

# Lab 1. CheckStyle & Git

Authors: Yida Tao, Yao Zhao

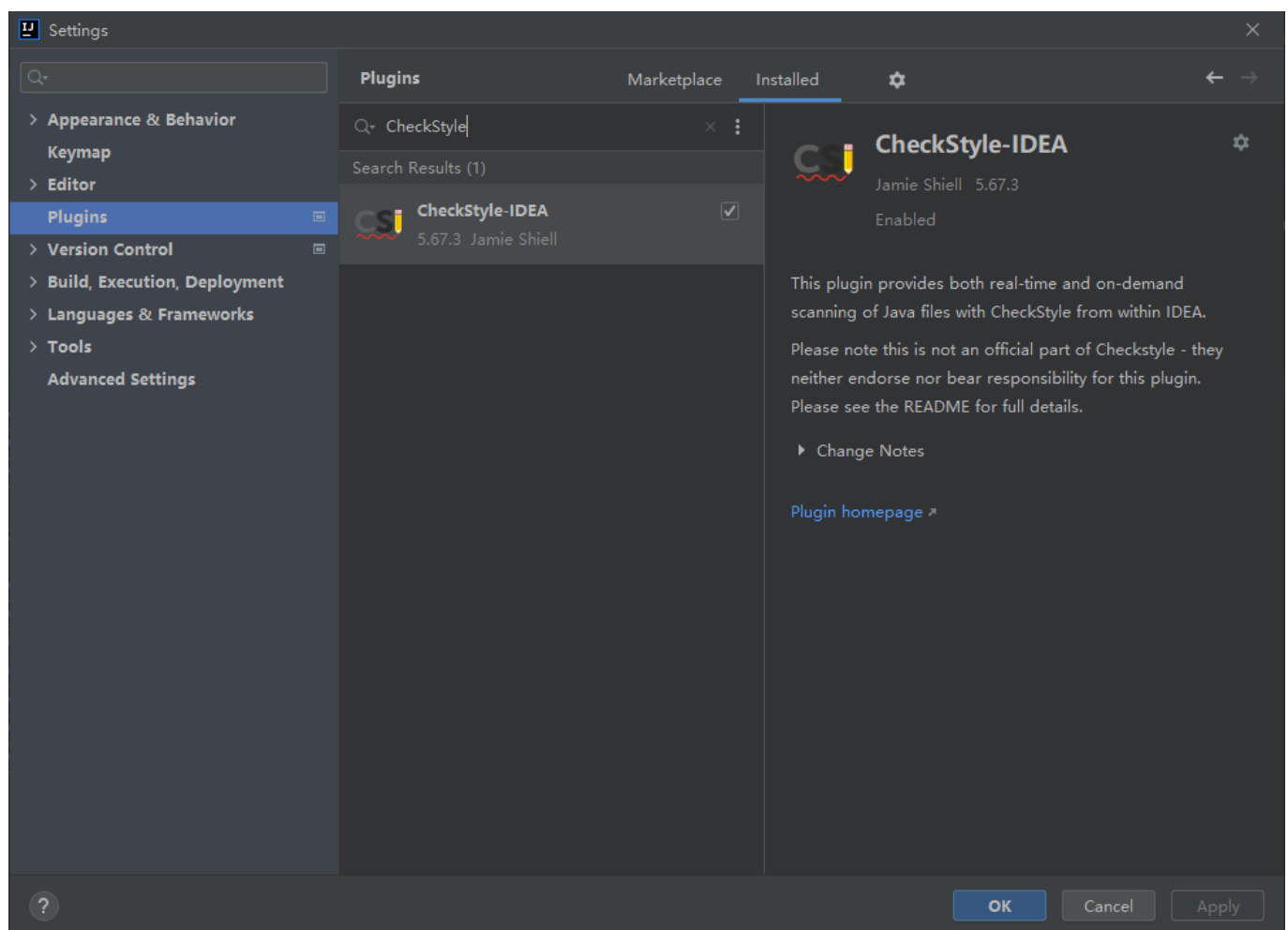
## CheckStyle

Good coding styles increase code readability and reduce software maintenance efforts. Big Tech companies also enforce strict coding styles. In this lab, we'll learn how to install and use the CheckStyle tool to improve our coding styles.

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It currently supports the *Sun Code Conventions* and the *Google Java Style*. You may also configure it to have a customized coding standard. For detailed description of this tool, please refer to its [documentation](#).

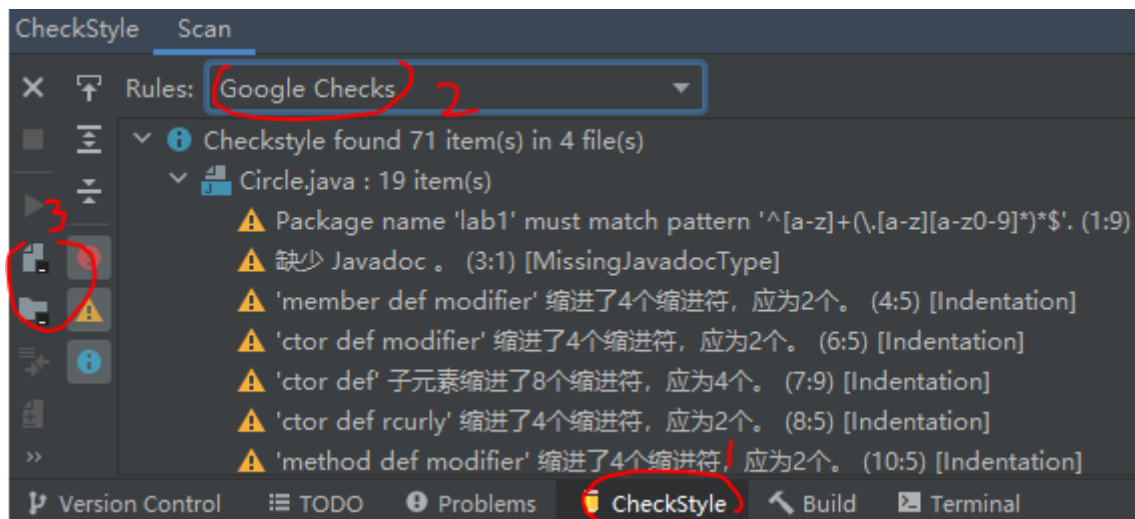
### 1. Installing the CheckStyle plugin for IntelliJ IDEA

In IntelliJ IDEA, click **File->Settings->Plugins**, search "CheckStyle", and install it.



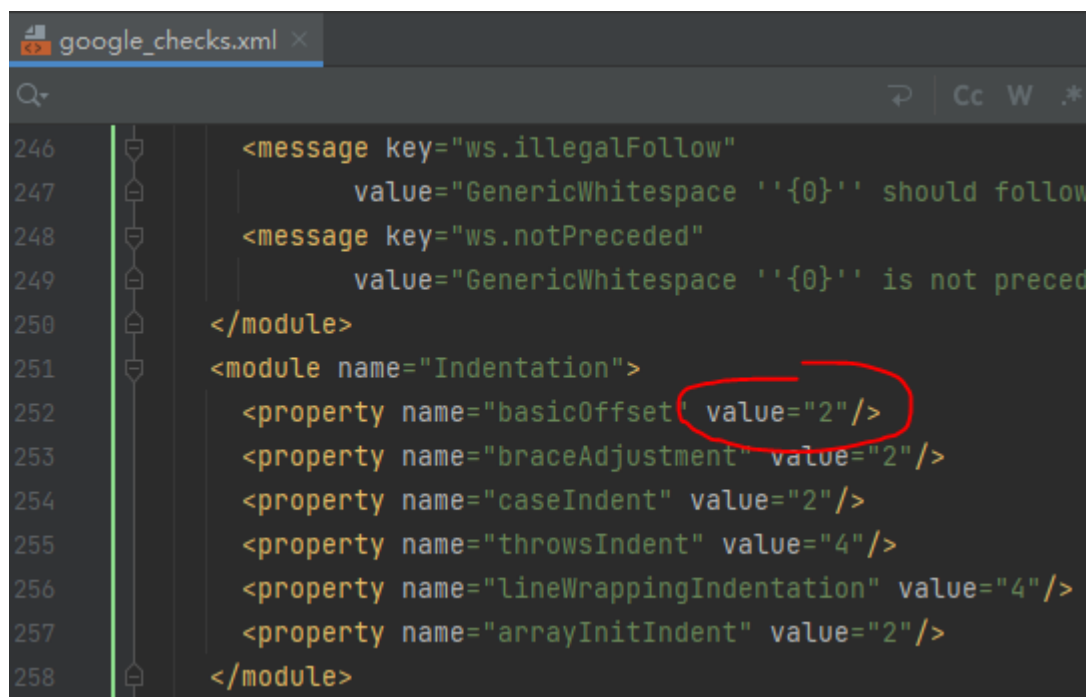
Upon successful installation, you'll have a "CheckStyle" tab in the console pane. First, select a rule (Sun Checks or Google Checks). Then, for the project that is currently active, you may ask CheckStyle to "Check file" (click the green triangle button), "Check Module", or "Check Project" (the bottom-left icon).

At this point, you'll be able to view how your code perform w.r.t. the selected coding standard.



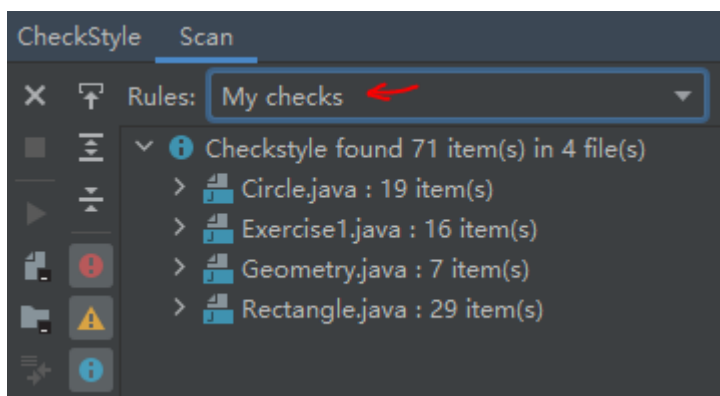
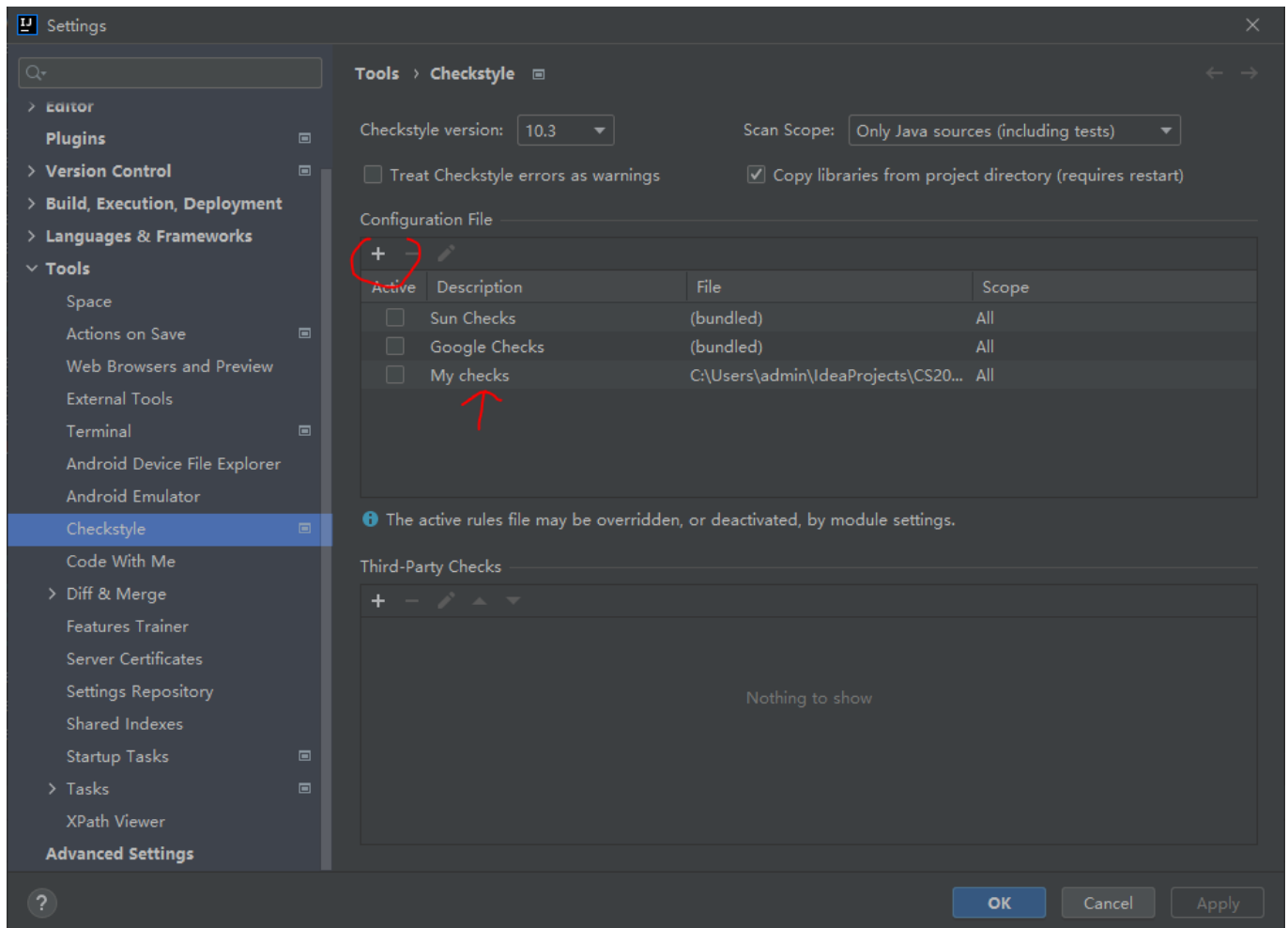
## 2. Customize CheckStyle configuration

In addition to [Sun Checks](#) and [Google Checks](#), you may also customize the coding style checks. For instance, let's download [google\\_checks.xml](#) and rename it to `my_checks.xml`. Find the element "Indentation" and change the value of its `basicOffset` property from 2 to 4.



We could then add our customized configuration file to the CheckStyle IntelliJ plugin. In **File->Settings->Tools->CheckStyle->Configuration File**, click **+** and add `my_checks.xml`. Give it a description (e.g., My checks), then **Next->Apply**. At this point, you'll find that a new rule is added and available for selection.

**Note, please use CheckStyle plugin version >10.3. Older version of the plugin may yield errors.**



### 3. Using CheckStyle in command line

We could also run CheckStyle via the [command line](#).

First, download [checkstyle-10.7-all.jar](#) (We also uploaded it to Sakai). An example run is:

```
java -jar checkstyle-10.7-all.jar -c /sun_checks.xml MyClass.java
java -jar checkstyle-10.7-all.jar -c /google_checks.xml MyClass.java
```

To export the result in .xml format, use `-o` to specify the output file and `-f` to specify the output format:

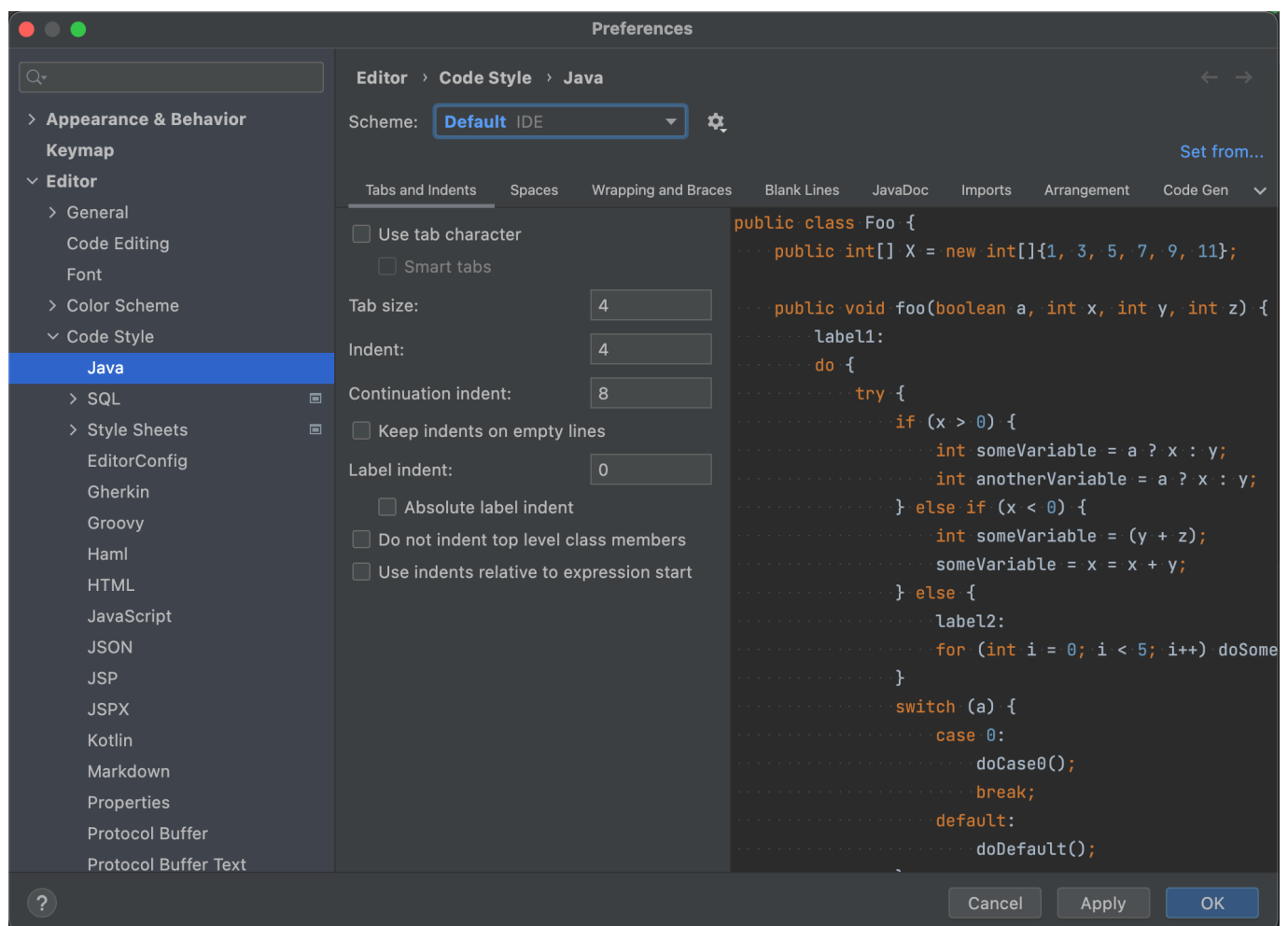
```
java -jar checkstyle-10.7-all.jar -c /sun_checks.xml MyClass.java -o result.xml -f xml
```

Executing the following command with `my_checks.xml`, you'll also see that the indentation warning produced by `Google checks` no longer appears.

```
java -jar checkstyle-10.7-all.jar -c my_checks.xml MyClass.java
```

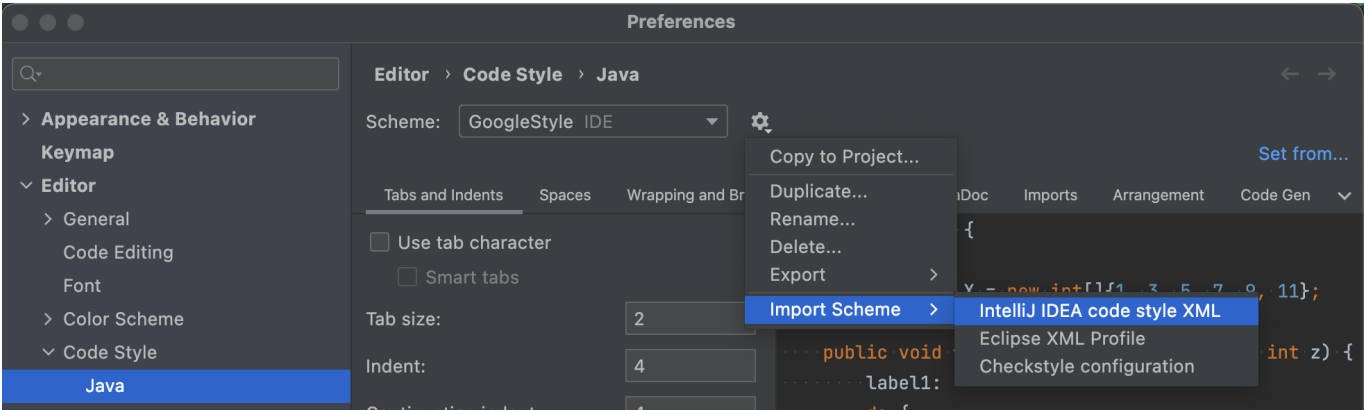
## 4. Customize IntelliJ IDEA Code Style configuration

In IntelliJ IDEA, you can click **File->Settings->Editor->Code Style->Java** to configure code styles.



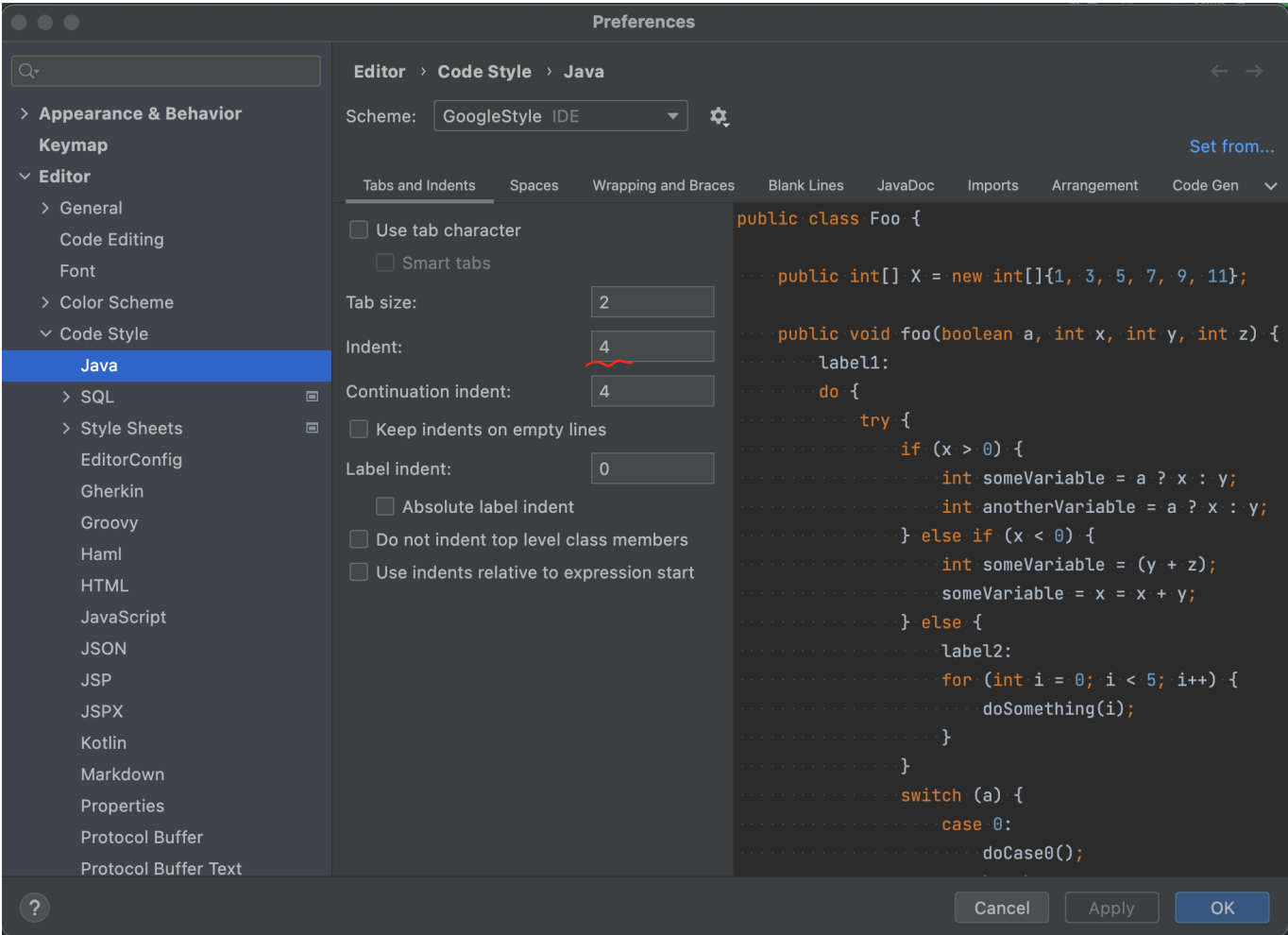
Select the code style scheme that you want to configure. The configuration could work for entire IDE or individual projects.

Click [Google Java Code Style](#) to get a IntelliJ IDEA Google Java Code Style.



Browse through the tabs and configure code style preferences.

For example, change the Indent from 2 to 4:



For a detailed description on each setting for Java, please refer to [Code Style Java](#).

Use the right-hand section of the dialog to preview the changes. When you change a setting, one or several blinking areas appear in the preview area emphasizing the changes.

Click **OK** to save the configuration.

Then you can reformat your code according the changed scheme.

## Git Setup

Git is a version control tool to manage your development process and facilitate collaboration.

Illustration of [git branch](#):

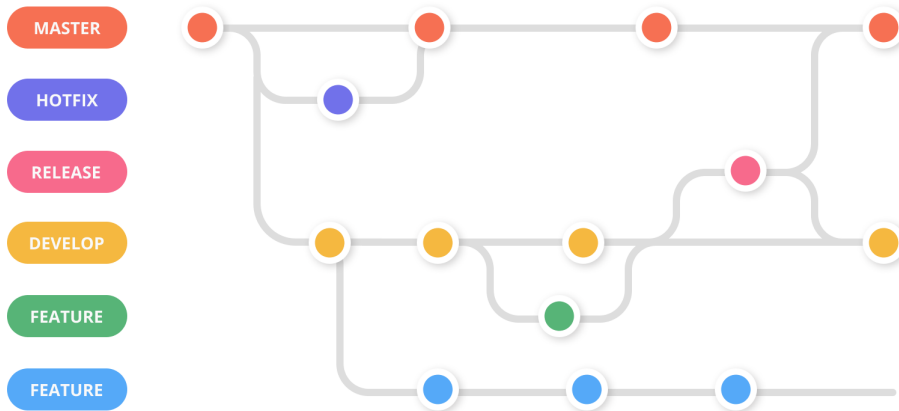
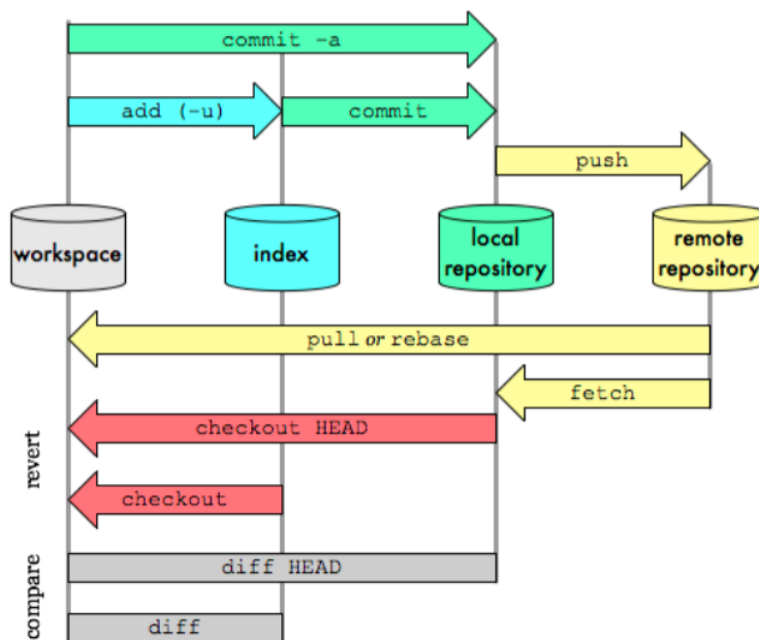


Illustration of [basic git workflow](#)



## Setup Remote Repo

We'll use GitHub as our remote repo host. Please sign up GitHub if you haven't had an account and create a new repository on GitHub follow the instructions [here](#).

Upon successful creation of a new repo, you'll get a URL like <https://github.com/alice/HelloWorld.git> indicating the repo.

## Install Git

To setup local git repository, we first need to install [git](#) on your laptop. Follow the instructions [here](#). Upon successful installation, you may use [Git Bash](#) to run any git command.

# Setup Local Repo

## Approach 1

You could make commits directly on GitHub ([see how](#)). and you could commit directly to the `main` branch. Then, run `git clone` on your laptop to clone the entire remote repo to your local workspace.

```
git clone https://github.com/alice/HelloWorld.git
```

Here, you need to **replace "alice" to your account name and "HelloWorld" to the repo name you chose when creating the repo.**

## Approach 2

**Note: If you choose this approach, please don't make any commit on GitHub yet and leave it as is. Otherwise, your local repo and remote repo will have conflicts.**

Using Git Bash, **navigate to your project folder** and execute the following commands.

Initialize the local repo

```
git init
```

Add files that you want to track

```
git add Demo.java
```

Think: which files shouldn't be tracked? Put them into `.gitignore`

Commit your change

```
git commit -m "first commit"
```

Change the branch name

```
git branch -M main
```

Link the local repo to the remote repo

```
git remote add origin https://github.com/[your-account]/[your-repo].git
```

Push the commits to the remote repo

```
git push -u origin main
```

Now, you should be able to see your first commit on GitHub repo.

### Approach 3

You may also use git directly within IntelliJ IDEA. See its [official documentation](#) for more details.