代码风格规范

一、Java 开发规范

严格遵循《阿里巴巴Java开发手册（嵩山版）》，关键条款包括：

类别	规范示例
命名规约	类名UpperCamelCase，方法名lowerCamelCase，常量全大写+下划线（`MAX_COUNT`）
代码格式	缩进4空格，`if/for`等后必须加空格和大括号，行宽不超过120字符
OOP规约	禁止使用`finalize()`，覆写方法需加`@Override`，静态方法调用用类名而非对象
集合处理	集合初始化指定容量，禁止在`foreach`循环内进行元素增删操作
异常处理	捕获异常需处理（不吞异常），自定义异常继承`RuntimeException`
并发处理	线程池需通过`ThreadPoolExecutor`显式创建，锁必须成对释放

二、Vue 3 开发规范（自定义）

1. 组件设计规范

规则类型	规范
组件命名	文件名和组件名统一使用 PascalCase（如 `UserProfile.vue`）
基础组件	统一前缀 `Base`（如 `BaseButton.vue`），存放于 `/src/components/base/`
业务组件	按模块划分（如 `OrderList.vue` 放在 `/src/components/order/`）
Props	使用详细类型定义，禁止使用 `Array` 或 `Object` 简写： `props: { status: { type: String, required: true } }`
模板语法	属性名用 kebab-case（如 `<modal :is-visible="value">`）

2. 代码风格规范

规则类型	规范
缩进	2空格缩进（与Vue官方风格一致）
引号	属性值、JS字符串统一使用 **单引号**
指令简写	优先使用 `@` 代替 `v-on`，`:` 代替 `v-bind`（如 `@click` 而非 `v-on:click`）
响应式数据	明确区分 `ref`（基本类型）和 `reactive`（对象）

3. 脚本部分规范

```vue
// 1. Composition API组织顺序
<script setup lang="ts">
// 按逻辑分组排序：依赖 -> 响应式数据 -> 计算属性 -> 方法 -> 生命周期
import { ref, computed } from 'vue'

// 2. 类型定义优先使用 interface
interface User {
  id: number
  name: string
}

// 3. 响应式数据命名规范
const isLoading = ref(false)           // 布尔状态加 is/has/can 前缀
const userList = ref<User[]>([])       // 列表数据加 List 后缀

// 4. 方法命名采用 camelCase
const fetchUserData = async () => {
  // 异步请求用 try/catch 包裹
  try {
    const res = await api.getUser()
    userList.value = res.data
  } catch (error) {
    console.error('Failed to fetch:', error)
  }
}
</script>

```

4. 样式规范

```vue
// 1. Scoped样式必须加scoped属性
<style scoped lang="scss">
// 2. 类名使用 BEM 命名法
.user-profile {
  &__header { /* 块元素 */ }
  &--active { /* 修饰符 */ }
}

// 3. 禁止使用 !important
.error-text {
  color: red; /* 正确 */
  // color: red !important; /* 禁止 */
}
</style>

```

5. 目录结构规范

```
src/
├── assets/           # 静态资源
├── components/       # 公共组件
│   ├── base/         # 基础组件（BaseButton等）
│   └── business/     # 业务组件（按模块划分）
├── views/            # 路由页面
│   ├── HomeView.vue
│   └── user/
│       ├── UserList.vue
├── stores/           # Pinia状态管理
│   ├── userStore.ts
├── router/           # 路由配置
├── utils/            # 工具函数
├── types/            # TypeScript类型定义
└── api/              # API接口封装
```