

JC2002 Java 程序设计

第 1 天：介绍 (CS)

10 月 30 日星期一

JC2002 Java 程序设计

第 1 天，第 1 课时：课程介绍和实际操作

破冰活动

- 我是谁？
- 您对 Java 和一般编程有多熟悉？
- 您对本课程有何期望？

课程安排

- **讲座**（理论课）：周一、二、三、五
- 周一、二、四的**实践课程**
- **实践作业**，占评估的 30
 - 作业将在第三周发布
 - 在 Codio 上提交（截止日期：12 月 10 日）
- **考试**，占评估的 70

- 纸质闭卷考试

课程主题

- **第 1 周**Java 基础知识
 - 使用基本 Java 工具、版本控制和测试
 - Java 语法、条件结构、循环、类、对象、方法
- **第 2 周**面向对象编程、图形用户界面
 - 内存概念、异常
 - 使用 Swing 在 Java 中实现图形用户界面
- **第 3 周**并发和数据结构

- 多线程和内存概念
- 字符串操作、集合
- **第 4 周**Java 高级主题
 - 文件处理、数据库编程等。

课程主题 - 不仅仅是 Java

- 专业软件开发实践
 - 版本源控制，跟踪开发情况
 - 使用测试和自动测试来简化开发工作
 - Linux 命令行、阅读文档等。



Java Platform, Standard Edition Tools Reference

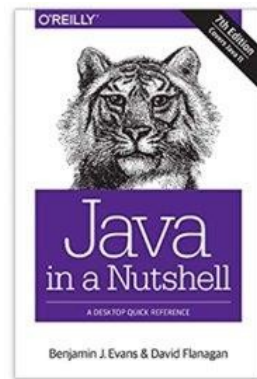
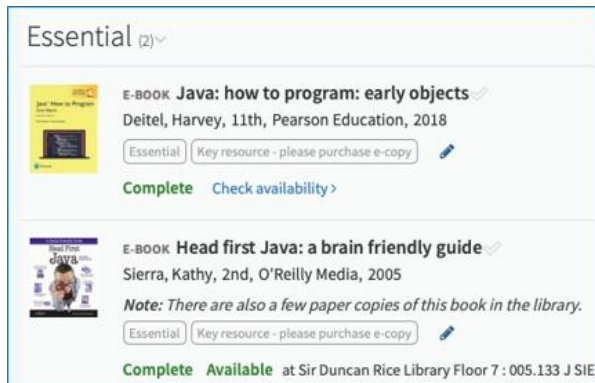
[Contents](#) [Previous](#) [Next](#)

javadoc



阅读课程教科书

讲座只是学习过程的一部分。您还需要阅读有关语言的书籍！



- 图书馆和网上的课程资料（我们将增加更多资料）
- 图书馆有大量关于软件开发的资料

-> 图书和在线书籍

任务

- 编程作业 - 占 30
- 考试 - 占 70% 的分数
- 迟交政策
 - 最长迟到 24 小时，您的分数将被扣 10
 - 逾期 1 周 以内将被处以 25% 的罚款
 - 在此之后的任何递交都将被归类为 未递交。

- 反馈意见

- 反馈将在您提交作品后三周内给出

学术诚信

- **剽窃**是指在作品中不注明出处地使用他人的作品或观点。
提交评估。
- **串通**是指学生在撰写个人作业时未经授权进行合作。
- **"合同作弊"**是一种学术不端行为，即学生提交的作品是由学生以外的其他人制作的，无论是否有报酬。

参见：<https://www.abdn.ac.uk/students/academic-life/academic-integrity.php>

学术诚信

- 剽窃

Reference and cite appropriately sources used in assessments and other learning activities.

Include direct quotes, if they are deemed appropriate by your subject area, acknowledged with quotation marks or indented, and remember to reference or cite your sources.

Paraphrase ideas from your sources using your own words, but always reference or cite the original author(s).

Resist making only minor changes to the original text and using them in assignments.

Do not copy and paste into your work sections of material from the Internet, books, papers, reports, or other published work.

Do not self-plagiarise by re-using previous marked assessments and re-submitting them.

Reference or cite the sources of tables, figures, diagrams, and images used in assignments.

- 串通

Do not collude with others when writing your own individual assignment.

Create a study plan and timetable for assessments, to avoid the pressure to collude.

Do not allow your peers to access, copy, or submit your assignment as their own work.

- “合同欺骗”

Never commission assignment outlines or plans from online sources.

Do not request anyone, including friends, family, or tutors to prepare an assignment for you, whether paid or unpaid.

Never engage with 'essay mills' or custom sites to prepare assessments of any kind.

报告中的剽窃行为

- 如果是逐字摘录，*请加引号并注明出处*
- 或者，进行转述或概述，并*注明出处*
- 根据经验，这样做的句子不要超过 3-4 句
- 见 <https://integrity.mit.edu/handbook/writing-original-work>
好榜样

剽窃源代码

- 优秀指南：<https://integrity.mit.edu/handbook/writing-code>
 - 切勿复制其他学生的代码
 - 不要简单地重复使用代码
- 引用您在代码中使用的代码来源
- 确保遵守任何许可条件

自动抄袭检测

- 相似性检查工具可发现修改，例如
 - 更改变量、常量或函数/程序名称
 - 只需替换语言结构
 - 更改源代码中的缩进或分隔
 - 见 http://www.upenn.edu/academicintegrity/ai_computercode.html 更多例子



在本课程中，Codio 的剽窃检测工具
将用于检查课程作业！

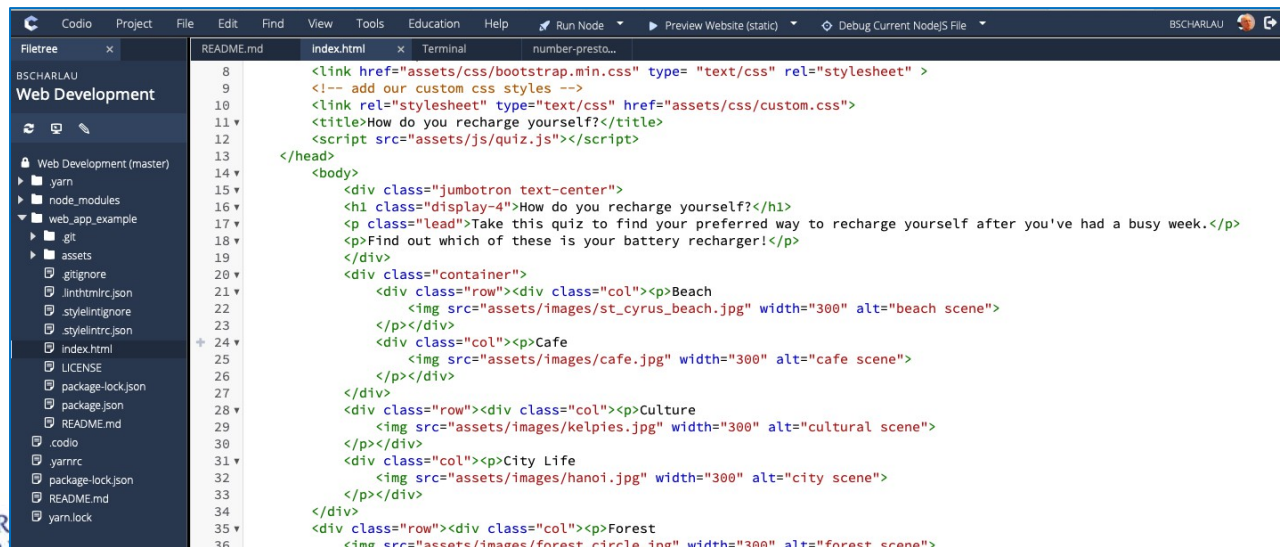
遇到困难时寻求帮助

- 如果有不清楚的地方，就去问别人或查找资料
- 如果您仍然觉得困难，请与老师联系
- 抄袭别人的东西，或通过互联网抄袭别人的东西，无助于你学习如何完成工作

提交一份差强人意的作业，并对自己的尝试和努力作出评价，总比提交别人的作业要好。

在线编码环境：Codio

- 通过 MyAberdeen 将 Codio 整合到本课程中



版本控制

- 仅在本地机器上保留代码不安全（磁盘故障）
- 如果团队成员各自开发自己版本的代码，就很难开展团队合作
- 解决方案：**版本控制**
 - 版本控制允许远程复制代码，以便共享代码和轻松回滚到早期版本
 - 在本课程中，我们将使用 **Git** 进行版本控制，并开始使用 **GitHub** 等远程仓库。

有问题或意见？

JC2002 Java 程序设计

第 1 天，第 2 课时：Java 语言简介

为什么选择 Java?

- Java 是世界上使用最广泛的计算机编程语言之一。
语言
- 对于许多组织而言，满足企业编程需求的首选语言是 Java
- 根据甲骨文在 2016 JavaOne 大会上的主题演讲
(<http://bit.ly/JavaOne2016Keynote>), 目前全球有 1000 万 Java 开发人员, 150 亿台设备上运行着 Java, 其中包括 20 亿辆汽车和 3.5 亿台医疗设备

- 安卓是 Java 的一种

Java 历史

- Java 是一种古老的语言
 - 1991 年, Sun Microsystems 资助了一个由 James Gosling 领导的公司内部研究项目, 该项目产生了一种基于 C++ 的面向对象编程语言, Sun 称其为 Java
- 互联网帮助 Java 发展
 - 由于人们对互联网的极大兴趣, Java 引起了商业界的关注
 - Java 程序可在多种计算机系统和计算机上运行。
受控设备 ("一次写入, 随处运行": 详情稍后介绍)
- 现已用于开发大型企业应用程序、增强网络服务器的功能、为消费类设

备提供应用程序、开发机器人软件以及许多其他用途

Java 版本

- 目前使用的 JAVA 版本各不相同；最新版本的标准版 (SE) Java 开发工具包 (JDK) 是 20
 - 长期支持 (LTS) 的最新版本是 JDK 17
 - 并非所有版本都一样；本课程主要基于长期支持 (LTS) 的 Java 11
 - 对于本课程中的基础概念，任何 Java 8 版本的代码看起来都几乎相同
- 较旧的 Java RE 运行时可能无法运行为较新版本编写的代码，较

新的运行时可能缺少为较旧程序编写的程序所需的库！

Java 批评

- 过于冗长
 - 不过，更容易阅读！
 - 当您响应停机呼叫，或需要维护和修补由早已离职的开发人员编写的代码时，额外的冗余可能会给您带来好处
- 变化缓慢
 - 最近版本中出现的新语言功能是朝着解决最常见的功能缺失问题迈出的重要一步

- 性能低下
 - 对于早期版本来说是如此，但不再是限制因素

Java 批评（续）

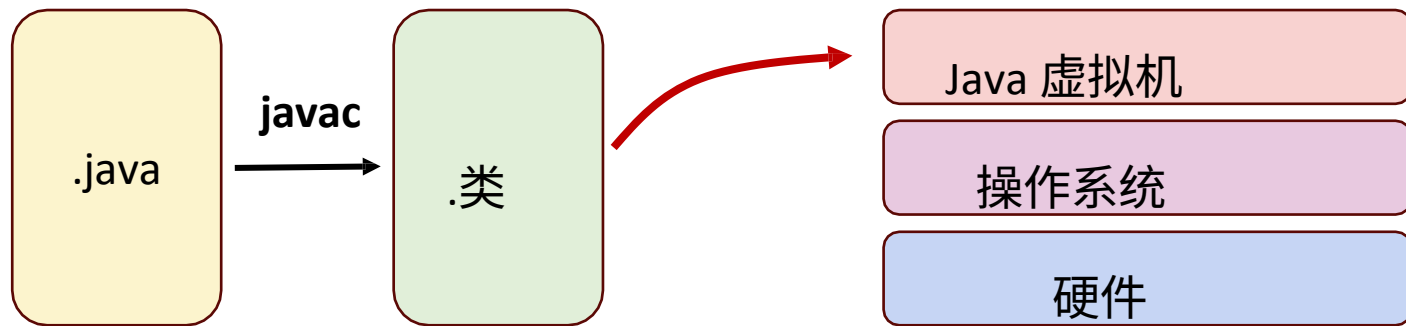
- 过于冗长
- 变化缓慢
- 性能低下
- 安全问题
 - 2013 年，Java 平台出现了大量安全漏洞，导致 Java 8 的发布日期被推迟
 - 其中许多漏洞涉及 Java 系统的桌面和图形用户界面组件，并不影响用 Java 编写的网站或其他服务器端代码

- 过于企业化

- 事实上，Java 是一种广泛应用于开源软件项目的语言

Java 虚拟机 (JVM)

- JVM 是一个提供 "运行时环境" (或执行环境) 所必需的



- 要将 .java 文件编译成 .class 文件, 请使用 `javac Program.java`

- 要运行 .class 文件，请使用 `java <arguments>` 程序

JVM 的优势

- 包含一个供应用程序代码在其中运行的容器
- 提供安全可靠的执行环境（与 C/C++ 相比）
- 将内存管理从开发人员手中解放出来
- 提供跨平台执行环境，即 "一次编写，随处运行"（WORA）
- 利用运行时信息进行自我管理，即适时管理 (JIT) 编译

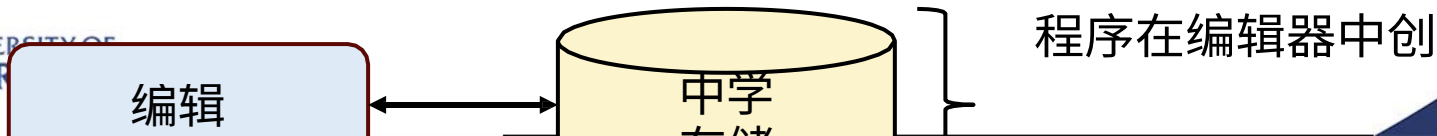
典型的 Java 开发环境

- 通常，Java 程序分为五个阶段发展：
 - 第 1 阶段：**编辑**代码
 - 第 2 阶段：**编译**为 *字节码*
 - 第 3 阶段：**加载**字节码
 - 第 4 阶段：**验证**字节码

- 第 5 阶段：**执行**字节码

第 1 阶段：创建计划

- 第 1 阶段包括使用 *编辑器程序* (编辑器) 编辑文件
- 使用编辑器，您可以
 - 键入 Java 程序 (源代码)
 - 进行必要的更正
 - 保存在辅助存储设备上



建，并存储在以
.java 结尾的文件
中。

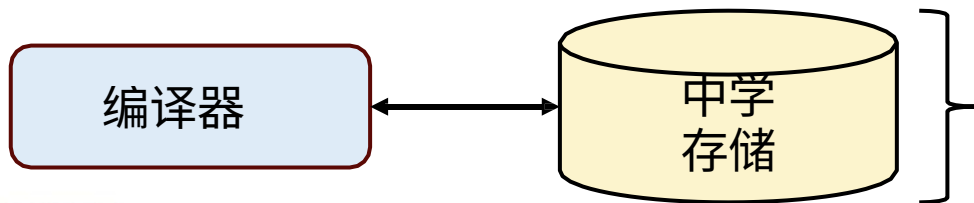
第 1 阶段：编辑程序文件

- 您可以使用任何文本编辑器编辑源代码文件（Vim、记事本、TextEdit 等）。
- 您还可以使用*集成开发环境*（IDE）
 - 提供支持软件开发过程的工具，如编辑器、查找逻辑错误的调试器等。
 - 最流行的 Java IDE 包括
 - 日蚀 (<http://www.eclipse.org>)
 - IntelliJ IDEA (<http://www.jetbrains.com>)
 - NetBeans (<http://www.netbeans.org>)

第 2 阶段：编译 Java 程序

- 使用 `javac` (Java 编译器) 命令编译源代码到程序
- 要编译源文件 `Welcome.java`，您可以键入

`javac Welcome.java`



编译器创建字节码
并将其存储在以
.class 结尾的文件

中

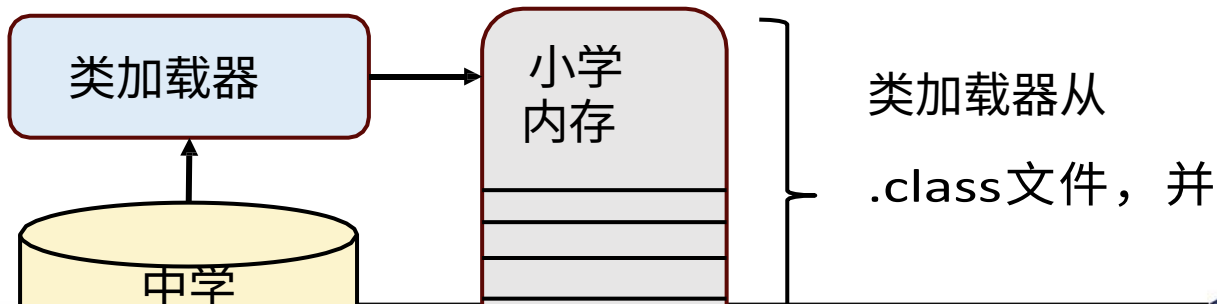
第 2 阶段：编译字节码

- Java 编译器将 Java 源代码翻译成字节码，这些字节码包括代表执行阶段要执行的任务
 - 虚拟机可以隐藏底层操作系统和硬件，使与之交互的程序无法识别：如果同一虚拟机在多个计算机平台上运行，那么为该类型虚拟机编写的应用程序就可以在所有这些平台上使用
 - JVM 是 JDK 的一部分，也是 Java 平台的基础，它执行字节码
- 因此，Java 的字节码是可移植的，相同的字节码指令可以在任何包含 JVM 的平台上执行，JVM 可以理解编译字节码的

Java 版本。

第 3 阶段：将程序载入内存

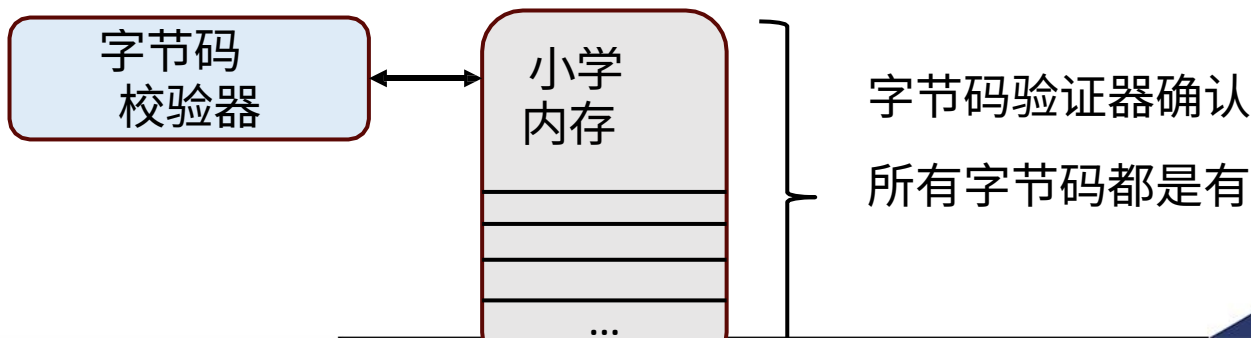
- JVM 会将程序放入内存中执行，这就是称为装载
- 类加载器接收包含程序字节码的 .class 文件，并将其传输到主内存中。它还会加载 Java 提供的、程序使用的任何 .class 文件。



将字节码放入内
存中

第 4 阶段：验证字节码

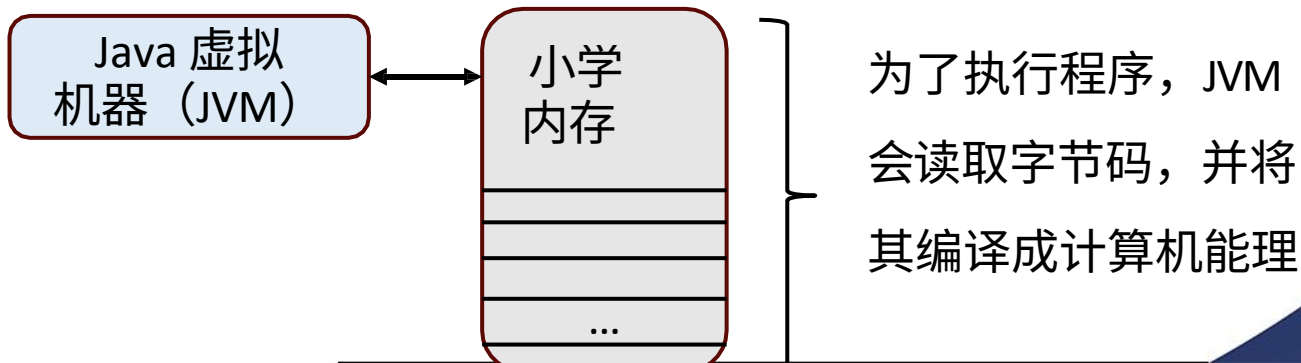
- 在加载类时，字节码校验器会检查它们的字节码，以确保它们有效且不违反 Java 的安全限制
 - Java 具有强大的安全性，可确保 Java 程序不会损坏您的文件或系统（如计算机病毒）。



效的，没有违反安
全限制

第 5 阶段：执行

- JVM 执行程序的字节码
 - 如今的 JVM 在执行字节码时，通常会结合使用解释和所谓的*即时*编译 (JIT)
 - 通过 JIT，JVM 可以在解释字节码时对其进行分析、搜索热点--频繁执行的字节码



解的语言。

第 5 阶段：采用准时制（JIT）执行

- JIT 编译器（如 Oracle 的 Java HotSpot™ 编译器）将字节码翻译成计算机的机器语言
 - 当 JVM 再次遇到这些已编译的部分时，速度更快的机器语言代码会执行
- 使用 JIT 时，Java 程序会经历两个编译阶段
 - 第一种是将源代码翻译成字节码（以便在不同计算机平台上的 JVM 之间进行移植）
 - 第二种情况是，在执行过程中，字节码被翻译成机器语言，供

实际执行程序的计算机使用。

常见错误

- 使用 javac 时，如果出现 *"Bad command or filename"*（错误命令或文件名）或 *"javac: command not found"*（找不到命令）等错误消息，则表示 Java 软件安装未正确完成。
 - 通常是 PATH 环境变量设置不当；如果出现这种情况，请仔细查看安装说明
 - 在某些系统上，更正 PATH 后需要重启计算机，以使更改生效

- 使用 java 运行 .class 文件时，如果出现诸如 `"java.lang.NoClassDefFoundError"` (`java.lang.NoClassDefFoundError`) 的错误信息，通常意味着 Java CLASSPATH 环境变量未正确设置

Java Classpath

- Java 解释器需要知道在哪里查找类（.jar 或 .class文件），这些文件不属于核心Java
 - 类路径定义了查找外部字节码文件的位置
- 设置 Classpath 的两个选项：
 - 定义 CLASSPATH 环境变量：

`export CLASSPATH=./path/to/external/library.jar` (LINUX)

`set CLASSPATH=./path/to/external/library.jar` (Windows)

- 使用命令行 switch -cp 或 -classpath 和 java 命令：

```
java -classpath ./path/to/external/library.jar ProgramName arg
```

```
java -cp ./path/to/external/library.jar ProgramName arg
```

有问题或意见？

JC2002 Java 程序设计

第 1 天，第 3 节：版本控制

本场会议的参考资料

- Evans, B. and Flanagan, D., 2018. *Java in a Nutshell: 桌面快速参考 第7 版*。O'Reilly Media。
- Deitel, H., 2018. *Java 如何编程, 早期对象, 全球版, 第11 版*。皮尔逊
- 本-林恩, 《Git Magic》, 2007 年, 可在线查阅: <http://www-cs-students.stanford.edu/~blynn/gitmagic/book.html>
- 汤姆-普雷斯顿-维尔纳, 《Git 寓言》, 2009 年, 可上网查阅: <https://tom.preston-werner.com/2009/05/19/the-git-parable.html>
- Johan Herald, NDC TechTown, 2008 年, 可查阅
在线: <https://docs.google.com/presentation/d/1u0cM0r07iL9v7Myo6RWGWRR3>

[o2IYlebDlayG8rMagVw/edit#slide=id.g3bcc4c1a94_0_6](#)

JAR 文件

- 编译后的 JAVA 程序（即 .class 文件）可打包到 .jar 文件
- 第三方库通常以 .jar 文件（通过 Maven）的形式发布
 - 稍后，当我们开始对项目依赖关系进行自动化管理时，我们将讨论 Maven。
- 您也可以将自己的应用程序打包为可执行的 .jar 文件。
 - 运行 .jar 文件： `java -jar example.jar`

- 有关使用 .jar 文件的更多信息：

<https://docs.oracle.com/javase/tutorial/deployment/jar/basicsindex.html>

版本控制

- 版本控制为我们提供了撤消功能，以便我们进行更改有安全网
- 版本控制有多种类型：
 - 本地（如 RCS）
 - 基于服务器（如 Subversion）
 - 分布式（如 Git）
- 本课程的重点是 Git

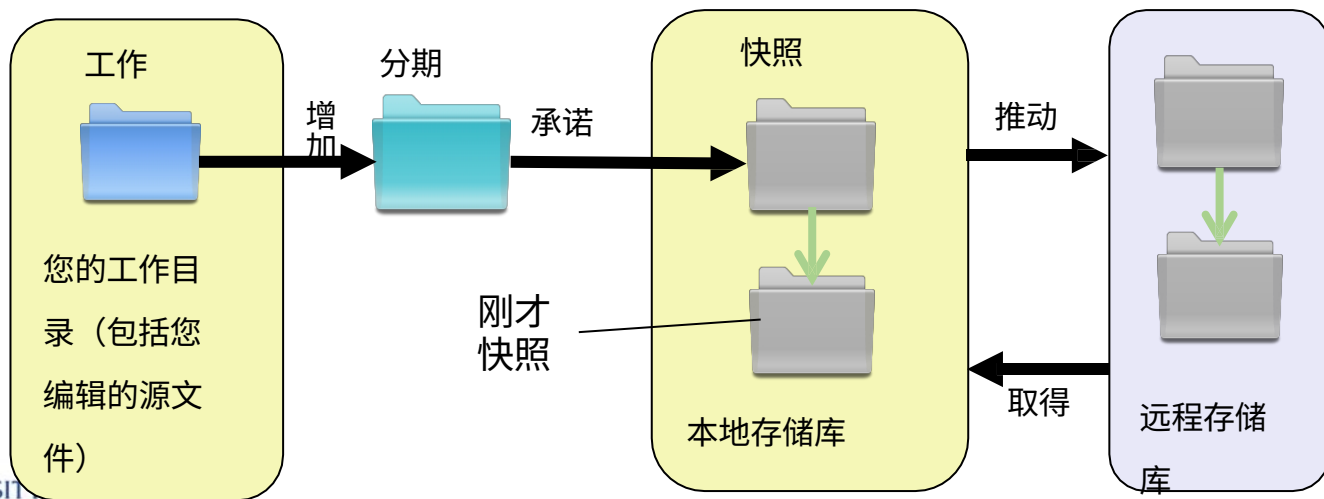
什么是 Git?

- 开源分布式版本控制系统
 - 原作者 Linus Torvalds
 - 为开发 Linux 内核而创建
 - 当今最流行的版本控制系统
 - 由 *github.com* 等许多常用服务提供商提供支持、*bitbucket.com* 等。
- Git 为这两方面都提供了手段：

- 一个全球性的 "如果"
- 项目中的全局 "撤消"

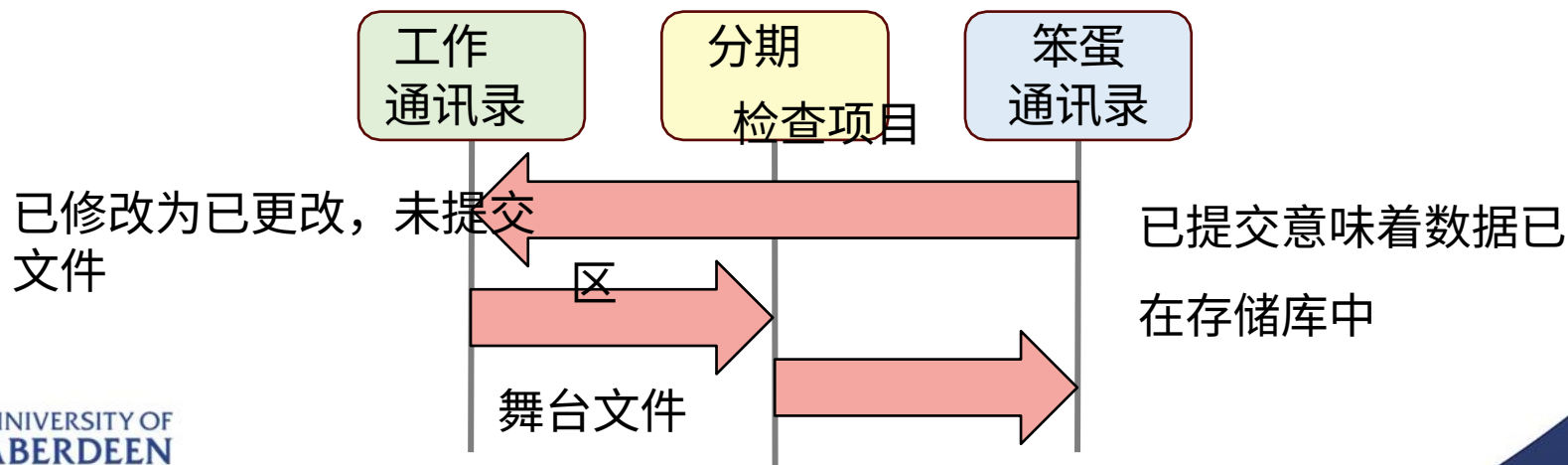
Git 快照

- 在 Git 中，快照是项目文件在某一时刻的状态记录。
时点



Git 行动区

- Git 有三个运行区域：工作目录、暂存区域和 git 目录（仓库）

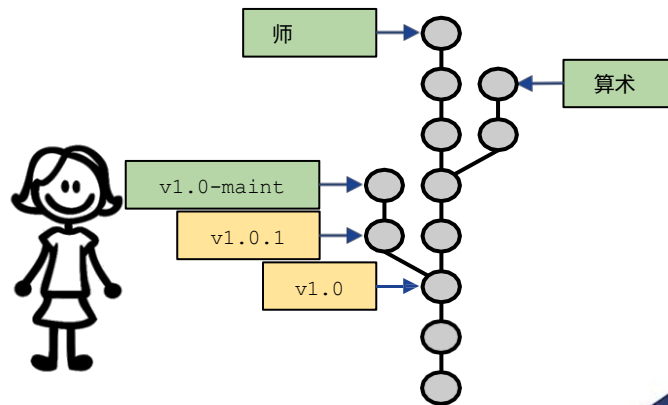
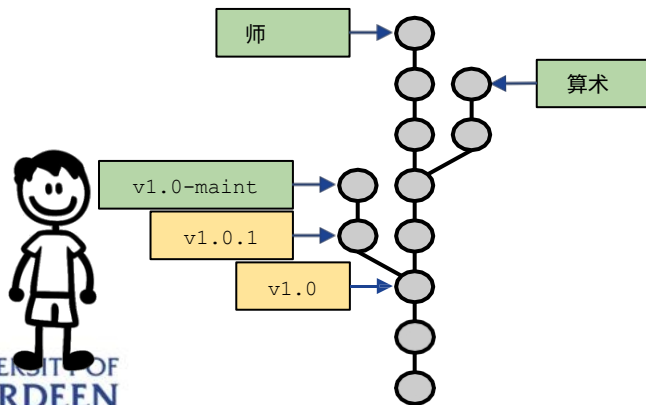


承诺

暂存的是已修改的文件，标记为
进入提交快照

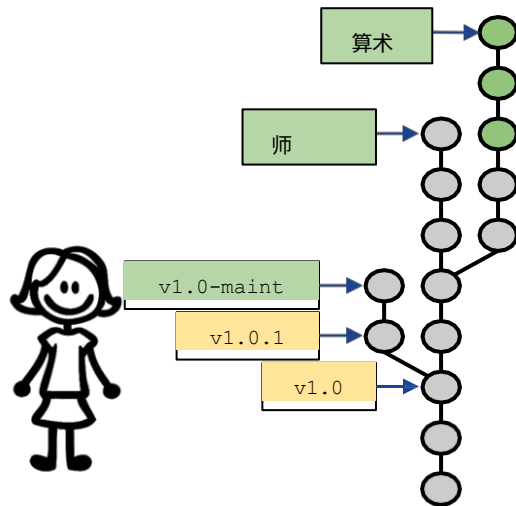
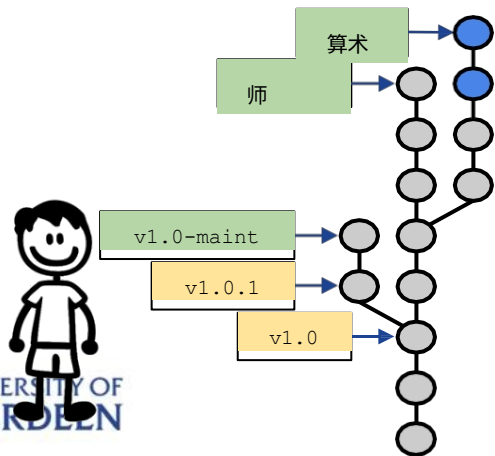
快照和分支

- 鲍勃和爱丽丝从同一个本地目录开始



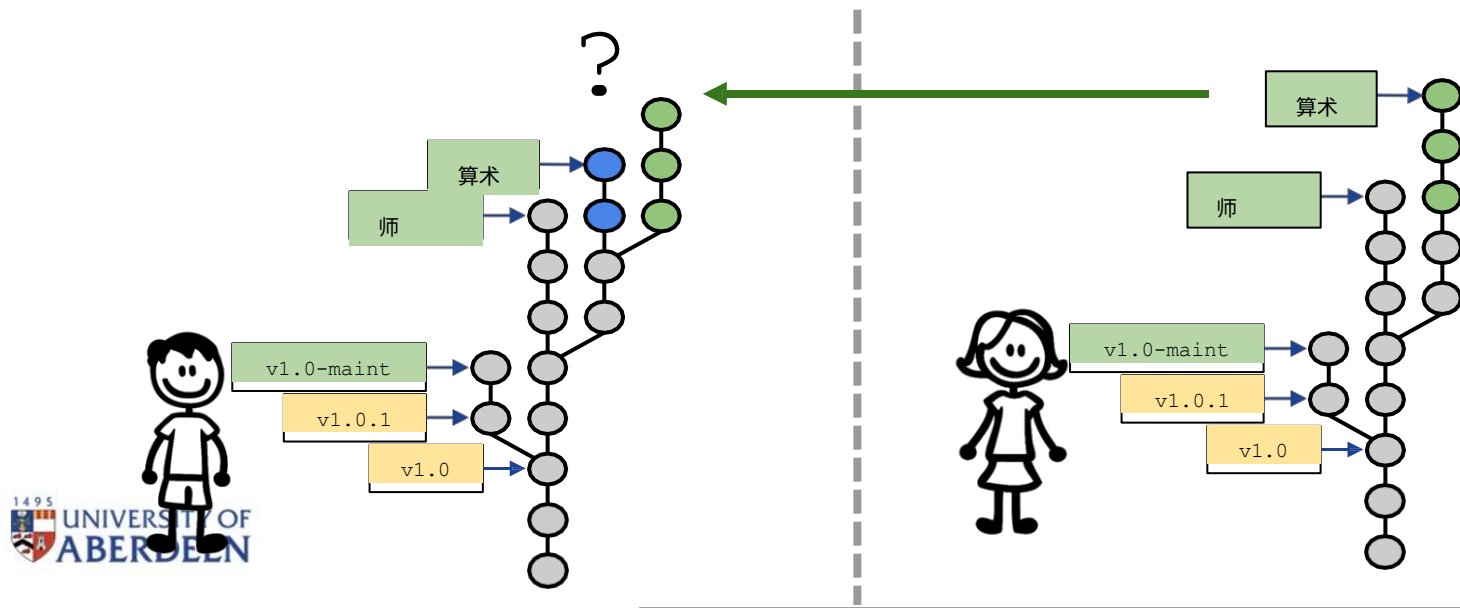
合并

- 鲍勃和爱丽丝进行本地更改

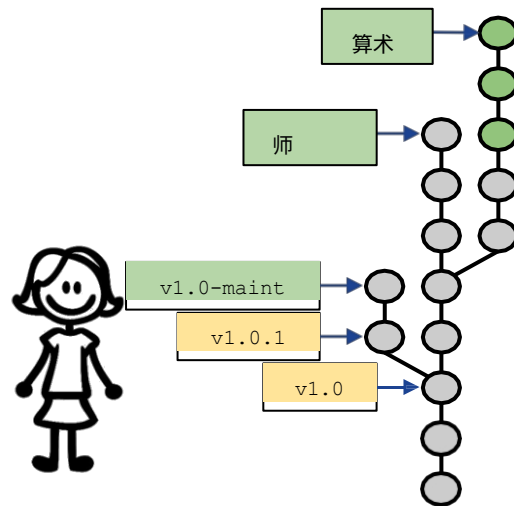
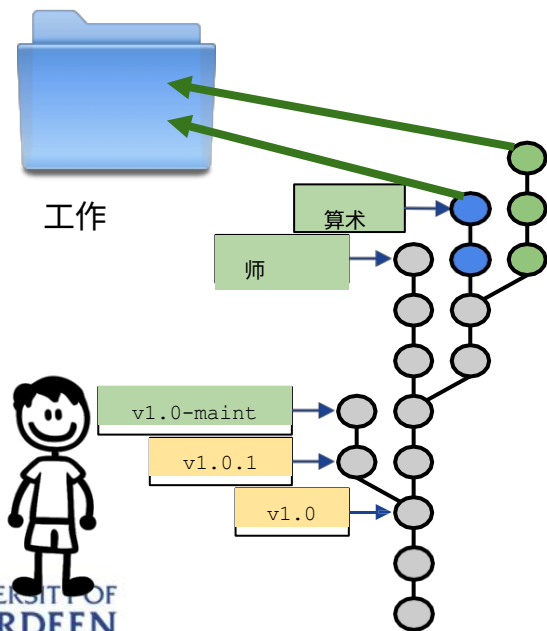


合并

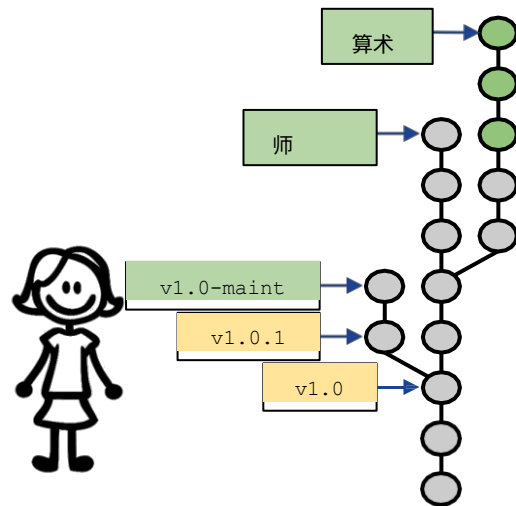
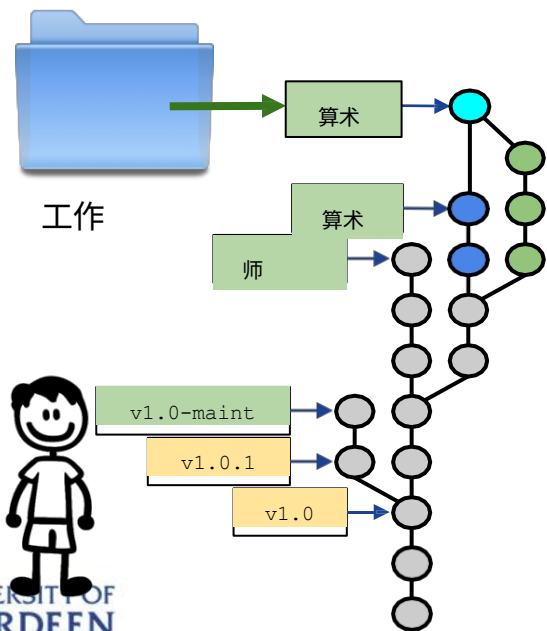
- 如何合并 Bob 和 Alice 的本地更改？



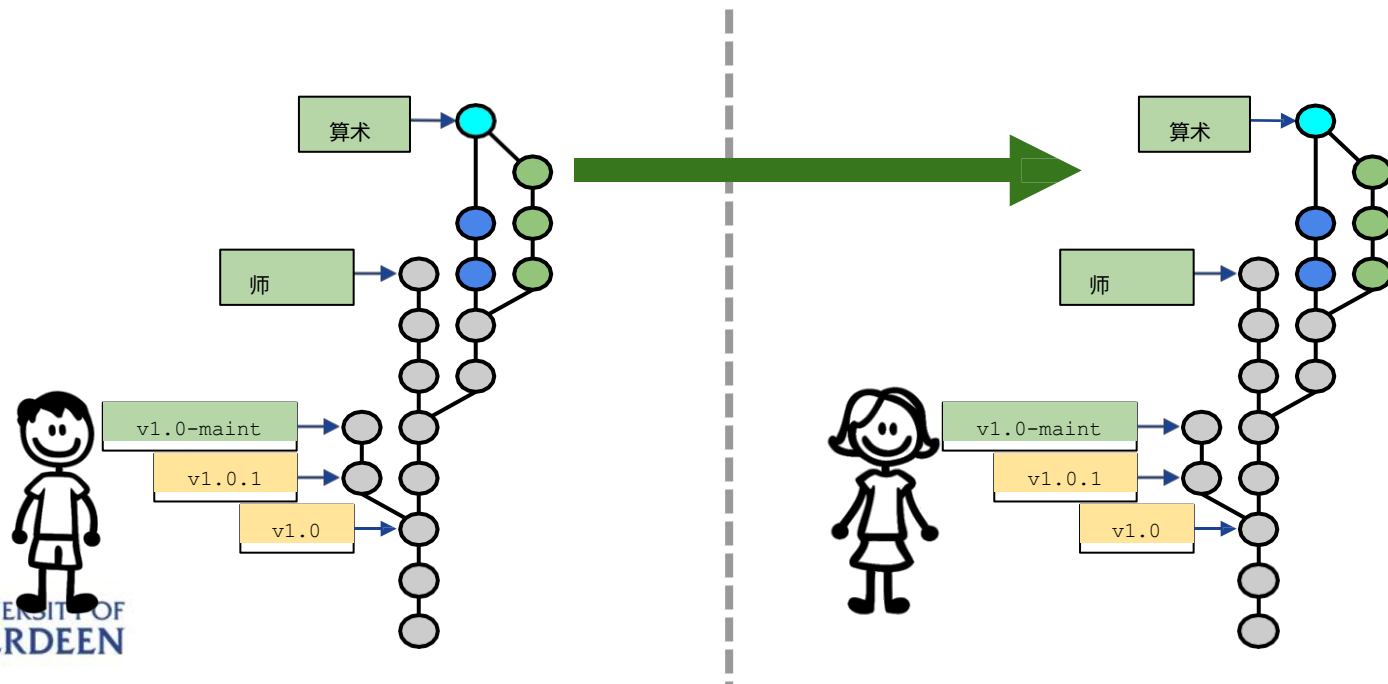
合并



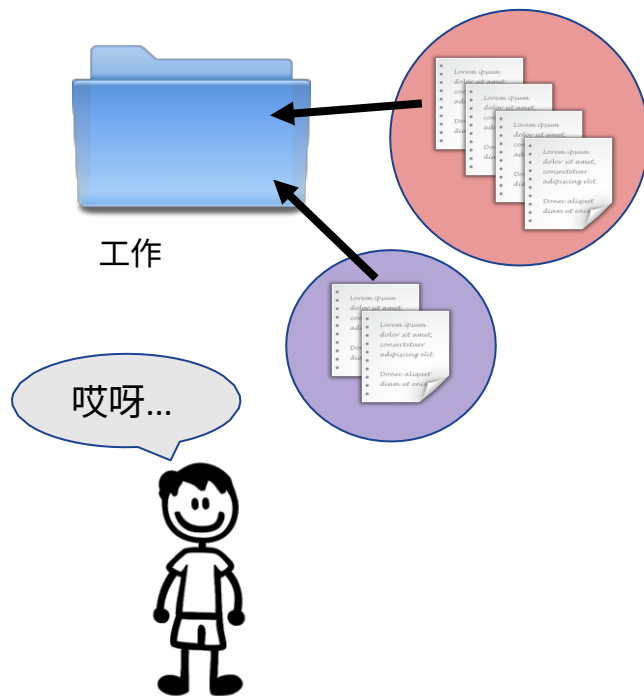
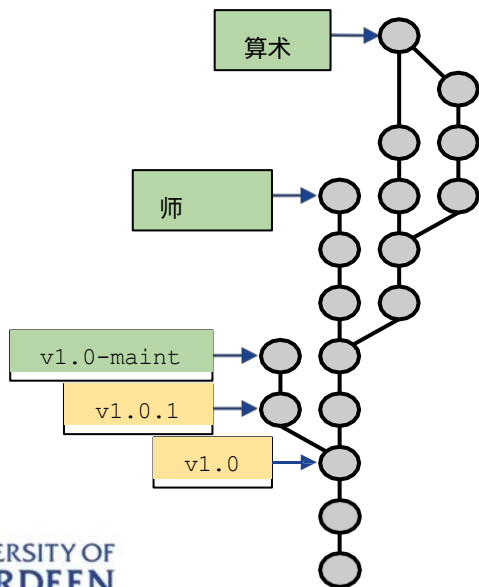
合并



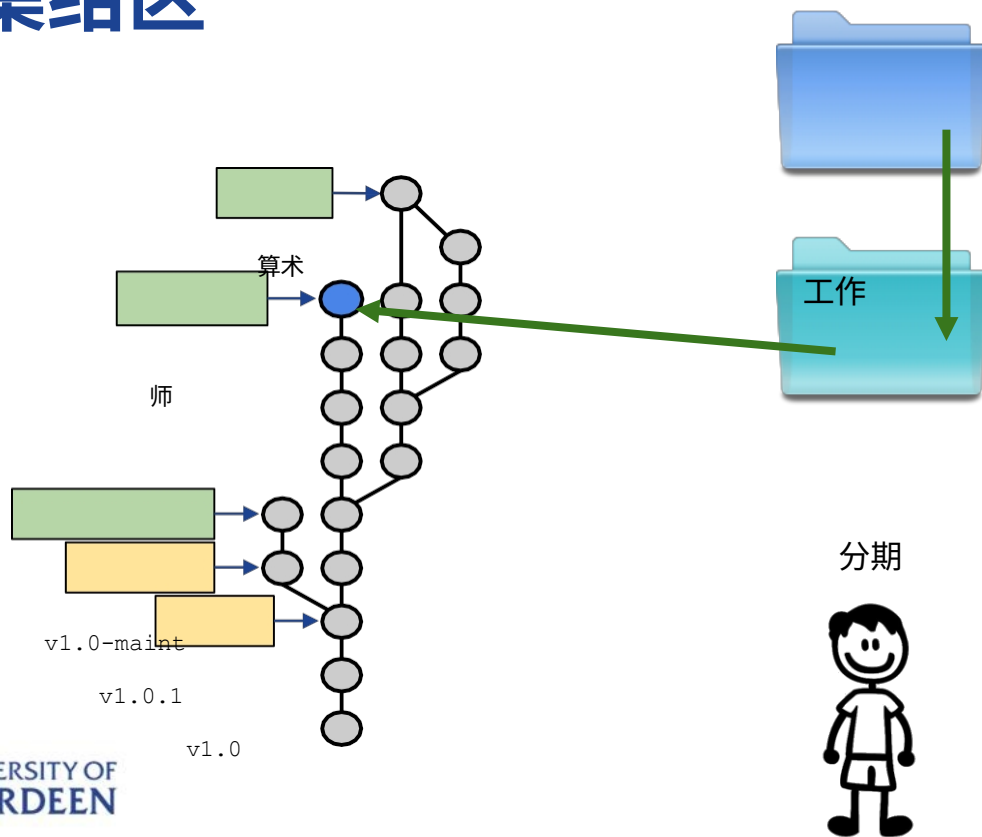
合并



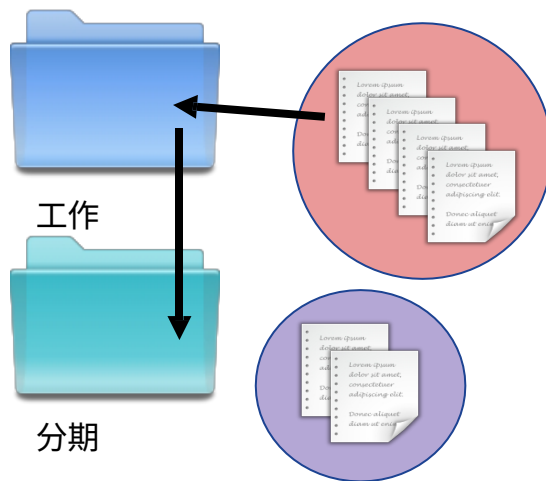
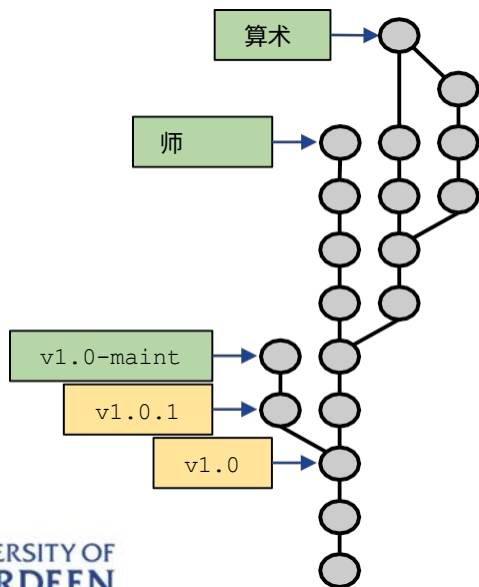
集结区



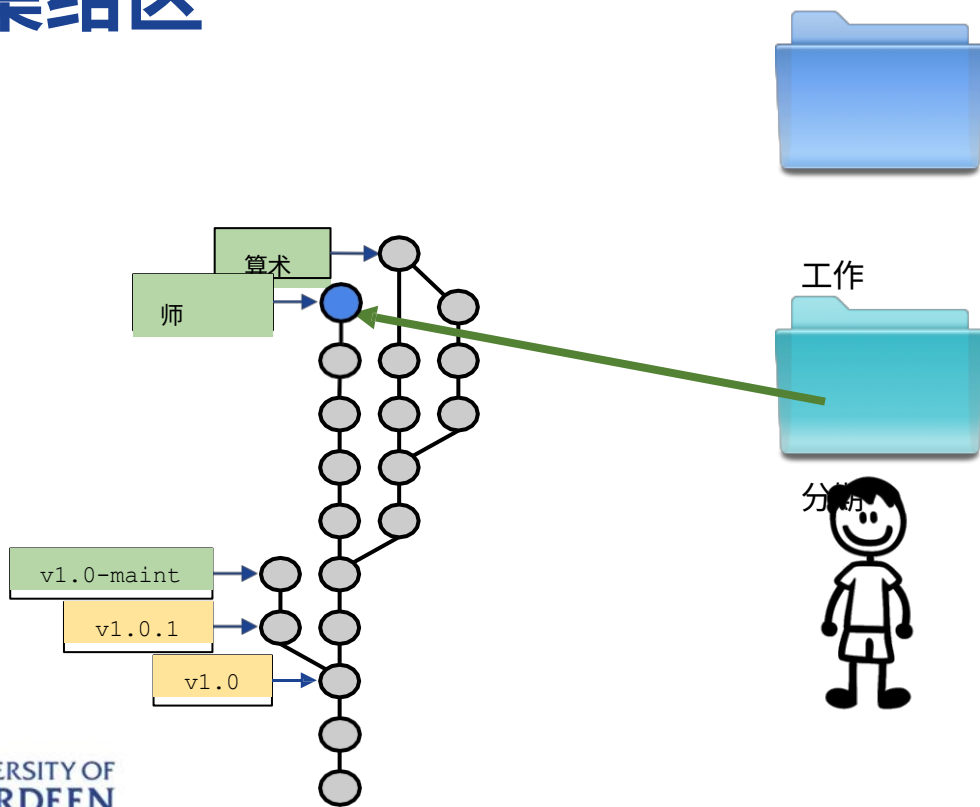
集结区



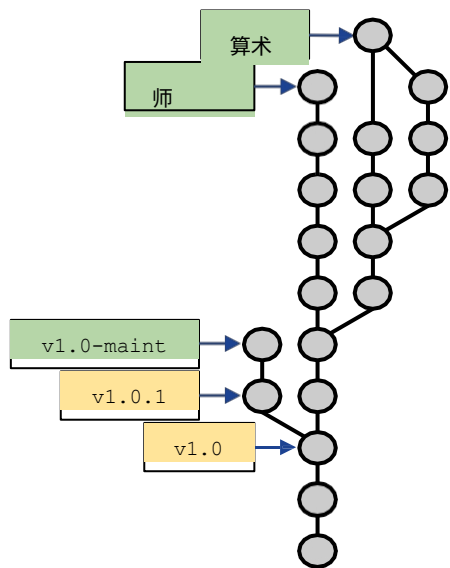
集结区



集结区



差异



工作



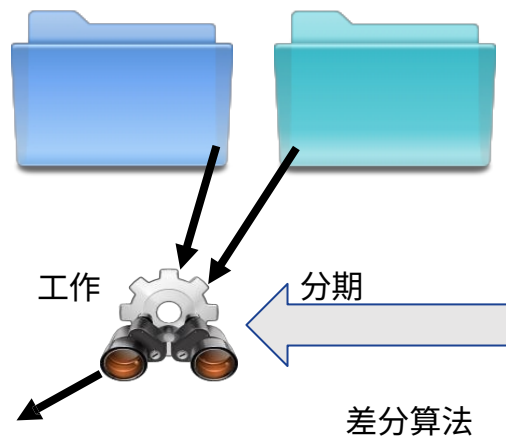
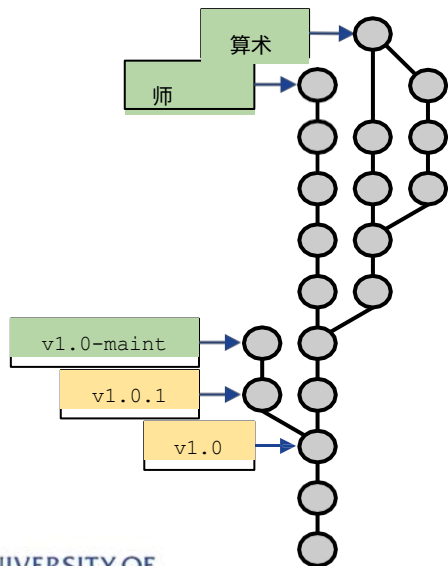
分期

有哪些变化?

- 工作与分期?
- 工作与快照 X?
- 分期与快照 X?
- 快照 X 与快照 Y?



差异：工作与暂存



----分期

+++ 工作

动物

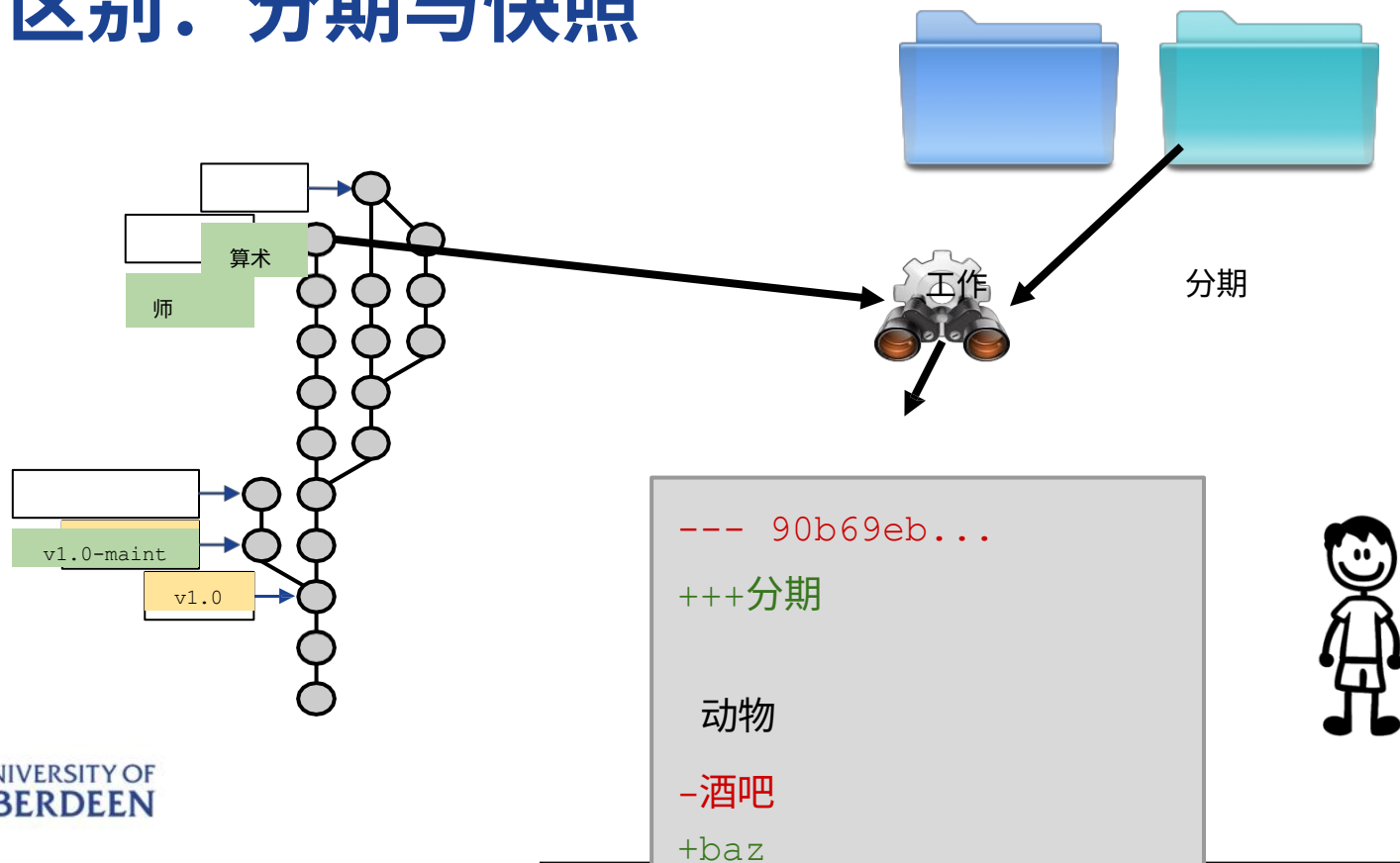
-巴兹

+bazzle

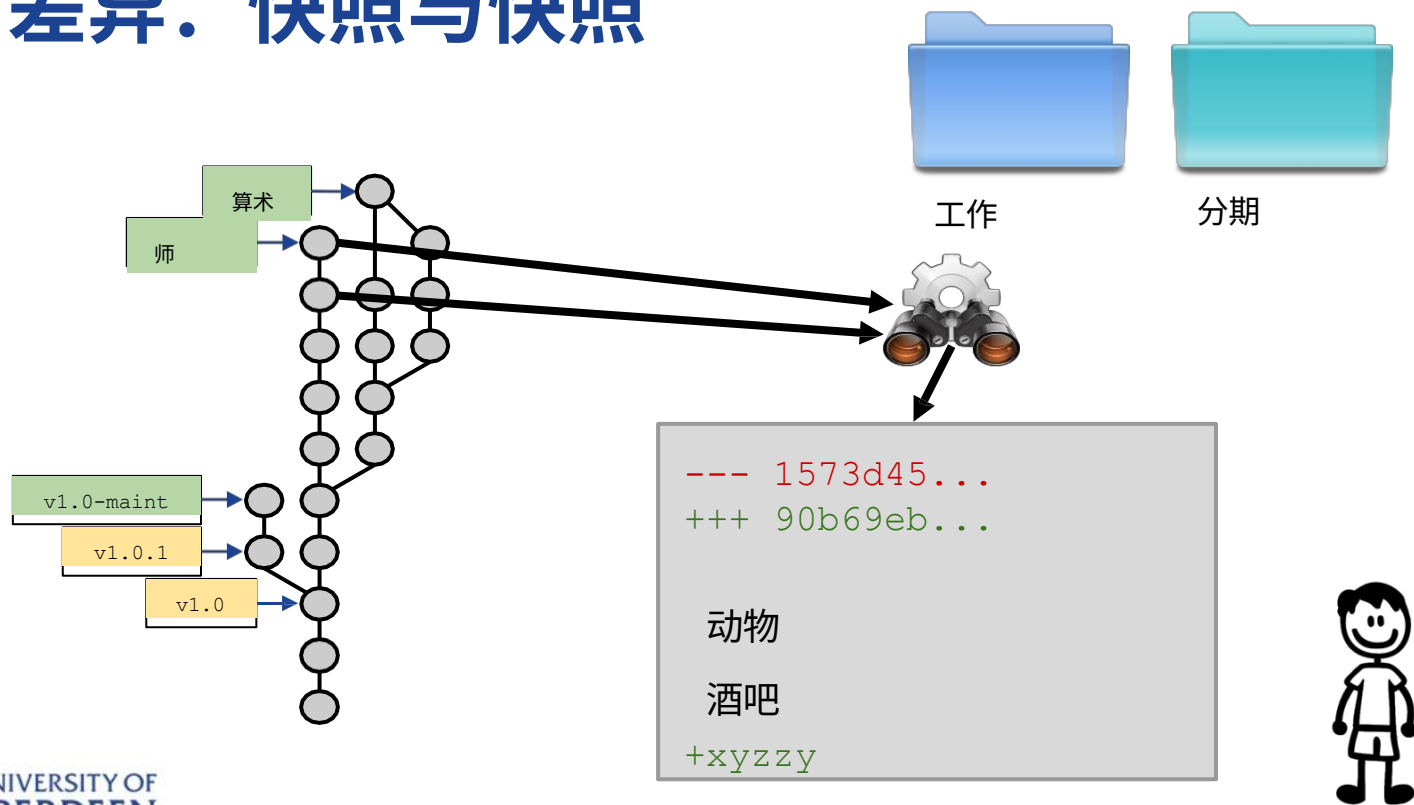
xyzzzy



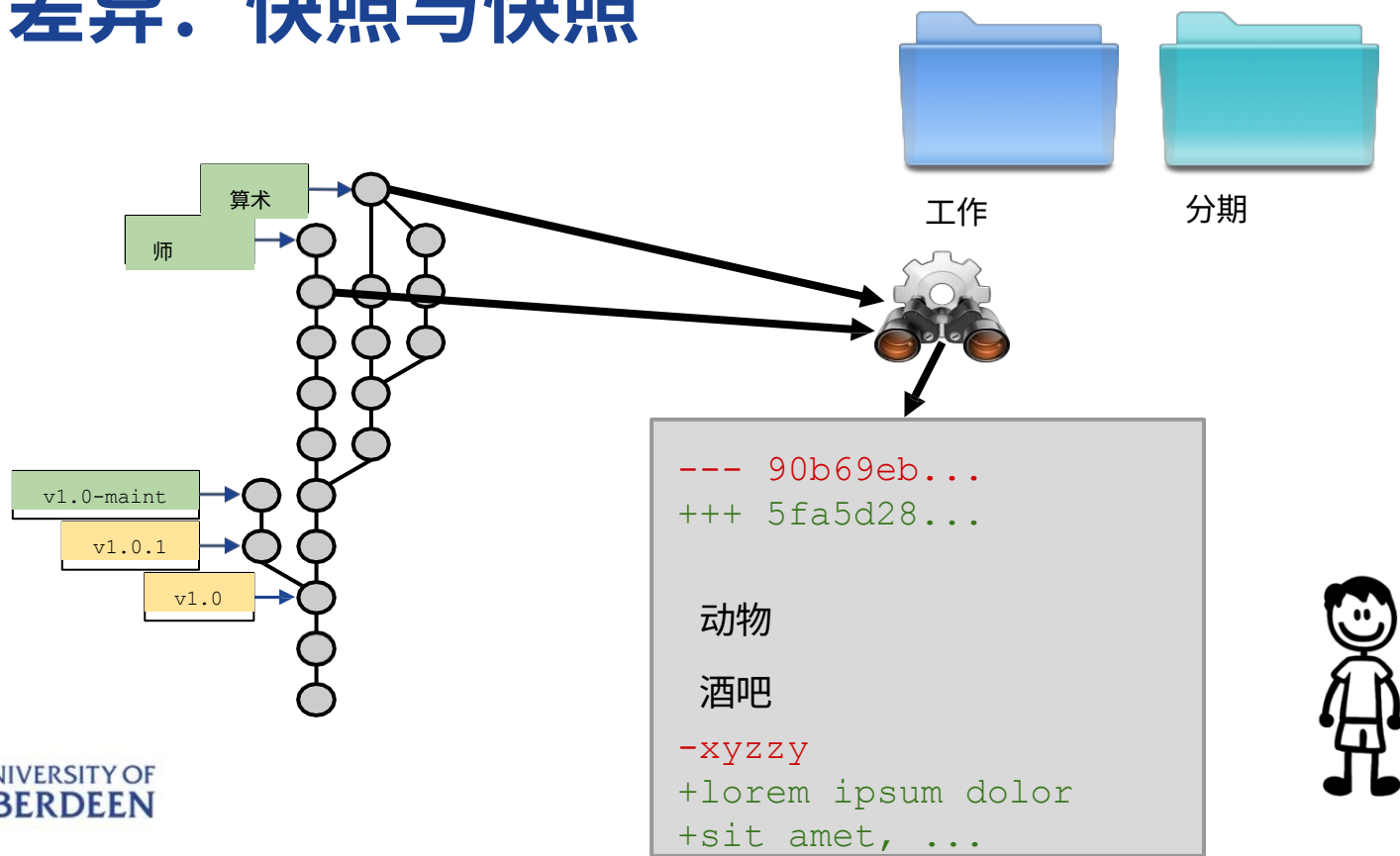
区别：分期与快照



差异：快照与快照



差异：快照与快照



Git 命令：入门

- 首先，告诉吉特你是谁：

```
git config --global user.name "我的名字"
```

```
git config --global user.email "my@email.address"
```

- 寻求帮助：

```
git <command> -h
```

```
git help <命令>
```

- 启动一个新的 Git 仓库：

```
git init
```

获取、合并、推送

git remote add <name> <URL>

push origin <name>

git fetch <name>

git

merge



<name>/<branch>

git



clone <URL> <project> git

git pull

git init <项目

> cd <项目

git remote add origin

<URL> git fetch

origin

git checkout master

添加分支和标记

git 分支

git branch <branch>

git checkout <branch>

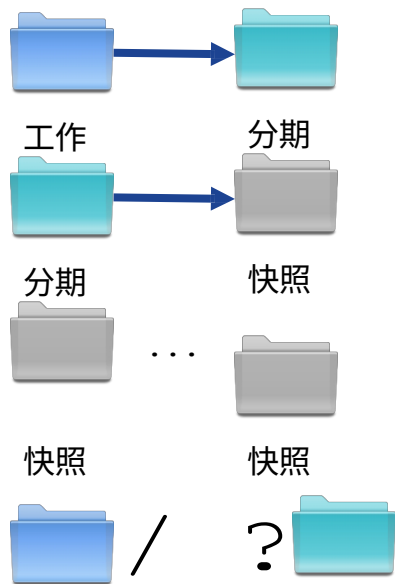


git checkout -b <branch>

git tag -l

git 标签 <tag>

制作快照



工作分期

git add

git commit

git 日志

git commit -a

gitk

- Add the simple scripts I used to do a merge with con
- Merge the new object model thing from Daniel Barkal
- [PATCH] Switch implementations of merge-base, port
- [PATCH] Port fsck-cache to use parsing functions
- [PATCH] Port rev-tree to parsing functions
- [PATCH] Implementations of parsing functions
- [PATCH] Header files for object parsing
- [PATCH] fix bug in read-cache.c which loses files when
- [PATCH] Fix confusing behaviour of update-cache --re
- Make "commit-tree" check the input objects more car
- Make "parse_commit" return the "struct revision" for t
- Do a very simple "merge-base" that finds the most re
- Make "rev-tree.c" use the new-and-improved "mark_re

冲突

- 合并两个独立修改了同一文件的分支，将导致冲突
- 如果两个人在同一个分支中独立处理同一个文件，那么第二个人的提交就会产生冲突
- 冲突可手动解决
- 定期提取最新更改，避免冲突

`git pull <branch name>`

diffs 命令



对



工作

分期



对



分期



对



工作

对

`git diff`

`git diff --staged`

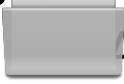
快照

快照



快照

git diff

i  AD

快照

git diff <from> <to>

关于 Git 的更多信息

- 哦，我的 Git! : <https://ohmygit.org> (一款关于学习 Git 的游戏)
- Git 主页: <http://git-scm.com>
- Pro Git: <http://git-scm.com/book>
- GitHub: <http://github.com>
- 学习 Git 分支: <https://learngitbranching.js.org>
- Git Ready: <http://gitready.com>

有问题或意见？

JC2002 Java 程序设计

第 1 天，第 4 课：Java 语言基础

参考文献和学习目标

- 大部分材料基于《**Java：如何编程**》的幻灯片、第 2、4、5、7 章，可通过 MyAberdeen 获取
- 本课结束后，您应该能够
 - 编写简单的 Java 控制台程序，将输出打印到控制台并接收来自键盘的用户输入
 - 使用变量、基本算术运算和比较

如何编写基本的 Java 程序？

- Java 程序以类的形式组织，其中包括方法
 - 基本规则是，公共 Java 类的源代码必须保存在与公共类同名的 .java 文件中
 - 类和方法将在后面详细说明
- 每个 Java 程序都需要一个带有公共静态作为程序起点的方法 `main()`
 - 当使用 `java MyClass` 命令启动程序 `MyClass` 时，程序会在类 `MyClass` 的方法 `main()` 中启动

声明类和方法

- 声明类的基本语法：

public static class MyClass { ... }

修改器
可选

关键词
类

类
名字

班级主体
(在大括号内)

- 方法（在类内）声明的基本语法：

protected final void myMethod(int a, int b) { ... }

修改器

可选

返回

类型

(如果为空，则使用

void)

方法
名字

参数 方法主体
(可以为 (在大括号内
空))



简单 Java 控制台程序示例 (1)

```
1 // 示例文本打印 Java 程序 Welcome1.java
2
3
4 公共类 Welcome1 {
5     // main 方法开始执行 Java 应用程序 public static void
6     main(String[] args) {
7         System.out.println("Welcome to Java programming!");
8     } // 主体部分结束
    } // end class Welcome1
```

简单 Java 控制台程序示例 (2)

```
1 // 示例文本打印 Java 程序 Welcome1.java 2
3 公共类 Welcome1 {
4     // 主方法开始执行 Java 应用程序
5     public static void main(String[] args) {
6         System.out.println("Welcome to Java programming!");
7     } // 主体部分结束
8 } // end class Welcome1
```

您可以使用 `//` 或 `/* ... */`
添加编译器忽略的注释。

简单 Java 控制台程序示例 (3)

```
1 // 文本打印 Java 程序示例 我们
2
3 公共类 Welcome1 {
4
5     // main 方法开始执行 Java 应用程序 public static void
6     main(String[] args) {
7         System.out.println("Welcome to Java programming!");
8     } // 主体部分结束
    } // end class Welcome1
```

每个 Java 程序都需要

至少一门课

方法 main 是启动
计划要点

简单 Java 控制台程序示例 (4)

```
1 // 示例文本打印 Java 程序 Welcome1.java
2
3
4 公共类 Welcome1 {
5     // main 方法开始执行 Java 应用程序 public static void
6     main(String[] args) {
7
8 } // end class Welcome1
```

内置方法 **System.out.println** 是
用于在控制台中打印一行文本

简单 Java 控制台程序示例 (5)

```
1 // 示例文本打印 Java 程序 Welcome1.java 2
3 公共类 Welcome1 {
4     // 主方法开始执行 Java 应用程序
5     public static void main(String[] args) {
6         System.out.println("Welcome to Java programming!");
7     } // 主体部分结束
8 // end class Welcome1
```

```
$ javac Welcome1.java
$ java Welcome1
```

欢迎使用 Java 编程!

编译文件 Welcome1.java 中的代码

运行已编译的程序 Welcome1

程序打印输出

简单 Java 控制台程序示例 2

```
1 // 示例文本打印 Java 程序 Welcome2.java
2
3
4 公共类 Welcome2 {
5     // main 方法开始执行 Java 应用程序 public static void
6     main(String[] args) {
7         System.out.print("Welcome to ");
8         System.out.println("Java programming!");
9     } // 主体部分结束
    } // end class Welcome2
```

```
$ java Welcome2
```

```
欢迎使用 Java 编程!
```

```
$
```

简单 Java 控制台程序示例 3

```
1 // 示例文本打印 Java 程序 Welcome3.java
2
3 公共类 Welcome3 {
4     // 主方法开始执行 Java 应用程序
5     public static void main(String[] args) {
6         System.out.println("Welcome\nto\nJava\nprogramming!");
7     } // 主体部分结束
8 } // end class Welcome1
```

```
% java Welcome3
```

```
欢迎来到
```

```
Java
```

```
编程
```

```
$
```

逃生序列

- 您可以使用 *转义序列* 格式化控制台输出

逃生序列	说明
<code>\n</code>	换行将屏幕光标置于 下一行的开头。
<code>\t</code>	水平制表符。将屏幕光标移动到下一个制表符位置。
<code>\r</code>	回车将光标置于当前行的开头。任意后输出的字符将覆盖当前行上的字符。
<code>\\</code>	反斜线。用于打印反斜杠字符 (\)。

\"

双引号。用于打印双引号字符（"）。

用 printf 格式化输出

- 使用 `System.out.printf` 方法将输出打印到控制台
 - `f` "表示"格式化": `printf` 显示 *格式化数据*
- 参数以 *逗号分隔的列表形式* 列出
- 调用方法也称为 *调用方法*
- Java 允许将大型语句分割成多行
 - 但是，不能在标识符或字符串中间分割语句

printf 的参数

- 方法 printf 的第一个参数是 *格式字符串*
 - 可由 *固定文本*和 *格式规范*组成
 - 固定文本的输出方式与打印或 println 方法相同
 - 每个格式指定符都是一个值的占位符，用于指定要输出的数据类型
- 格式指定符是一个百分号 (%), 后面跟一个表示数据类型的字符
- 对于字符串，%s 是占位符，对于 int，%d 是占位符

- 其他占位符：**%f** 表示浮点数，**%b** 表示**布尔值**

格式化打印示例

```
1 // 格式化打印 Java 程序示例
2
3 公共类 Welcome4 {
4     // 主方法开始执行 Java 应用程序
5     public static void main(String[] args) {
6         System.out.printf("%s\n%s\n", "Welcome to", "Java programming!");
7     } // 主体部分结束
8 } // 结束类 Welcome4
```

欢迎来到

Java 编程!

- 注意 `\n` 和 `%n` 都可用于换行

在 Java 中声明变量

- 在 Java 中，变量在使用前需要声明
- 声明带默认值变量的基本语法：

~~public int number;~~

修饰符（可选） 数据类型 变量名

- 声明变量和赋值的基本语法：

私用 double number = 5.8;

修改器 (可选)

数据类型变量

名称指定值

Java 中的原始类型

- 原始类型不是从其他数据类型派生而来的

类型	大小 (位)	数值范围
布尔	1	真假
烧焦	16	0 至 65535
字节	8	-128 至 +127 (-2^7 至 $+2^7 - 1$)
短	16	-32,768至+32,767 (-2^{15} 至 $+2^{15} - 1$)
int	32	-2^{31} 至 $+2^{31} - 1$ (约 $-2 \cdot 10^9$ 至 $+2 \cdot 10^9$) ⁹
长	64	-2^{64} 至 $+2^{64}-1$ (约 -10^{19} 至 $+10^{19}$)

浮动	32	约 (+/-) $1.4 \cdot 10^{-45}$ 至 $3.4 \cdot 10^{38}$
双人	64	约 (+/-) $4.9 \cdot 10^{-324}$ 至 $1.8 \cdot 10^{308}$

输入和整数的格式化打印

```
1 // 带有输入的格式化打印 Java 程序示例
2 import java.util.Scanner; // 需要输入
3 公共类 加法 {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Enter first integer: "); // 提示
7         int number1 = input.nextInt(); // 读取第一个数字
8         System.out.print("Enter second integer: "); // 提示
9         int number2 = input.nextInt(); // 读取第二个数字
10        int sum = number1 + number2; // 将数字相加并存储结果
11        System.out.print("Sum is: %d\n", sum);
```

```
12         } // 主体部分结束
```

```
13     } // 结束加法类
```

输入第一个整数：42 输入第二个整

数： 88 总和为： 130

导入类

- 默认情况下，每个 Java 程序都会导入 **java.lang** 包；因此，**java.lang** 中的类（如 **System**）是唯一不需要导入的类
- 在上一个示例中，需要使用**扫描仪**类来启用读取用于程序的数据的程序
 - 数据有多种来源，例如键盘上的用户或磁盘上的文件
 - 使用**扫描仪**之前，您必须创建**扫描仪**并指定来源数据

二进制溢出

- 在其他一些编程语言（如 C 语言）中，变量可以溢出其分配的位数范围
 - 例如，对于字节， $127+1$ 的结果是 -128
- 在 Java 中，这种情况通常是可以避免的，并且会出错，但对于 `int` 和 `long`，二进制溢出仍然可能发生。对大数要谨慎！

java 加法

输入第一个整数: 2000000000 输入第二个

整数: 2000000000 和为: -294967296

按格式打印浮点数

```
1 // 带有输入的格式化打印 Java 程序示例
2 import java.util.Scanner; // 需要输入
3 公共类 Addition2 {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Enter x: "); // 提示
7         float x = input.nextFloat(); // 读取第一个数字
8         System.out.print("Enter y: "); // 提示
9         float y = input.nextFloat(); // 读取第一个数字
10        float          // 将数字相加并存储结果
11        "Sum is: %f%n"
12        "Sum is: %.2f%n"
```

```
13         }// 主体部分结束
14     }// end class Addition2
```

输入 x: 1.255

输入 y: 2.75

总和为: 4.005000

总和为: 4.01

Java 中的算术运算

Java 操作	操作员	代数表达式	Java 表达式
加法	+	$f + 78$	<code>f + 78</code>
减法	-	$f - c$	<code>f - c</code>
乘法	*	bm	<code>b * m</code>
分部	/	x / y 或 $\frac{x}{y}$	<code>x / y</code>

剩余	%	$r \bmod s$	r % s
----	---	-------------	--------------

Java 中的算术先例

- Java 中的算术运算遵循标准先例
 - **乘法、除法和余数** (*, /, %) 首先求值。如果有多个此类运算符，则*从左到右*依次运算
 - **加法和减法** (+, -) 接下来进行运算。多重运算符*从左到右*进行评估。
 - **赋值** (=) 最后评估
- 为了提高可读性和避免错误，可以在复杂表达式中使用括号。

Java 等式和关系运算符

代数运算符	Java 操作员	Java 条件示例	意义
=	==	x == y	x 等于 y
≠	!=	x != y	x 不等于 y
>	>	x > y	x 大于 y
<	<	x < y	x 小于 y
≥	>=	x >= y	x 大于或等于 y

\leq

\leq

$x \leq y$

x 小于或等于 y

比较和布尔示例

```
1 // Java 程序比较示例
2
3 import java.util.Scanner; // 需要输入 public class
4 Comparison {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7         System.out.print("Enter x: "); // 提示
8         int x = input.nextInt(); // 读取第一个数字 System.out.print("Enter y:
9         "); // 提示
10        int y = input.nextInt(); // 读取第一个数字
11        boolean isLarger = x > y; // 比较 x 是否大于 y boolean
12        isLess = x < y; // 比较 x 是否小于 y System.out.printf("x is
13        larger than y: %b\n", isLarger); System.out.printf("x is
14        less than y: %b\n", isLess);
15        } // 主体部分结束
16    } // 结束类比较
```

输入 x: 5

输入 y: 10

x 大于 y: 假 x 小于 y: 真

输入 x: 5

输入 y: 5

x 大于 y: 假 x 小于 y: 假

输入 x: 10

输入 y: 5

x 大于 y: 真 x 小于 y: 假

摘要

- Java 是一种古老的语言，在许多平台上广泛使用
 - Java 字节码无需重新编译即可移植
- Java 开发周期包括五个阶段
 - 编写和编辑代码、编译为字节码、加载字节码、验证字节码并执行程序
- 版本控制对于与他人合作开发程序至关重要
 - 允许合并不同人员的更改以及滚动更改
返回软件的早期版本

- 介绍 Java 程序的基本语法和结构

有问题或意见？