# Lab 1. CheckStyle & Git

Author: Yida Tao

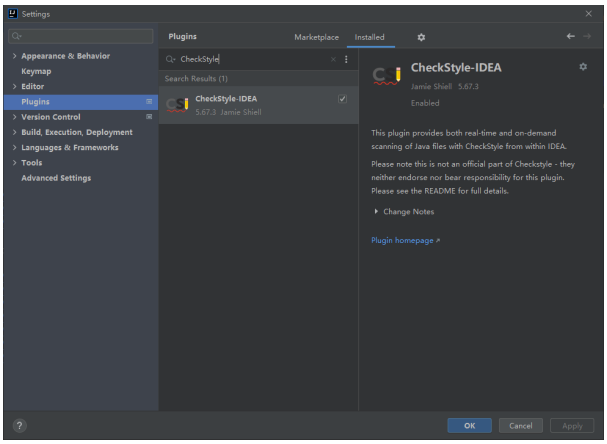
## CheckStyle

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.

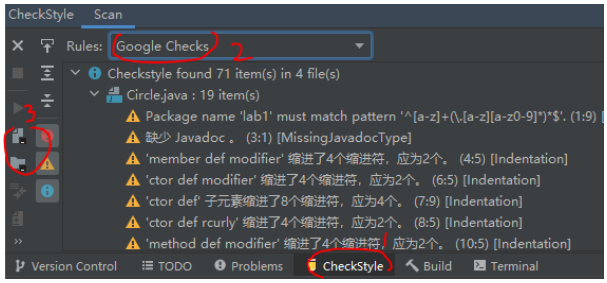
### 1. Installing the CheckStyle plugin for IntelliJ IDEA

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. Exporting the CheckStyle results

Unfortunately, the CheckStyle IntelliJ plugin doesn't support result export. Yet, we could directly run it via the command line.

First, download checkstyle-10.3-all.jar (We also uploaded it to Sakai). An example run is:

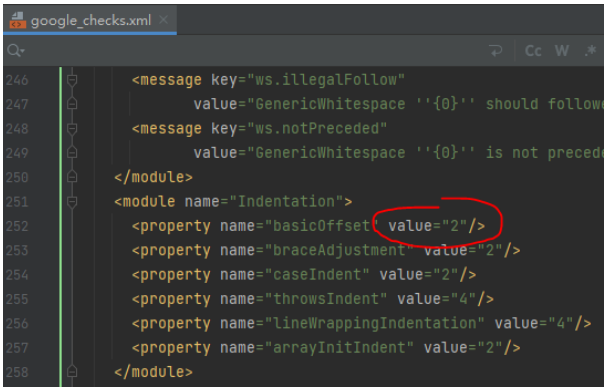
```
java -jar checkstyle-10.3-all.jar -c /sun_checks.xml MyClass.java
java -jar checkstyle-10.3-all.jar -c /google_checks.xml MyClass.java
```

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

```
java -jar checkstyle-10.3-all.jar -c /sun_checks.xml MyClass.java -o result.xml -f
xml
```

### 3. Customize CheckStyle configuration

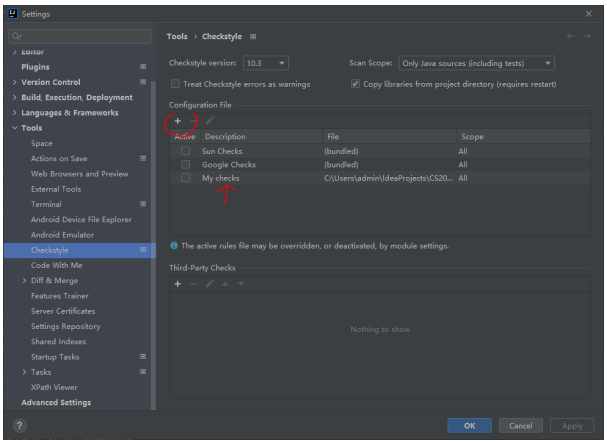
In addition to `Sun Checks` and `Google Checks`, you may also customize the coding style checks. For instance, let's download google_checks.xml and rename it to `my_checks.xml`. Find the element "Indentation" and change the value of its `basicOffset` property from `2` to `4`.

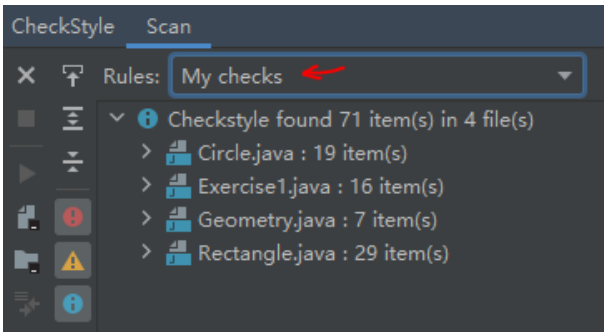



Now, executing the following command, and you'll see that the indentation warning produced by `Google checks` no longer appears.

```
java -jar checkstyle-10.3-all.jar -c my_checks.xml MyClass.java
```

We could also 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.




# Git Setup

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

Illustration of git branch:

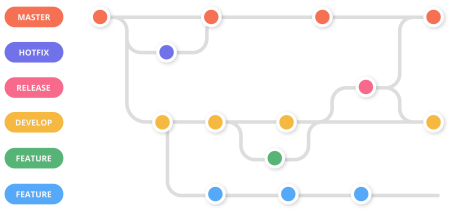
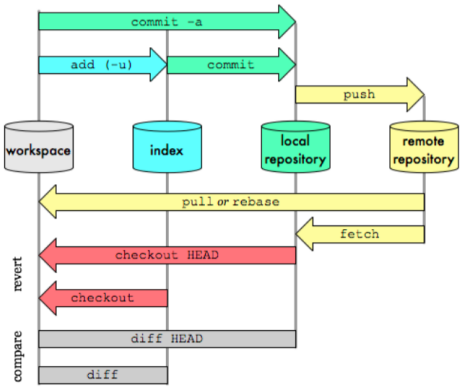



Illustration of basic git workflow




# 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.

## 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

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/alice/HelloWorld.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.