

# 系统设置

## Git

Git 是一种版本控制系统 (VCS)，是一种用于跟踪代码版本之间的变更并与他人共享这些变更的工具。通过版本控制系统，我们不仅可以查看旧版本代码的改动，还可以还原到旧版本，并同时管理为一个项目做出贡献的多个不同团队成员。本课程的每个学生都有自己的个人仓库 (repo)，所有代码及其历史都存储在其中。

在本课程中，我们将使用 CSE GitLab，这是 CSE 支持的 GitHub 的替代版本，您可能以前听说过。在 CSE GitLab 上备份您的 repo 不仅方便您在电脑崩溃时查找过去的工作版本，还简化了提交项目的流程，并允许您与他人协作。

▶ Windows

▶ MacOS

▶ Linux

## SSH 密钥

使用 CSE GitLab 需要生成 SSH 密钥。SSH 密钥的作用与用户名和密码相同，但它们很长，而且是随机生成的。

### 生成新的 SSH 密钥对

如果你已经有一个 SSH 密钥对，可以使用它。否则，步骤如下：

1. 在 Linux 或 macOS 上打开终端，或使用 Git Bash
2. 生成新的 SSH 密钥对：

```
ssh-keygen -t ed25519
```

接下来，系统会提示你输入 SSH 密钥对的文件路径。按键盘上的 Enter 键接受默认值。你可以选择口令，但不是必须的（尤其是在使用个人电脑的情况下）。

如果系统提示“已存在密钥”，说明你已经有了一个密钥。你可以键入 n-Enter 退出，并进入下一节。

### 为 GitLab 账户添加 SSH 密钥

复制你的 SSH 公钥（从你的主目录 .ssh/id\_ed25519.pub）。你可以从终端查看文件内容：

```
cat ~/.ssh/id_ed25519.pub
```

2. 访问 CSE GitLab SSH 密钥设置页面。

3. 将复制的公钥粘贴到“密钥”字段。如果还没有标题，给它起个标题。然后点击添加密钥。

## IntelliJ

IntelliJ 是我们将在本课程中使用的 IDE（集成开发环境）。如果你以前使用过 jGRASP，那么 IntelliJ 就是其中一个很大的进步。

### 1) Git

#### 2) SSH 密钥

2.1) 生成新的 SSH 密钥对

2.2)

将 SSH 密钥添加到  
GitLab 账户

#### 3) IntelliJ

3.1) 下载 IntelliJ

3.2) 获取版本库

3.3) 安装 Java JDK 17

3.4) 配置 Checkstyle

3.5) 配置泛型检查

IntelliJ 就是其在现实世界中的生产就绪版本。

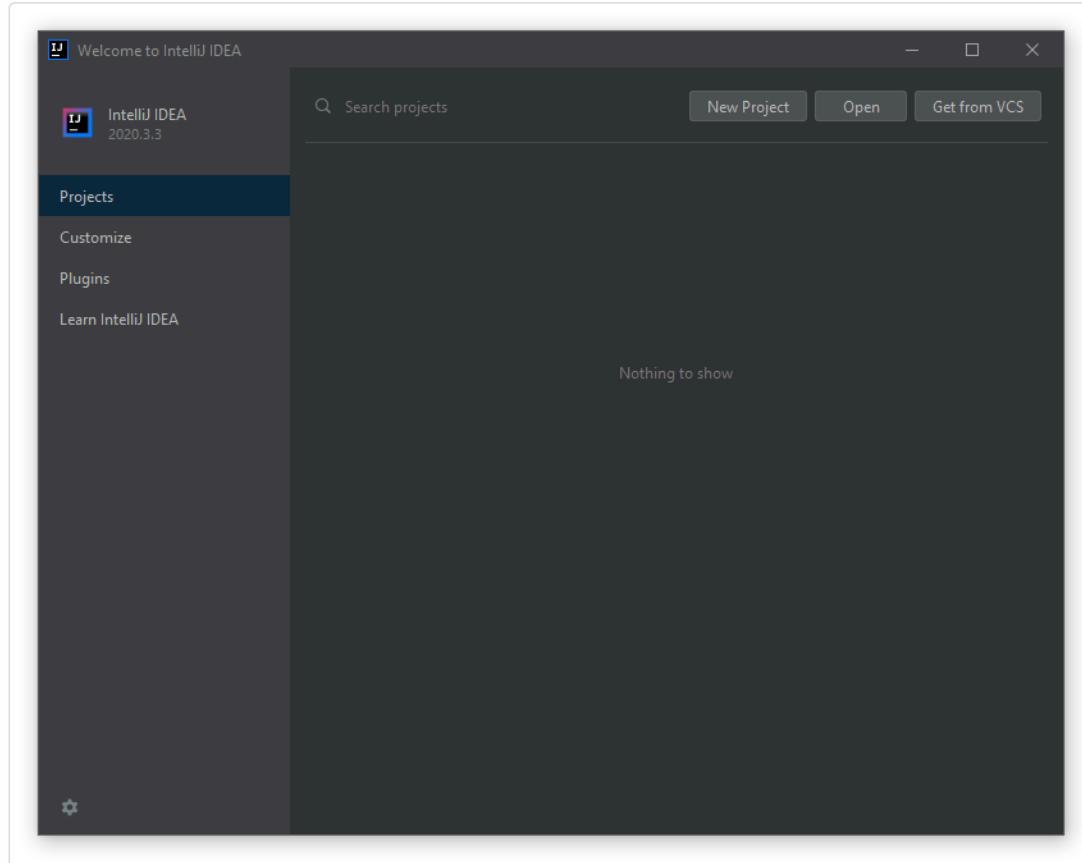
## 下载 IntelliJ

根据计算机操作系统下载并安装 IntelliJ Community Edition。如果您使用的是基于 Apple M1/M2 的计算机（2021 年后销售的大多数 Apple 计算机），请务必下载 Apple Silicon

### 1. 版本。安装时，默认的

安装好 IntelliJ 后，运行 IntelliJ IDEA，选择“跳过剩余部分”（Skip Remaining）和“设置

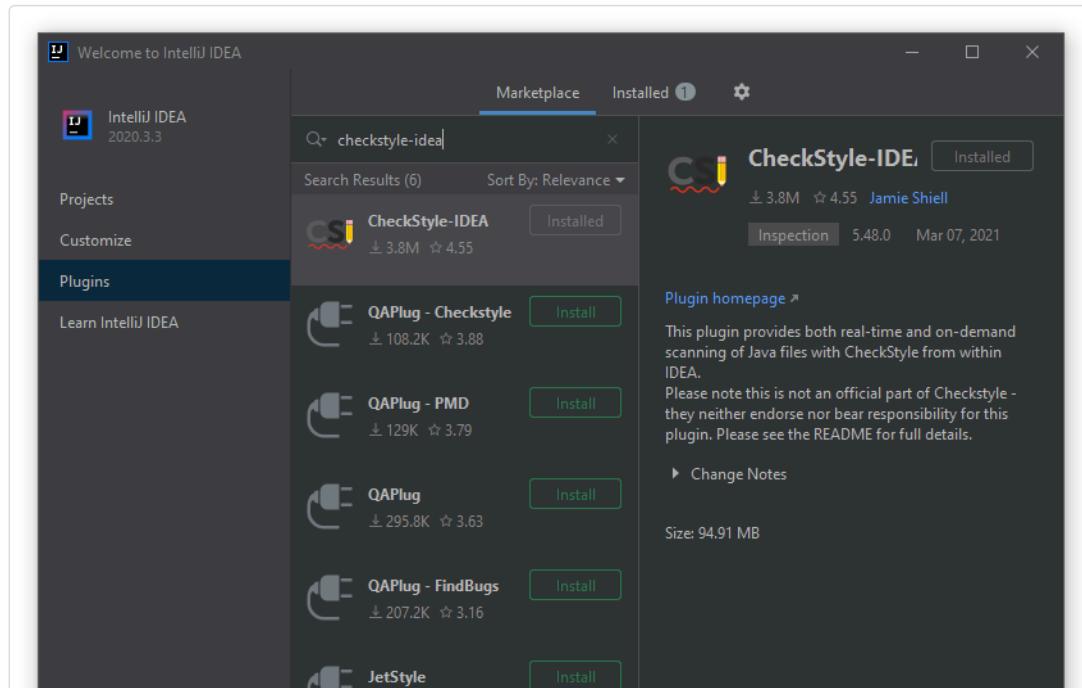
### 2. 默认值”（Set Defaults）。你应该会看到欢迎使用 IntelliJ IDEA 的界面。

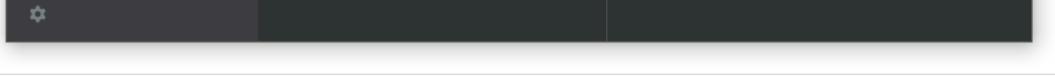


在同一个欢迎页面中，点击左侧导航窗格中的插件项目，然后点击市场选项卡。在这里，

### 3. 我们将安装两个插件：

- Checkstyle-IDEA：这是一个能发现样式错误和常见 bug 的插件。
- jGRASP：数据结构可视化工具，由 IDE 的制作团队开发。





4. 安装插件后，可能会要求您重新启动 IntelliJ。请听从它的建议。

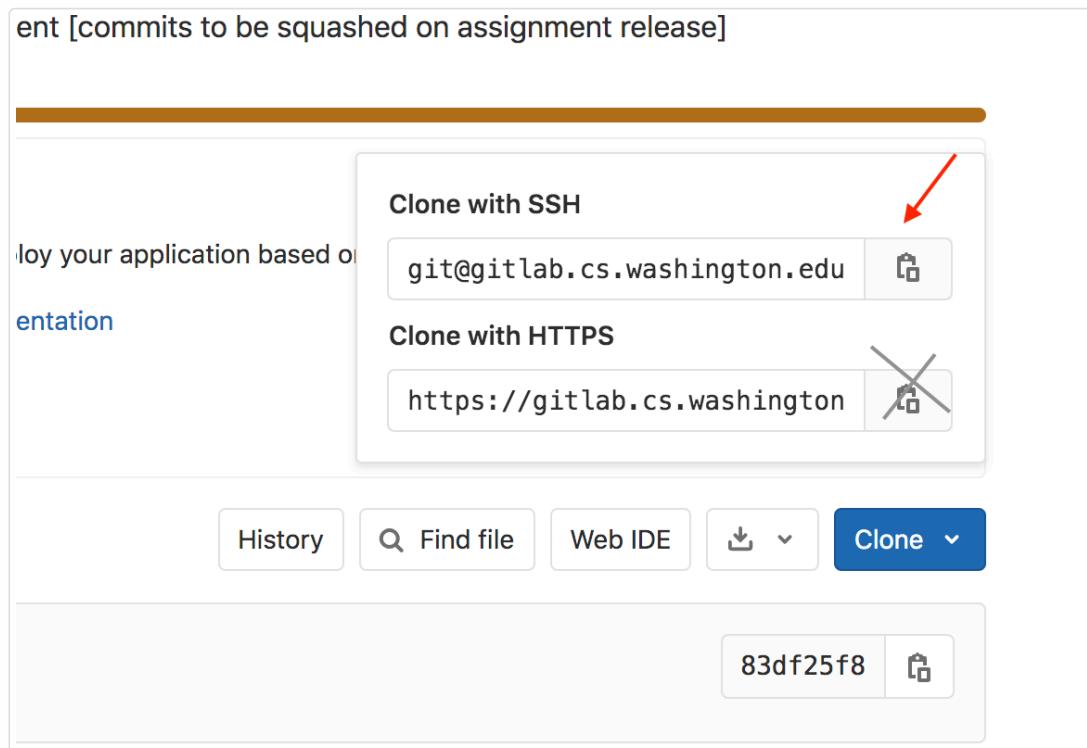
## 获取存储库

现在，IntelliJ 的初始设置已经完成，我们可以通过克隆您个人的 CSE GitLab 代码库来下载代码副本。

访问 GitLab，你会看到一个以你的 UWNetID 命名的仓库。您将在本课程中使用该仓库，

1. 里面应该已经有一些文件和文件夹，包括第一个项目 cse143review 的骨架代码。
2. 单击蓝色的克隆按钮，复制克隆与 SSH Git 地址。地址应如下所示

```
git@gitlab.cs.washington.edu:cse373-23su-students/YOURUWNETID.git
```



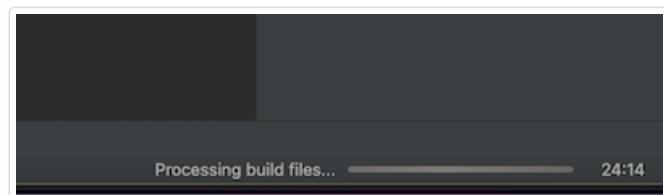
在 IntelliJ 欢迎窗口中，点击“从 VCS 获取”。粘贴之前的 Git 地址，并选择一个目录来存

3. 储你的个人仓库。
- 一旦 IntelliJ 完成了仓库克隆，一个新窗口会询问是否“信任并打开 Gradle 项目”。点击“信

4. 任项目”继续。

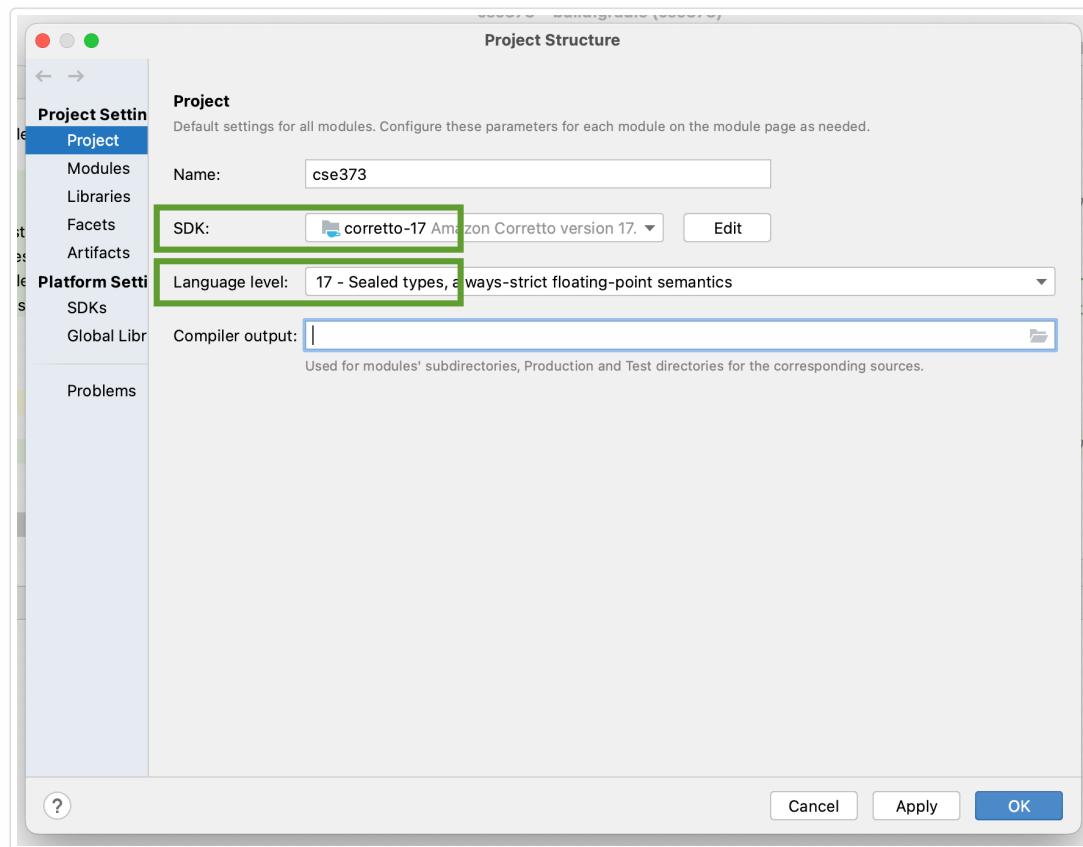
随后，IDE 主窗口将打开，IntelliJ 将开始导入 Gradle 项目；这个过程可能需要几分钟才能完成，在导入之前，IDE 的所有功能都无法正常工作。您可能需要等待屏幕右下角的进度条

5. (非常细微) 完成后才能进入下一步：



我们的项目使用 Java JDK 17；以后版本的 Java 也可以使用，但课程工作人员可能无法为与 JDK 17 以外版本相关的问题提供支持。

首先，从顶部菜单栏打开“文件”|“项目结构”，检查 IntelliJ 项目使用的 Java 版本。



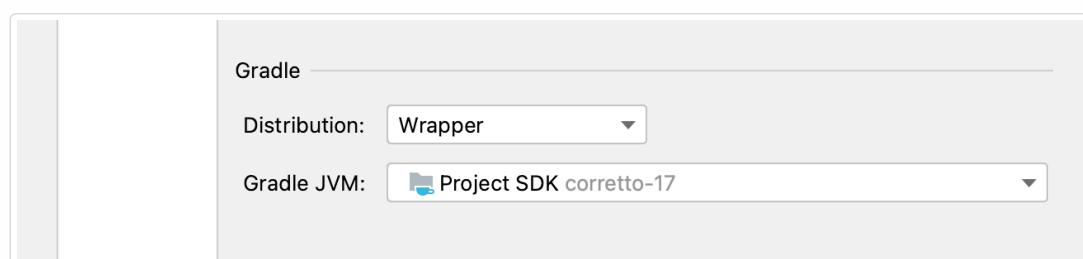
如果 IntelliJ 检测到现有的 Java SDK，它将列在检测到的 SDK 下。

- 如果有版本为 17 或更高的 SDK，请选择它。
- 如果没有 SDK 或列出的 SDK 版本低于 17，请选择添加 SDK | 下载 JDK，然后从任何供应商处选择最新版本。我们喜欢 Eclipse Temurin (AdoptOpenJDK HotSpot)，但对于本课程来说，没有具体的区别。

如果您使用的是基于 Apple M1/M2 的计算机（2021 年后销售的大多数 Apple 计算机），请注意选择标有 aarch64 架构的 SDK--这会对性能产生重大影响。

还要确保将项目语言级别设置为 SDK 默认设置。

最后，关闭该窗口并打开“设置”对话框（在 macOS 中为 IntelliJ IDEA | 首选项，否则为文件 | 设置）。导航至“构建、执行、部署”|“构建工具”|“Gradle”，并确保将 Wrapper 和 Gradle JVM 选项分别设置为“分发”和“项目 SDK”：



## 配置 Checkstyle

在本课程中，没有手动评分的样式点。相反，只要你修复了所有 Checkstyle 警告，就能获得代码风格的满分。我们需要用 CSE 373 的特定设置来设置 Checkstyle 插件。

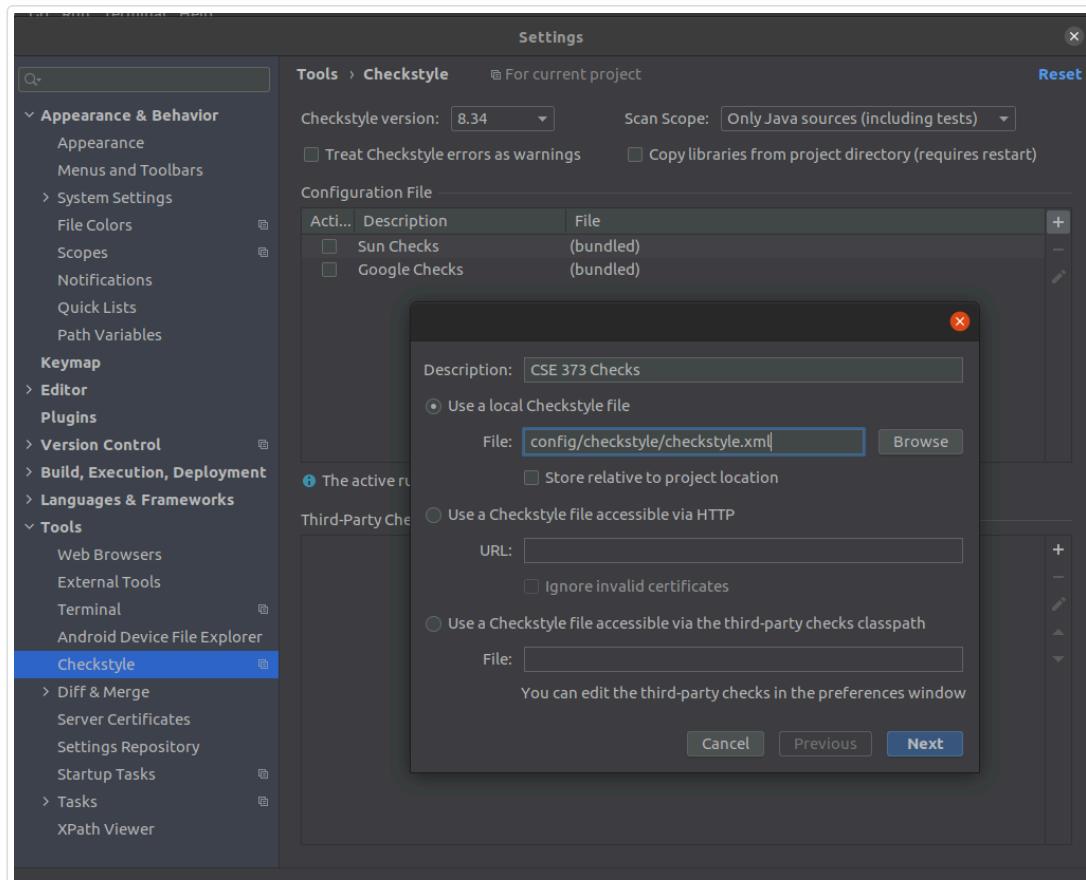
1. 打开设置对话框（在 macOS 上为 IntelliJ IDEA | 偏好设置，否则为文件 | 设置）。

2. 导航至工具 | Checkstyle。
3. 确保选择的 Checkstyle 版本为 8.36.2 或更高。
4. 将右上角的“扫描范围”下拉菜单更改为“仅 Java 源（包括测试）”。
5. 添加 CSE 373 Checkstyle 配置。



(如果计划移动或重命名项目目录, 请启用“相对于项目位置存储”; 否则启用与否并不重要)。

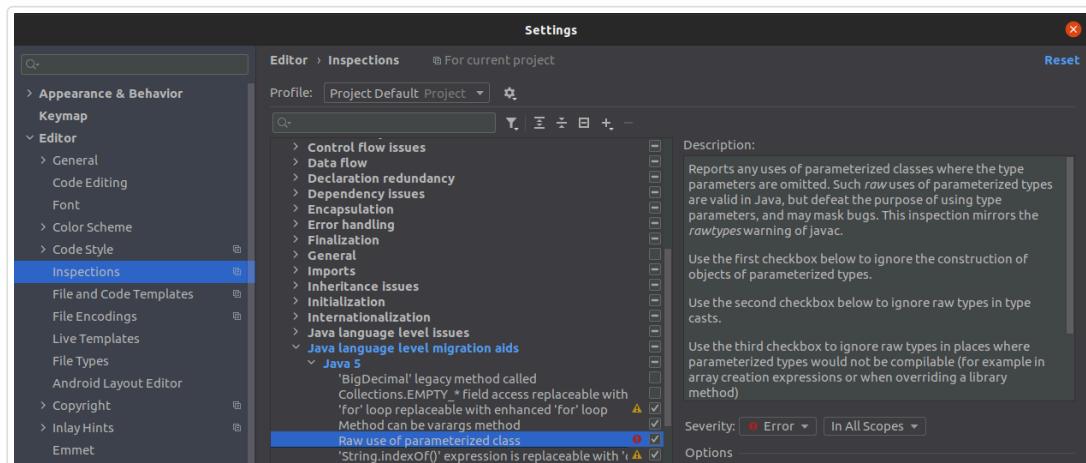
- 在“检查样式”设置中, 选中“CSE 373 检查”(或任何你选择的名称)旁边的复选框以启用它。这也是点击“应用”的好时机。
6. 用它。这也是点击“应用”的好时机。

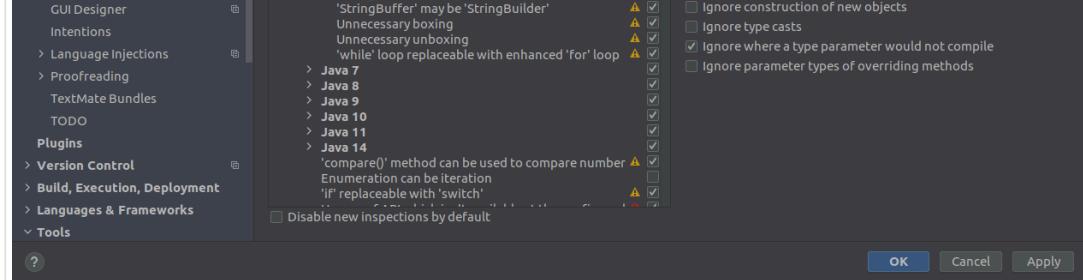


## 配置泛型检查

最后, 我们将配置 IntelliJ 以防止出现常见的泛型相关问题。在“设置”窗口仍然打开的情况下

1. 导航至编辑器 | 检查。
2. 打开 Java | Java 语言级别迁移辅助工具 | Java 5 部分。
3. 启用 Raw 使用参数化类。如果已经启用, 请单击其行以显示更多选项。
4. 将“严重性”改为“错误”, 并更改下面的其他选项, 使其与此图一致:





现在，可以单击“确定”保存并应用设置。