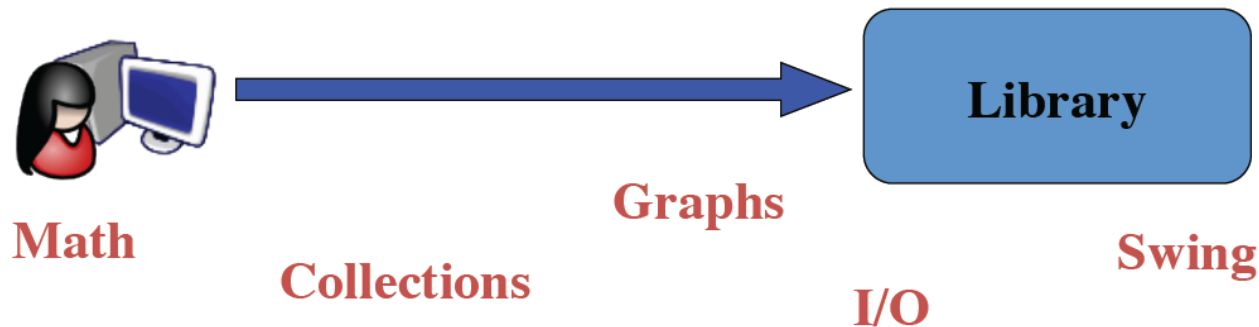


## (3) Library-level reuse: API/Package



# Libraries

- **Library:** A set of classes and methods (APIs) that provide reusable functionality

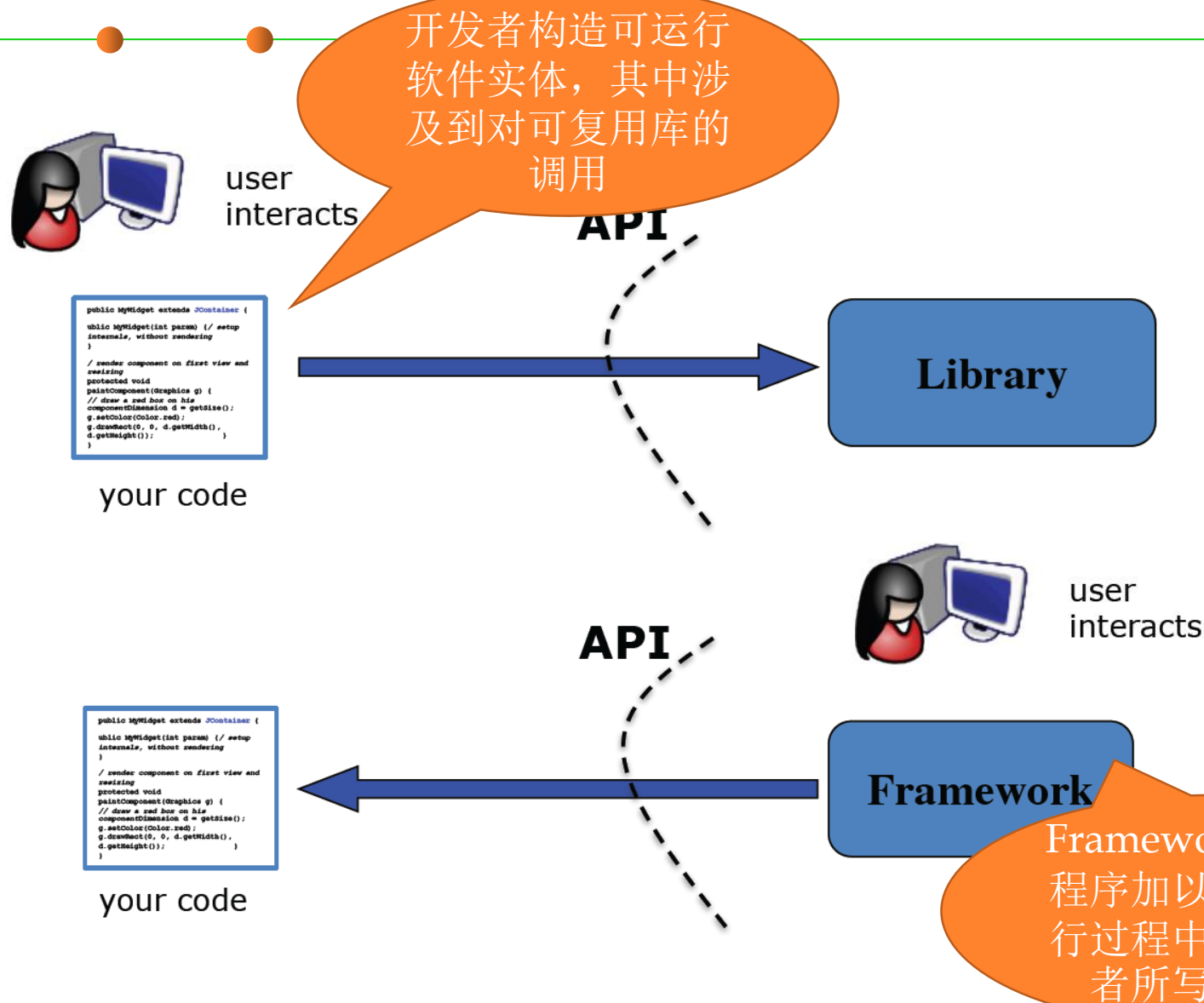


# Framework

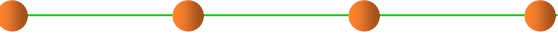
- **Framework: Reusable skeleton code that can be customized into an application**
- **Framework calls back into client code**
  - The Hollywood principle: “Don’t call us. We’ll call you.”



# General distinction: Library vs. framework



# Characteristics of a good API

- 
- 
- **Easy to learn**
  - **Easy to use, even without documentation**
  - **Hard to misuse**
  - **Easy to read and maintain code that uses it**
  - **Sufficiently powerful to satisfy requirements**
  - **Easy to evolve**
  - **Appropriate to audience**

# Guava: Google core libraries for Java

The screenshot displays the GitHub repository for Google's Guava library. At the top, the repository name 'google / guava' is shown with 2,107 watches, 23,088 stars, and 5,311 forks. Below this, navigation tabs for Code, Issues (685), Pull requests (55), Projects (0), Wiki, and Insights are visible. The repository description 'Google core libraries for Java' is followed by tags for 'guava' and 'java'. A summary bar indicates 4,673 commits, 6 branches, 76 releases, 137 contributors, and the Apache-2.0 license. Action buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download' are present. The commit history table lists recent changes, including updates to 'Ignore FinalizableReferenceQueueClassLoaderUnloadingTest in JDK 9' across various submodules like android, guava-gwt, guava-testlib, guava-tests, and guava, as well as updates to .gitattributes, .gitignore, .travis.yml, CONTRIBUTING.md, COPYING, README.md, cycle\_whitelist.txt, javadoc-stylesheet.css, and pom.xml.

Commit Message	Time Ago
netdpb and ronshapiro Ignore FinalizableReferenceQueueClassLoaderUnloadingTest in JDK 9.	Latest commit 21f4dd7 6 days ago
android Ignore FinalizableReferenceQueueClassLoaderUnloadingTest in JDK 9.	4 days ago
guava-gwt Supersource nullToEmpty and emptyToNull in Strings.java to native JS.	4 days ago
guava-testlib Group overloads together.	2 months ago
guava-tests Ignore FinalizableReferenceQueueClassLoaderUnloadingTest in JDK 9.	4 days ago
guava Ignore FinalizableReferenceQueueClassLoaderUnloadingTest in JDK 9.	4 days ago
refactorings Open source refactorings directory. This is knowingly very simple, wi...	4 days ago
util Actually export print_surefire_reports.sh.	5 months ago
.gitattributes Add a .gitattributes file to control line ending normalization, which...	3 years ago
.gitignore Add .DS_Store to .gitignore.	2 years ago
.travis.yml Hide Downloading/Downloaded lines from output.	a month ago
CONTRIBUTING.md usage of American English spelling for "license"	2 years ago
CONTRIBUTORS fix indentation	7 years ago
COPYING fix indentation	7 years ago
README.md Prepare for release 24.1.	18 days ago
cycle_whitelist.txt Fix two bugs in MinMaxPriorityQueue (introduced in [] First is a bug ...	a year ago
javadoc-stylesheet.css Fix a couple of issues with JDK7 javadoc style by using a slightly cu...	5 years ago
pom.xml Work around breakage in which release builds run maven-javadoc-plugin...	2 months ago

# Apache Commons

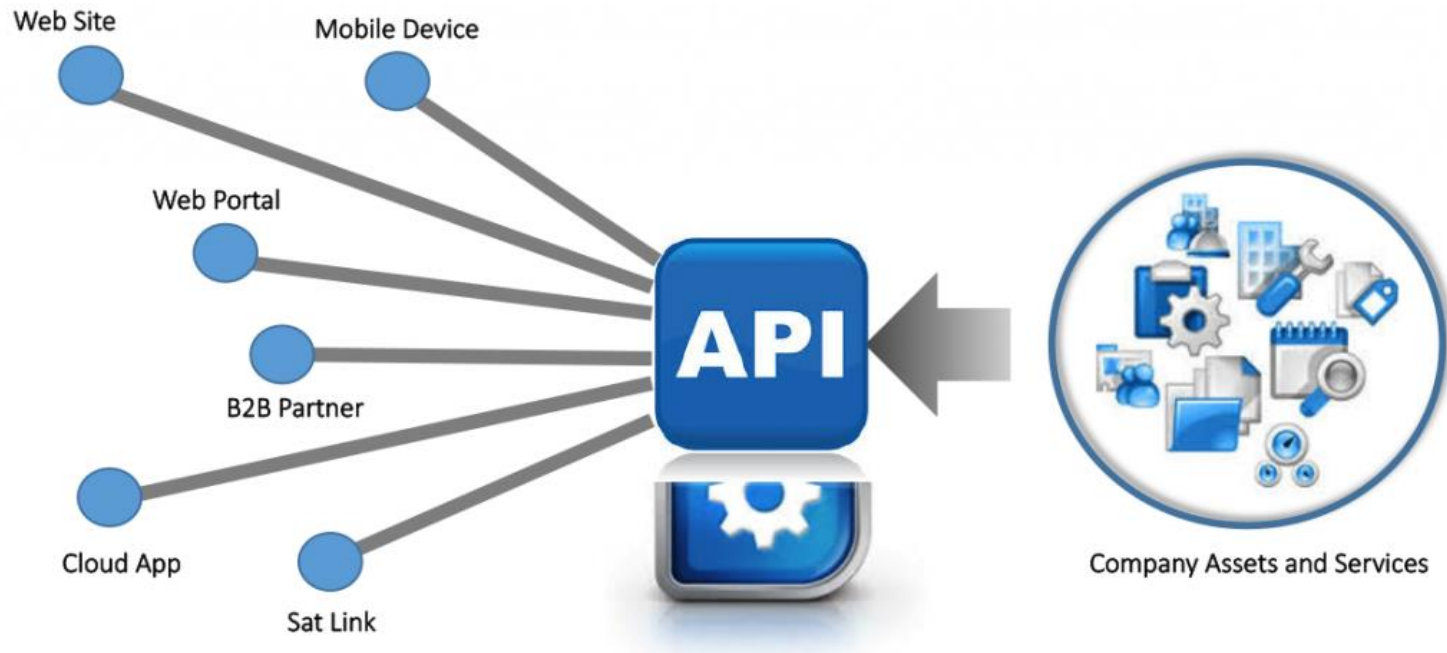
- Apache Commons is an Apache project focused on all aspects of reusable Java components.
  - <https://commons.apache.org>
  - [https://github.com/apache/commons-\\*](https://github.com/apache/commons-*)



# Apache



# API on Web/Internet: Web Services/Restful APIs



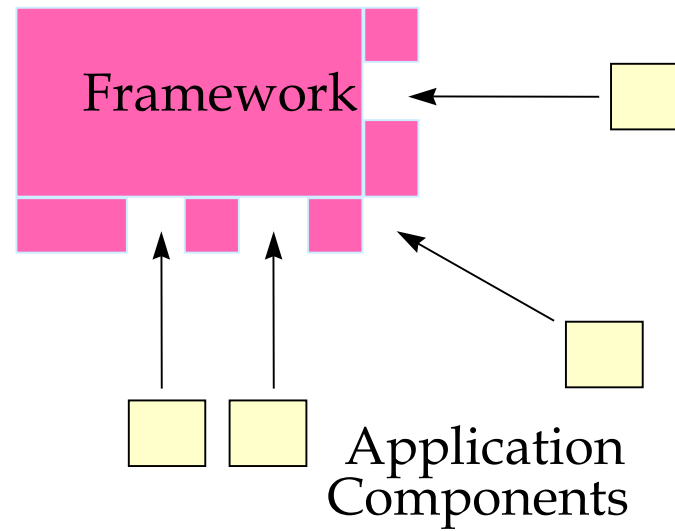


## (4) System-level reuse: Framework



# Application Frameworks

- Frameworks are sub-system design containing a collection of **abstract and concrete classes** along with interfaces between each class 框架：一组具体类、抽象类、及其之间的连接关系
  - 只有“骨架”，没有“血肉”
- A framework is an abstraction in which software providing **generic functionality** can be selectively changed by additional **user-written code**, thus providing application-specific software. 开发者根据 framework 的规约，填充自己的代码进去，形成完整系统

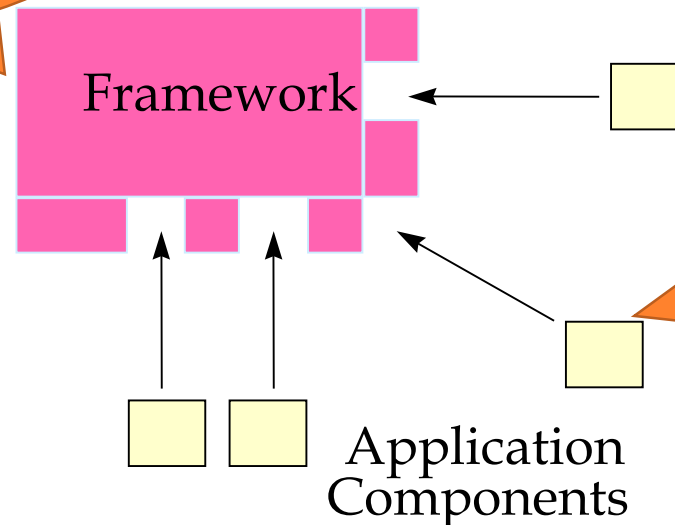


# Application Frameworks

骨骼清奇，天赋异禀

- Reusability leverages of the **application domain knowledge** and prior effort of experienced developers 领域知识的复用
  - Data processing, GUI, etc
  - 将framework看作是更大规模的API复用，除了提供可复用的API，还将这些模块之间的关系都确定下来，形成了整体应用的领域复用

Framework作为主程序加以执行，执行过程中调用开发者所写的程序



开发者根据Framework预留的接口所写的程序

# Application Frameworks

如何把你的Lab2 P3变成一个Framework, 允许其他人通过特定机制往里添加各类棋类规则, 从而不断扩展它的能力?

符合真实棋类规则的Action  
除了围棋和国际象棋规则之外的其他棋类游戏  
AI自主下棋?  
深度学习?  
漂亮的GUI?

其实你的P3离AlphaGo距离并不远 ☺☺☺☺

# Framework Design

- **Frameworks differ from applications**

- The level of abstraction is different as frameworks provide a solution for a family of related problems, rather than a single one.
- To accommodate the family of problems, the framework is incomplete, incorporating hot spots and hooks to allow customization

- **Frameworks can be classified by the techniques used to extend them.**

- Whitebox frameworks      黑盒框架
- Blackbox frameworks      白盒框架

# White-box and Black-Box Frameworks

- **Whitebox frameworks** 白盒框架，通过代码层面的继承进行框架扩展
  - Extensibility achieved through **inheritance** and **dynamic binding**.
  - Existing functionality is extended by **subclassing** framework base classes and **overriding** predefined hook methods
  - Often design patterns such as the template method pattern are used to **override** the hook methods.
- **Blackbox frameworks** 黑盒框架，通过实现特定接口/**delegation**进行框架扩展
  - Extensibility achieved by defining **interfaces** for components that can be plugged into the framework.
  - Existing functionality is reused by defining components that **conform to a particular interface**
  - These components are integrated with the framework via **delegation**.




## 4 External observations of reusability





# External observations of reusability

- 
- **Type Variation** 类型可变
  - **Routine Grouping** 功能分组
  - **Implementation Variation** 实现可变
  - **Representation Independence** 表示独立
  - **Factoring Out Common Behaviors** 共性抽取

# Type Variation

- Reusable components should be **type-parameterized** so that they can adapt to different data types (input, computation, and output);
  - A reusable module should be applicable to many different types of element, without requiring developers to perform manual changes to the software text.
  - In other words, we need a facility for describing type-parameterized modules, also known more concisely as generic modules.
- Genericity: reusable components should be **generic**.
- 类型可变(泛型): 适应不同的类型, 且满足LSP

# Implementation Variation

- In practice a wide variety of applicable data structures and algorithms.
- Such variety indeed that we cannot expect a single module to take care of all possibilities; it would be enormous.
- We will need a family of modules to cover all the different implementations.
- 实现可变：ADT有多种不同的实现，提供不同的representations和abstract function，但具有同样的specification (pre-condition, post-condition, invariants)，从而可以适应不同的应用场景

# Routine Grouping

- A self-sufficient reusable module would need to include a set of routines, one for each of the operations.
- Completeness
- 提供完备的细粒度操作，保证功能的完整性，不同场景下复用不同的操作(及其组合)

# Representation Independence

- A general form of reusable module should enable clients to specify an operation without knowing how it is implemented.
- Representation Independence is an extension of the rule of Information Hiding, essential for smooth development of large systems: implementation decisions will often change, and clients should be protected. 内部实现可能会经常变化，但客户端不应受到影响。
- Representation Independence reflects the client's view of reusability — the ability to ignore internal implementation details and variants
- 表示独立性、信息隐藏

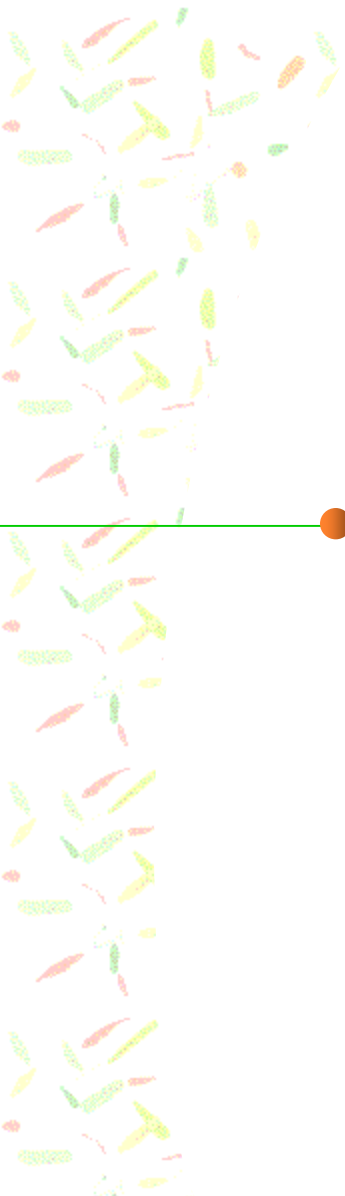
# Factoring Out Common Behaviors

- **Factoring Out Common Behaviors**, reflects the view of the supplier and, more generally, the view of developers of reusable classes.
- **The goal will be to take advantage of any commonality that may exist within a family or sub-family of implementations.**
- 将共同的行为（共性）抽象出来，形成可复用实体：父类、抽象类
- The variety of implementations available in certain problem areas will usually demand, as noted, a solution based on a family of modules.
- Each of these categories covers many variants, but it is usually possible **to find significant commonality between these variants.**

只要看到自己写了重复的或相似的代码，  
想办法抽取出来形成可复用方法/类



# Summary



# Summary

- **What is software reuse?**
- **How to measure “reusability”?**
- **Levels and morphology of reusable components**
  - Source code level reuse
  - Module-level reuse: class/interface
  - Library-level: API/package
  - System-level reuse: framework
- **External observations of reusability**
  - Type Variation
  - Routine Grouping
  - Implementation Variation
  - Representation Independence
  - Factoring Out Common Behaviors





The end

April 10, 2020