《面向对象程序先导》
Lec1-JAVA程序入门与代码管理

北京航空航天大学计算机学院

吴际

2023.9.8

内容

- 课程简介
- 什么是面向对象
- 面向对象的基本特征
- JAVA程序入门介绍
- 版本控制
- Git 基础知识
- 作业内容介绍

课程简介

- “先导”什么
  - 服务于《面向对象设计与构造》课程的教学目标
  - 面向对象的核心概念
  - Java语言及其初步使用技巧
  - 基础的层次化设计思维
- 如何“先导”
  - 讲练结合，每周一个小作业
  - 使用和OO正课完全相同的训练平台和过程（略去互测）
    - oo.buaa.edu.cn, gitlab.oo.buaa.edu.cn
  - 7次作业，逐步迭代，为OO正课的第一单元训练提供稳固基础

课程简介

- 先导课源自于OO课在开课前布置的Pre作业
- 目标是从Java小白到一周内完成200行有效代码的Java程序
- 快乐体验很重要
  - 8次课，7次丝滑的迭代式作业
- 记分规则
  - 总分：作业分 + 奖励分
  - 作业分：公测分 + 代码风格分
  - 公测分：中测基础分 + 中测过程分 + 强测结果分
  - 成绩：总分映射到百分制
- 请详细阅读推送给大家的课程规则文件

什么是面向对象

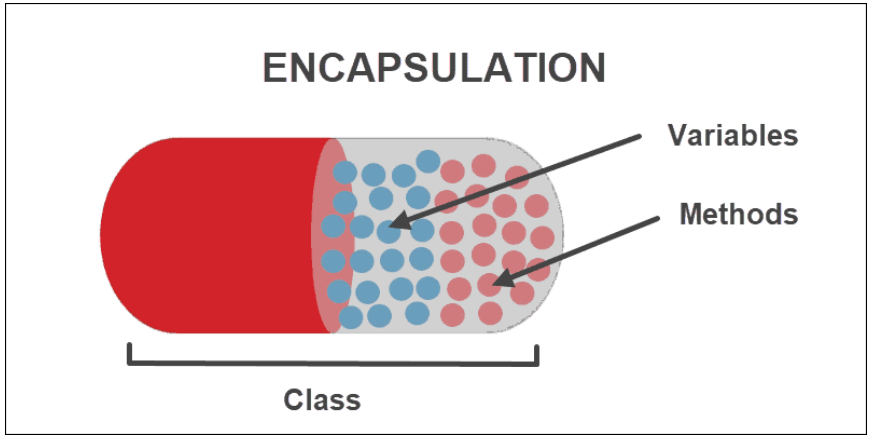
- 面向对象：以对象为中心来构建程序逻辑的方法
  - 一切皆为对象
  - 程序逻辑：数据及其关系、行为及其关系
- 面向对象语言使用“类”这个概念来抽象化“对象”，一个类可以实例化任意数目的对象
  - 类：数据与行为的综合体
  - 类关系：表征数据间关系或行为间关系
- 面向对象提供了控制复杂性的重要机制

面向对象的三个基本特征

- 封装：让内部复杂性外部不可见
- 继承：通过抽象层次来协同降低复杂性
- 多态：通过多种形态来解藕复杂性

封装

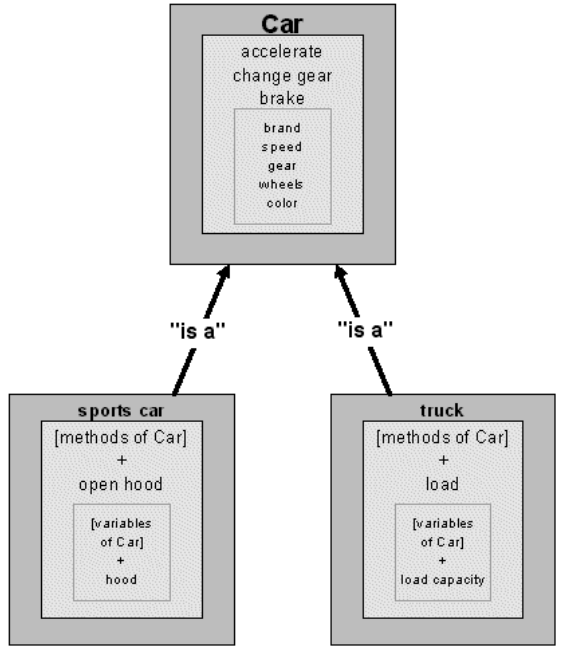
- 隐藏类实现细节的机制
  - 类封装了数据和方法，通过可见性(visibility)来限制外部对内部的数据和方法的访问
  - 类使用者无需了解类的内部细节，因而类可以进行变更而不影响使用者
  - 内部细节外部不可见，提高了类的易用性，可减少代码出错的风险




```
public class Person {
    private final String name;
    public Person(String name) {
            this.name = name;
    }
    public String getName() {
            return name;
    }
    public void print() {
            System.out.println(name);
    }
}
```

继承

- 一个类从另一个类中获得属性和方法的机制
  - 建立了上下层关系：父类与子类
- 子类可以复用父类的**设计**与**实现**
- 子类可以对父类进行扩展
  - 增加新的属性和方法
  - 重写父类实现的方法




```
public class Student extends Person {
    private final int id;
    public Student(String name, int id) {
        super(name);
        this.id = id;
    }

    public void print() {
        System.out.println(getName() + " " + id);
    }
}
```

多态

- 针对同一个指令（方法名），一个类拥有多种响应形态
  - 本质上有多种同名的方法
  - 要么是在不同的类中实现，要么拥有不同的参数
- 解藕了类针对同一个指令的多种处理逻辑，OO语言提供了自动根据对象和输入参数来匹配合适的方法
  - 分派机制
- 可有效提升代码的可扩展性

```
Person person = new Student("Amy", 21373215);
person.print();
```

Java程序的类

- 编程单位
  - C程序：函数
  - Java程序：类
- Java程序中的类由属性及方法组成
  - 属性：定义数据结构
  - 方法：定义对数据结构的操作函数
- 每个类都有一种构造方法，用于实例化对象
  - 构造方法：初始化属性变量，JVM为对象申请内存，返回对象指针
  - Book aBook = new Book(...);

从结构化编程到面向对象编程

- 在C语言编程中，我们首先要定义结构体（如栈和队列）：

```
#define DataType int
typedef struct Stack {
    DataType *_data;
    int _capacity;
    int _top;
} Stack;

typedef struct Queue {
    DataType *_data;
    int _capacity;
    int _front;
    int _rear;
} Queue;
```

从结构化编程到面向对象编程

- 需要单独为结构体定义和实现相应的操作函数

  int mypush(Stack *stack, DataType data){ … }

- 当看到Stack结构体类型时，需要自己去找操作它的函数
  - 结构体与操作函数分离，增加了复杂性
- 面向对象编程把结构体与操作函数**封装**在一起

```
public class Stack{
    DataType _data[];
    int _capacity, _top;
    int mypush(DataType e);
    DataType mypop( );
} Stack;
```

```
typedef struct Stack{
    DataType *_data;
    int _capacity, _top;
    int (*mypush)(struct Stack*, DataType);
    DataType (*mypop)(struct Stack*);
} Stack;
```

Java程序中的接口

- 在 Java 程序中，接口（interface）是与类处于同一层次的类型
  - 接口中只有方法，没有属性
  - 接口中的方法没有实现体
- 任何类都可以实现一个接口，即实现接口中所定义的所有方法
  - 一旦实现了一个接口，就可以使用该接口类型来引用相应的对象

```
public interface Showable {
    void print();
}
public class Person implements Showable {
    void print() {…}
}
```

```
public class Agent{
      …
    public void show (Showable person) {
        person.print();
    }
}
```

Java程序中关系

- 类关系
  - 继承关系
  - 关联关系
- 类与接口间的关系
  - 实现关系
  - 关联关系
- 接口间关系
  - 继承关系

```
public class Student extends Person {
    private final int id;
    public Student(String name, int id) {
        super(name);
        this.id = id;
    }

    public void print() {
        System.out.println(getName() + " " + id);
    }
}
```

```
public class WeiQi {
    private Position pieces[];
    private boolean black_turn;
    private int hands;
    boolean play (Position p, boolean black) {…}
    …
}
```

```
public class Position {
    private int x, y;
    private boolean empty;
    private boolean black;
    public put (boolean black){…}
    public clean() {…}
}
```

Java程序中的访问权限设置

- 类可以规定哪些成员能被外部访问
  - public, protected, private
  - 访问：obj.attr; obj.func(…);
- C 程序结构体成员变量可以被每个函数访问（读或写）
- Java程序则依据权限设置进行检查，如果违背则报编译错误

```
public class WeiQi {
    private Position pieces[];
    private boolean black_turn;
    private int hands;
    boolean play (Position p, boolean black) {
        if(p.x > N)…
    }
}
```

```
public class Position {
    private int x, y;
    private boolean empty;
    private boolean black;
    public put (boolean black){…}
    public clean() {…}
}
```

Java程序中的static声明

- Java程序中使用static关键字声明的方法和属性
  - 类级成员，否则为对象级成员
- 成员访问方式（都需要遵守可见性约定）
  - 类级成员通过类名：MyClass.member
  - 对象级成员通过对象名：obj.member
- 静态属性可以被类的所有对象共享访问
  - 类似于 C 语言中的 static变量
- 限制规则
  - 静态方法只能访问静态属性 vs 非静态方法能访问非静态属性和静态属性
  - 静态方法不能调用非静态方法 vs 非静态方法可以调用静态方法

Java程序中的this

- this指当前对象
- 程序中非静态方法可以使用this关键词
  - 指向当前代码运行时所处于的对象空间
- 任何一个非静态方法的执行
  - 必然通过一个对象来调用

Java程序中的Object

- Object是一个类
  - 所有你定义的类默认都是Object的子类
  - 任何时候都可以写这样的语句：Object obj = new MyClass(...)
- Object内置了很多重要的功能
  - equals方法：判断输入的Object对象与this是否相同
  - clone方法：对this克隆产生一个新对象
  - toString方法：把this对象的内容转换为String
  - 线程访问控制的方法

Java语言如何学

- Java中方法内的语句和C语言如出一辙
  - 没有C语言繁琐的指针及其运算
- 理解前面所介绍的关键概念
- 在IDE上动手写程序，修编译错误
- 在IDE上调试程序，修逻辑错误
- 学习和了解Java常用类库
  - 搜索引擎、博客、技术文章等

Java程序的错误处理

- 一般而言，程序会遇到两类错误
  - 编译错误：不符合程序语言文法要求
  - 运行时错误：不符合编译器所规定或程序自定义的运行时安全要求
- 编译错误导致程序无法成功编译，因而无法执行
- 运行时错误导致程序无法成功执行

初学者常遇到的Java编译错误

- …… Expected（缺少某些符号）
- Unclosed String Literal（字符串没闭合）
- Cannot Find Symbol（符号未定义：可能是变量名、类名、方法名等）
- Missing Return Statement（方法中存在某些路径没有return）
- … Cannot Be Referenced From a Static Context（静态作用域中引用了非静态的属性、方法）
- Constructor in class cannot be applied to given types（调用了不存在的构造方法）
- Cannot access private property（外部不能访问内的私有成员）

常见的Java程序运行时错误

- ArithmeticException（算术运算异常）
  - x = y/0;
- ClassCastException（类型转换异常）
  - Book b = (Book) new Object();
- NullPointerException（空指针异常）
  - Book b;b.borrow();
- IndexOutOfBoundsExcention（索引越界异常）
  - int[] arr = {1,2,3,4}; int b = arr[4];

如何管理程序版本

- 开辟多个目录?
  - Code20230903...?
- 如果你想恢复到几天的某个版本怎么办?
- 版本控制是指在软件开发过程中对程序代码、配置文件及说明文档等文件变更的管理
- 版本控制还能帮助我们分析版本间的差异

程序开发会存在多个版本

- 一个项目往往会存在多个不同的版本
- 每个版本都可能需要和发生改动
- 例如，对于一款游戏App可以同时存在：历史版本1.2，发布公测的 1.3 版本、正在内测的 1.4 版本、正在开发的 1.5 版本等

多人协同开发

- 一个项目往往需要多人协同开发，要控制多人对项目的修改所带来的相互影响
  - 了解做了哪些改动(diff)
  - 了解改动带来的影响（逻辑分析）
- 版本管理工具建立不同的分支
  - 分支之间互不干扰
  - 根据需要可以对多人的工作成果进行合并

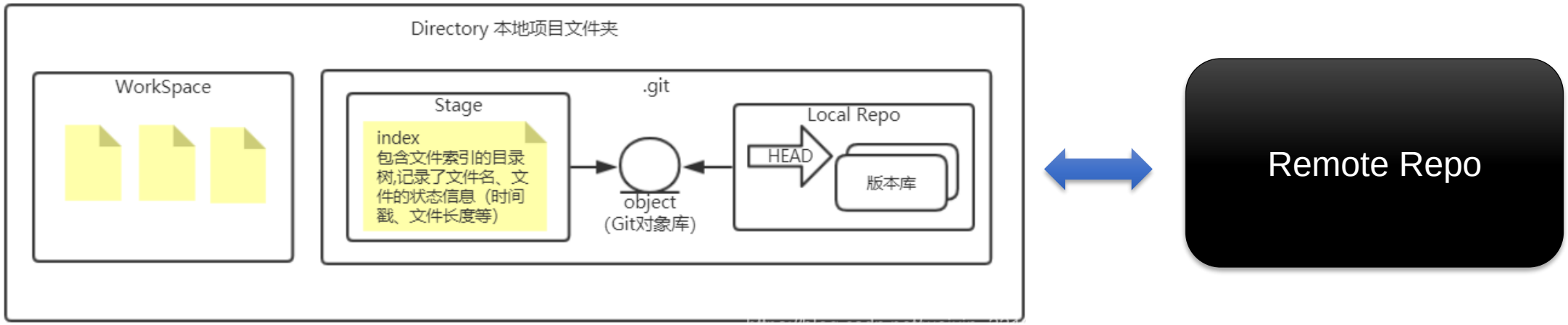
基于Git的版本控制系统

- 编程类课程的开发作业一般都要求提交代码进行评测
  - 有些课程要求将代码文件提交到评测平台，反复提交繁琐，且容易搞错
  - OO课程及先导课，项目代码提交到gitlab，评测机主动从gitlab拉取
- Gitlab系统是一种基于git协议的版本控制系统
  - 本地代码与远程代码
- Gitlab能管理一个项目的多个版本
  - 等待你去探索它更多的有趣功能

GitLab

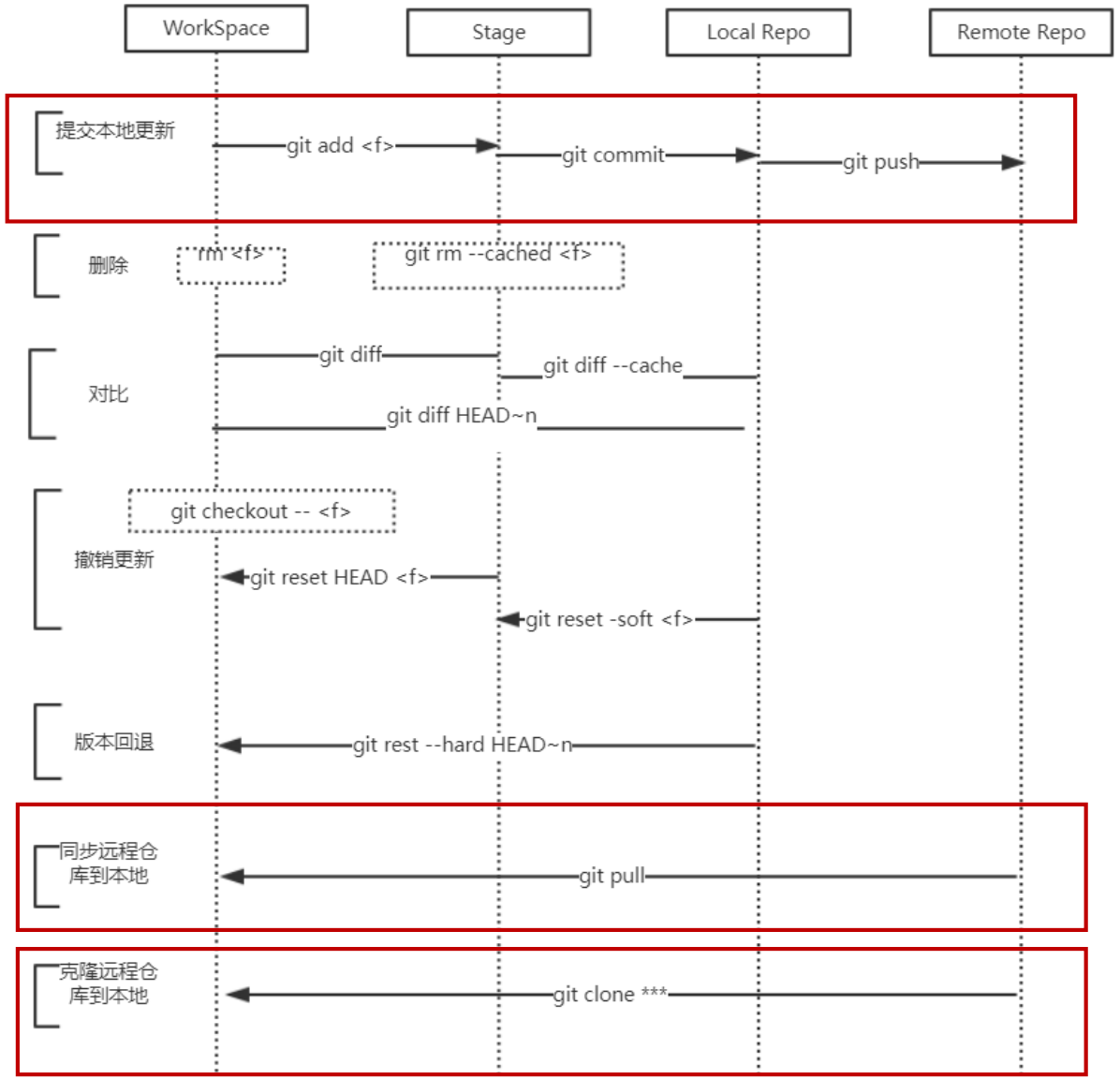
Gitlab的四个区域

- 工作区(workspace)：进行日常开发的区域
- 暂存区(stage)：运行 git add 命令后文件保存的区域，也是进行 commit 的区域
- 本地仓库(local repo)：本地版本库，记录工程的提交历史
- 远程仓库：保存在服务器上的仓库，用于团队协作



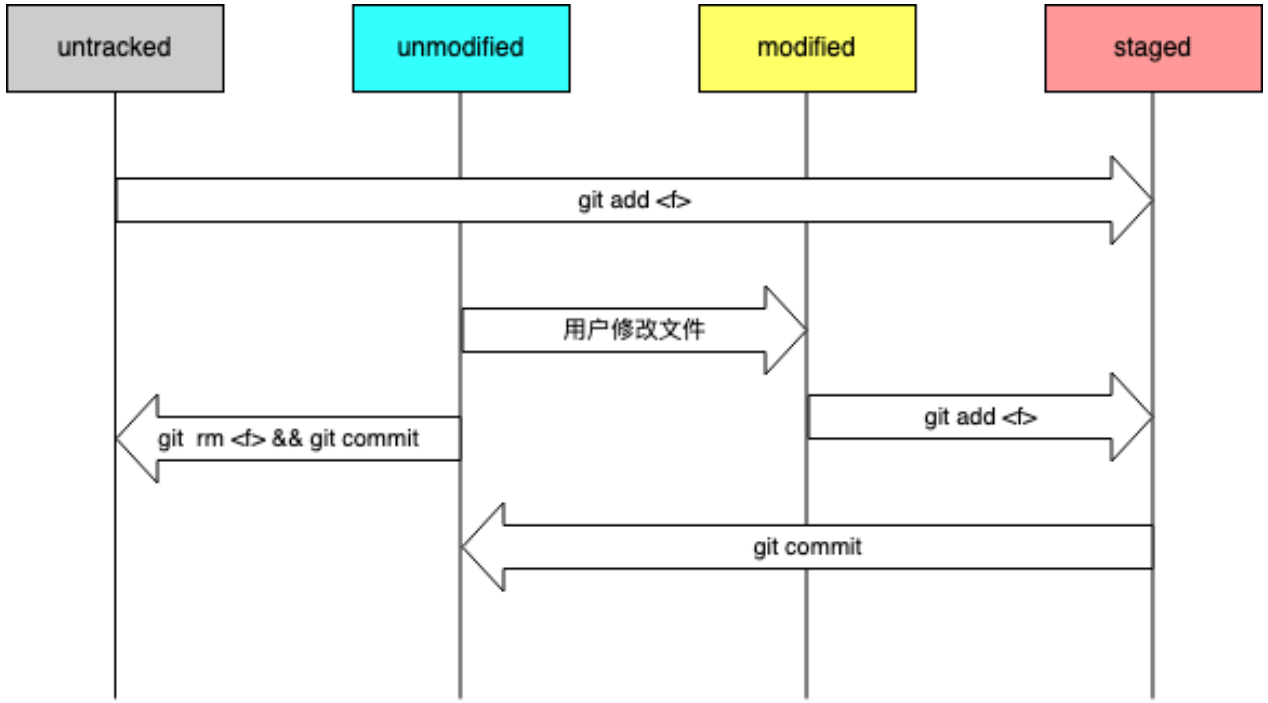
Gitlab的四个区域

- 个人电脑上的代码如何进入到服务器远程仓库?
- 如何从远程仓库代码同步到本地?
- 如何在本地克隆远程仓库代码?



Gitlab中的四种文件状态

- 未跟踪（untracked）文件：工作区中的新建文件，有待加入暂存区
- 已修改（modofied）文件：工作区中的文件被修改，有待加入暂存区
- 已暂存（staged）文件：已存到暂存区的文件，可以提交到本地仓库
- 已提交（unmodified）文件：已经通过commit提交到本地仓库的文件

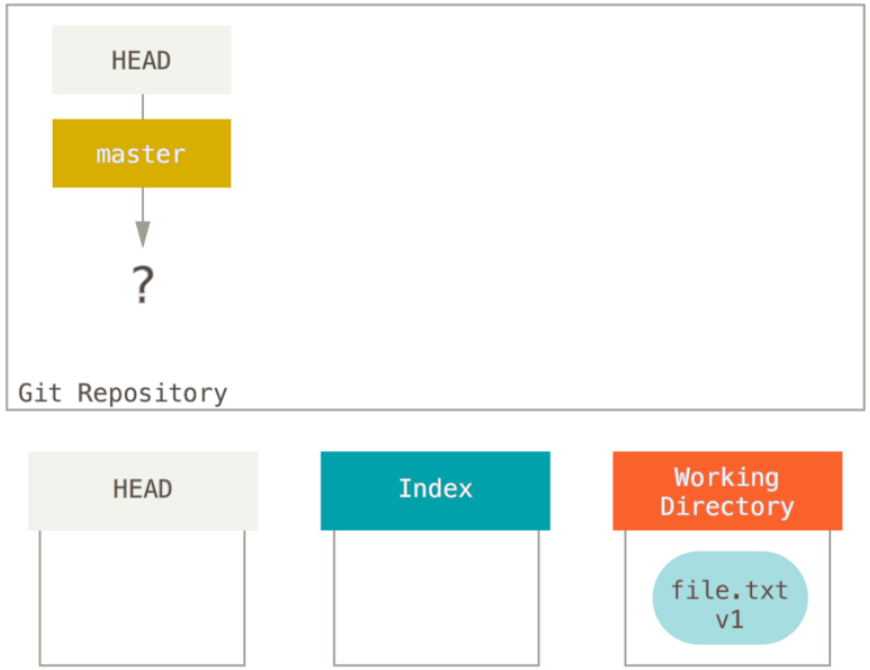


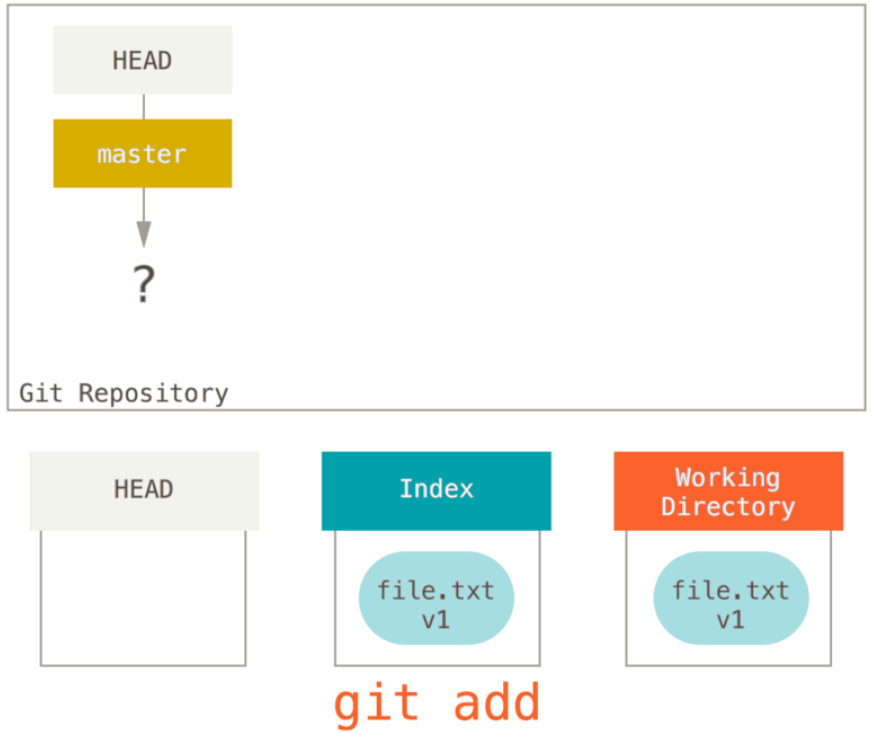
Git·基本流程·同步远程仓库

- 在提交作业代码前我们首先需要同步远程仓库，仓库已经在gitlab上被创建好
- 主要有两种方法进行同步：
  - 从远程仓库将项目clone到本地
    - 在gitlab的个人仓库中选择clone并点击复制。在命令行中使用git clone <复制的git远程仓库链接>
  - 或者在本地新建文件夹后使用git init初始化，再使用git remote add <复制的git远程仓库链接>

Git·基本流程·记录每次更新到仓库

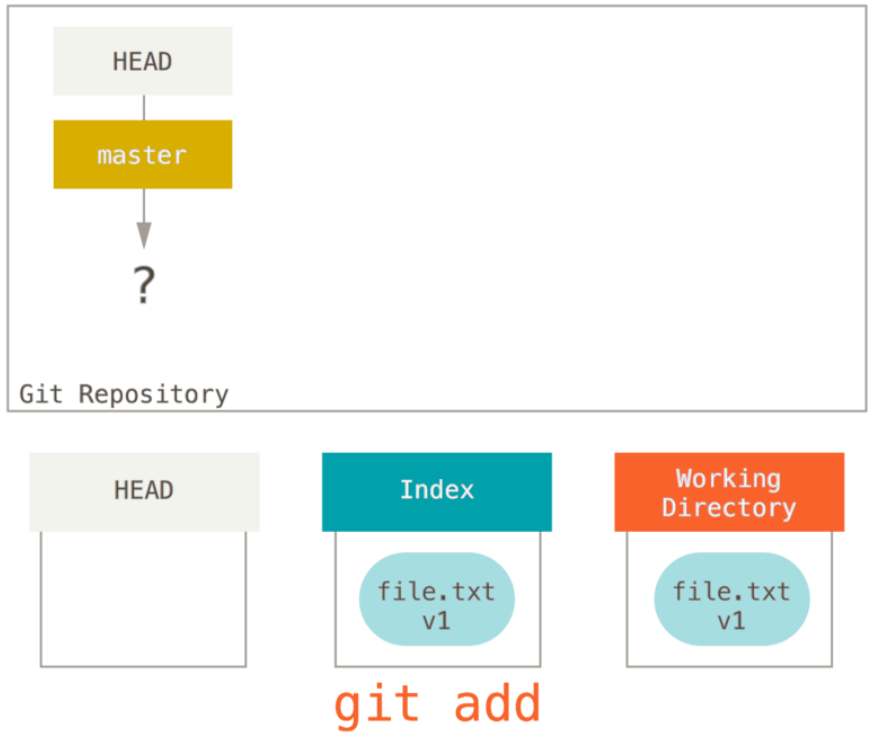
- 跟踪新文件
  - 使用命令 git add <file> 开始跟踪一个文件/文件夹
  - 例如在仓库下有一个 file.txt 文件，我们就可以用 git add file.txt 暂存该文件，此时 file.txt 处于 **staged** 状态

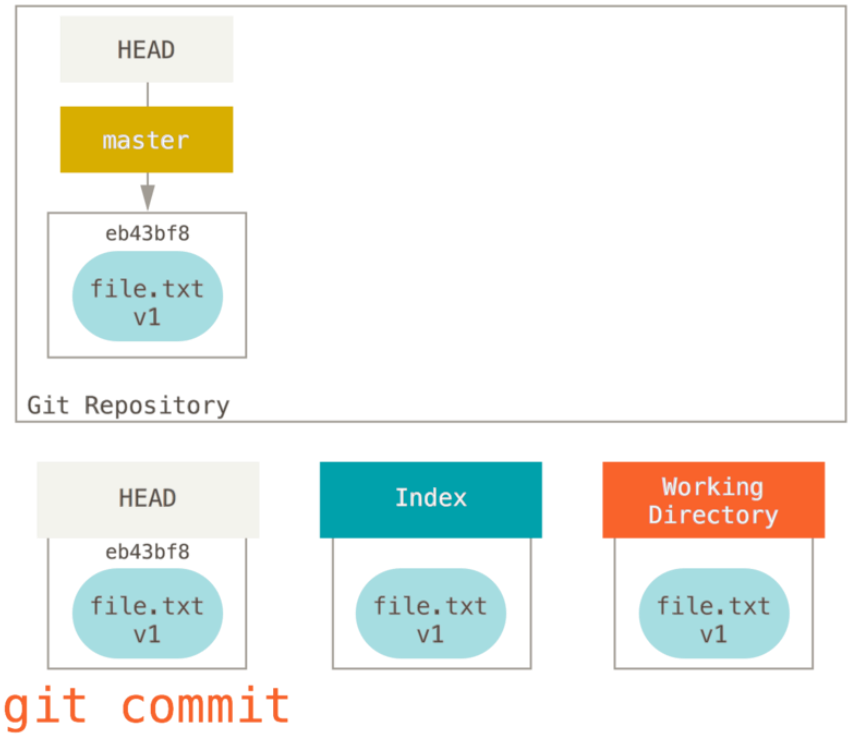




Git·基本流程·记录每次更新到仓库

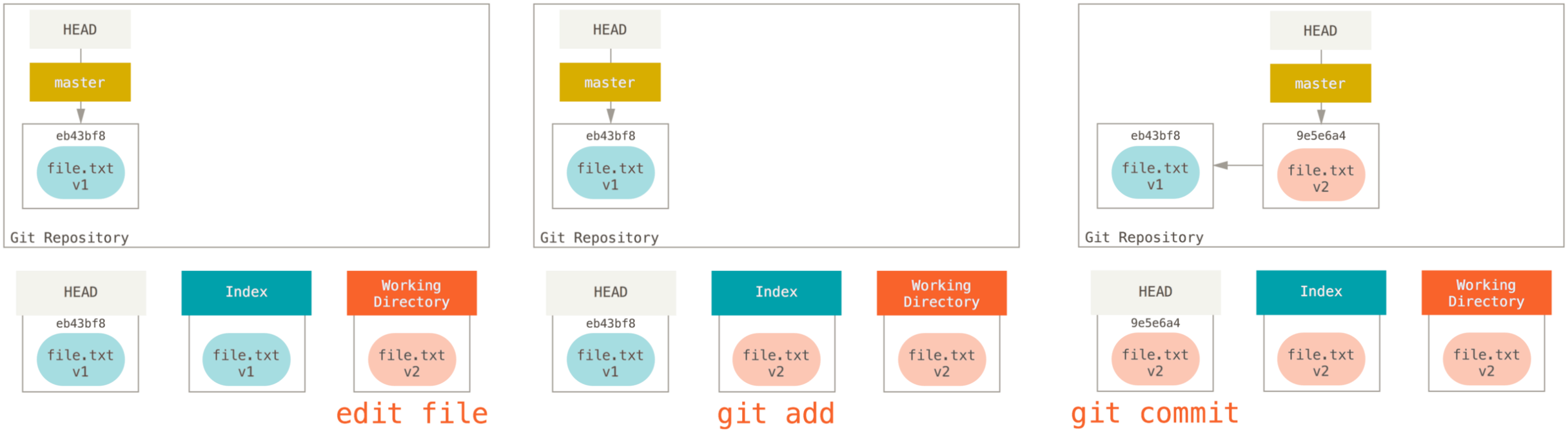
- 提交更新
  - git commit 把暂存区域文件提价到本地库。每一次提交都是对项目的一次快照，便于比较和回溯
  - 此时 file.txt 处于 **unmodified** 状态





Git·基本流程·记录每次更新到仓库

- 暂存已修改文件
  - 如果我们此时对 file.txt 进行修改，会变成 modified 状态，
  - 之后我们可以用git add指令使其暂存，git commit指令进行提交。



Git·基本流程·删除文件

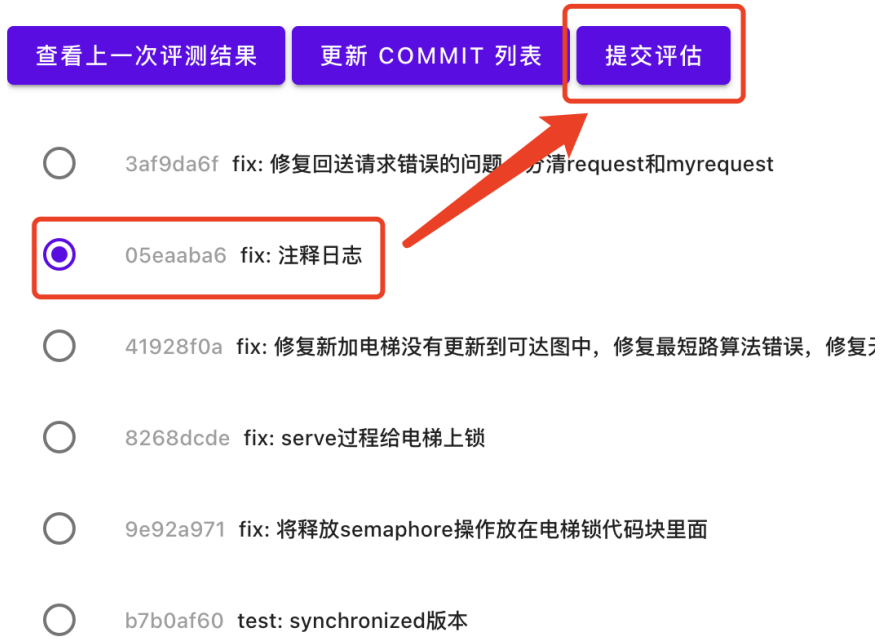
- git rm <file> 从暂存区域和工作目录中删除指定的文件。
- 如果只是简单地从工作目录中手工删除文件，运行git status会发现删除的文件在未暂存列表里
- git rm --cached <file> 可以把文件从 Git 仓库中删除（即从暂存区域移除），但保留在当前工作目录中

Git·基本流程·版本控制

- git status 可以查看文件状态
- git log 指令可以查看提交（commit）记录
- git diff 可以查看变更内容
- 版本回滚
  - git reset --hard HEAD 可以撤销工作目录中所有未提交文件的修改内容（这是一个**危险**的指令，可能会导致当前工作目录文件**丢失！**）
  - git checkout HEAD <file> 可以撤销指定未提交文件的修改内容
  - 上面两个只是版本回滚的简单应用，git提供了不同粒度的版本回滚

Git·基本流程·推送到远程仓库

- git push可以将我们本地仓库的代码推送到远程仓库
- 同步到远程仓库后，可以在课程作业网站选择要提交的代码版本
- 上述所有操作都可以通过idea图形化工具完成



作业内容介绍

- 课程网站和gitlab基础使用
  - 今天晚上会有课程网站使用的腾讯会议答疑
- 修改有语法错误的JAVA代码并提交
  - 阅读编译错误信息
  - 利用IDE工具（idea）来了解错误消息
  - 修语法错误不可以改变程序原本的功能
  - 评测机会在后台对比你提交的程序和原程序的运行结果