Notes

Scala 1

Packages 1

Emacs 2

Highlighting 2

Macros 2

Undo 2

Git 2

Kafka 5

# Scala

## Packages

From <http://www.ibm.com/developerworks/library/j-scala07298/>

“a combination of the Java language's declaration approach and C#'s scoped approach”

“you can use import anywhere inside the client Scala file, not just at the top of the file and correspondingly, will have scoped relevance”

“use import to bring not just nested types into lexical scope, but any member”

“import can take multiple, comma-separated targets”

“Scala:

* Does away with package-level qualification (in a way)
* Uses "public" by default
* Specifies "private" to mean "accessible only to this scope"

By contrast, "protected" is definitely different from its counterpart in Java code; where a Java protected member is accessible to both subclasses and the package in which the member is defined, Scala chooses to grant access only to subclasses. This means that Scala's version of protected is more restrictive (although arguably more intuitively so) than the Java version.”

“access modifiers in Scala can be "qualified" with a package name, indicating a level of access *up to* which the member may be accessed”

“the *object-private* specification, illustrated by private[this], which stipulates that the member in question can only be seen by members called on that same object, not from different objects, even if they are of the same type”

# Emacs

## Highlighting

<http://stackoverflow.com/questions/18090378/turn-on-background-color-when-highlighting-with-c-spc-on-a-mac>

M-x transient-mark-mode to toggle the highlighting

## Macros

c-x, ( to start

c-x, ) to end

f4 to execute

# Undo

c-\_

# Git

<https://git-scm.com/docs>

<http://ftp.newartisans.com/pub/git.from.bottom.up.pdf>

Make your own local repository by copying from a remote:

git clone -o MY\_NAME\_FOR\_REMOTE -b BRANCH <https://github.com/REPO.git> MY\_DIR\_TO\_WRITE\_TO

Ask for a list of references on any remote repository

git ls-remote REMOTE\_NAME\_OR\_URL

If you're in a git repository that you cloned, you can leave off the REMOTE\_NAME\_OR\_URL and it'll default to the remote you cloned from.

References are "alias" to sha1 commit ID numbers.

If you're inside a git directory, you can list the remotes your local git is tracking (most likely from cloning, pushing, or pulling)

git remote

If there isn't one listed that you want, you can add it with

git remote add WHATEVER\_NAME URL

After adding a remote, you can fetch and merge it into your repo

Fetch references (branches, tags, updated objects, etc.) from a remote repo to your repo

git fetch REMOTE\_NAME\_OR\_URL

Fetching will get all the changes in references from the remote but it will not touch files in your working directory. You can merge the changes yourself after the fetch with:

git merge REMOTE/BRANCH

merge will merge and commit to your local branch unless you run it with --no-commit option.

This tells me what remote branches I am tracking in my local repo and where it'll push and pull from

git remote show REMOTE\_NAME

List your remotes

git remote -v

Change which the URL of your remotes

git remote set-url REMOTE\_NAME NEW\_URL

Rename your remote

git remote rename OLD\_NAME NEW\_NAME

This tells me which local branch I am currently using

git branch -vv

If I want to add another remote for my local to track:

git checkout --track REMOTE/BRANCH\_IN\_REMOTE

(above) will create a local branch for me and then I can switch to it in my local with

git checkout BRANCH

Once I am in my local branch I can reconfigure where it pushes and pulls from

git branch -u REMOTE/SOME\_OTHER\_BRANCH\_IN\_REMOTE

One of the more helpful options is -p, which shows the difference introduced in each commit. You can also use -2, which limits the output to only the last two entries:

git log -p -2

List the branches you have

git branch

Synching your forked repo with the original

1. Make sure you added the original as a remote (assume you named it ‘origin’)
2. Fetch the original remote/branch (creates a new branch locally for you)
3. Switch over to your forked branch
4. Merge in changes from the original branch

git remote add YOUR\_ORIGIN\_REMOTE\_NAME YOUR\_ORIGIN\_REMOTE\_URL

git fetch YOUR\_ORIGIN\_REMOTE\_NAME

git checkout YOUR\_FORK\_BRANCH

git merge YOUR\_ORIGIN\_REMOTE\_NAME/YOUR\_ORIGIN\_BRANCH\_NAME

List what is checked into Git

git ls-files

Revert a file back to is checked in

git checkout FILE

Remove files from git (not from your working directory)

git rm FILE

# Kafka

Create a topic

/Applications/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --partitions 1 --replication-factor 1 --topic mytopic

List all topics

/Applications/kafka/bin/kafka-topics.sh --list --zookeeper localhost:2181

Write to topic

/Applications/kafka/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic mytopic

Read from topic

/Applications/kafka/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic mytopic --from-beginning