



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

DIT092 - Tutorial Session: Version Control and Issue Reporting

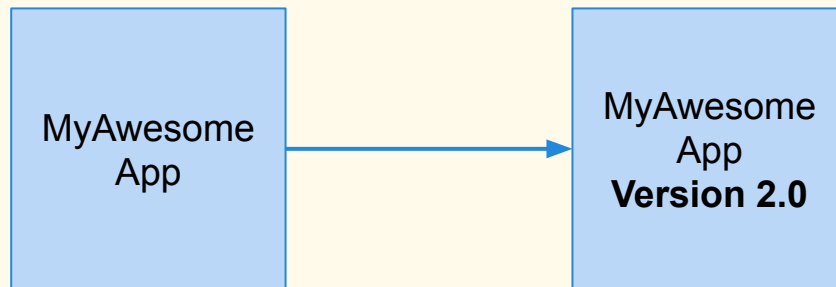
Gregory Gay

Managing Change

- Requirements, design, code, and tests evolve throughout development, and after release.
- Effective engineers control the impact of change.
 - and ensure that all artifacts evolve when any change.
- Today - **Version Control and Issue Tracking**
 - Controlling the evolution of code.

Code Evolution

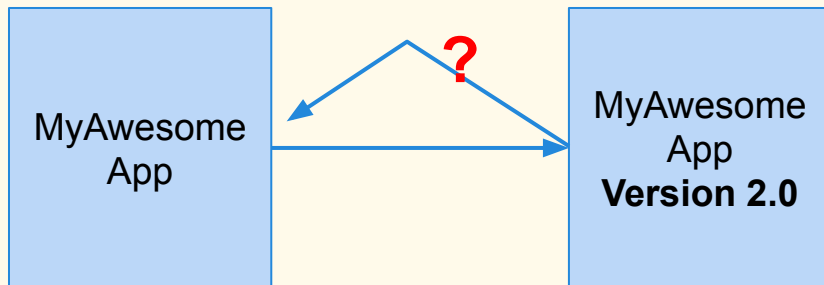
- I want to add a feature to MyAwesomeApp.



- Open in my IDE, edit the code, save.
- Done, right?**
 - Let's test it out...

Complicating Factors

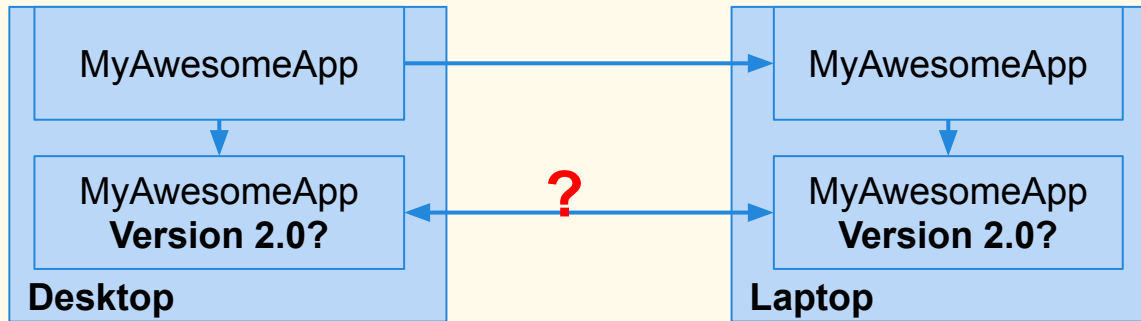
- What if the changes don't work?



- Can we go back to what we had before?
 - Especially hard if we've made a series of changes.
 - When do we overwrite a backup?

Complicating Factors

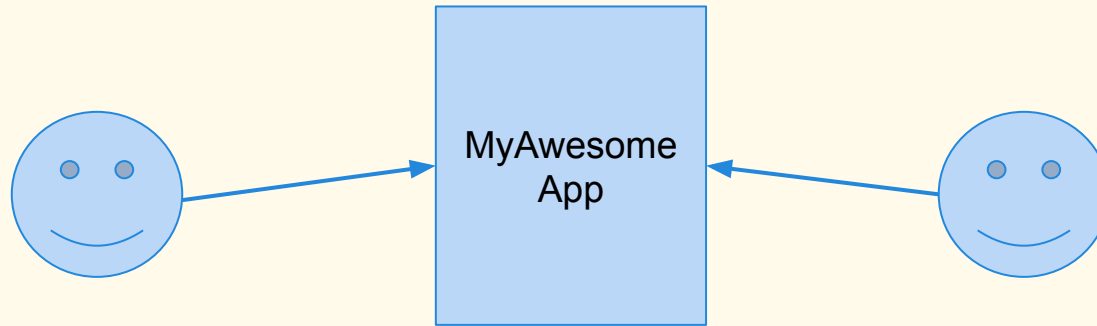
- What if I want to work on two computers?



- Did you copy over all files?
- Did you remember to sync changes?
 - What if you forgot to sync and edited files on both?

Complicating Factors

- What if multiple people are working on the project?



- What happens if multiple people edit the same file?
- When do you synchronize?
- How do you reconcile conflicting changes?

Version Control

- A version control system (VCS) tracks file changes.
 - Tracks addition, deletion, and modification of files.
 - Allows users to share and integrate these changes.
 - Enables work to progress on multiple machines.
 - Allows backup rollback to previous versions.
- VCS allows collaboration.
 - Each person edits their own copies of files and chooses when to share those changes.
 - Edits to the same file can be combined.

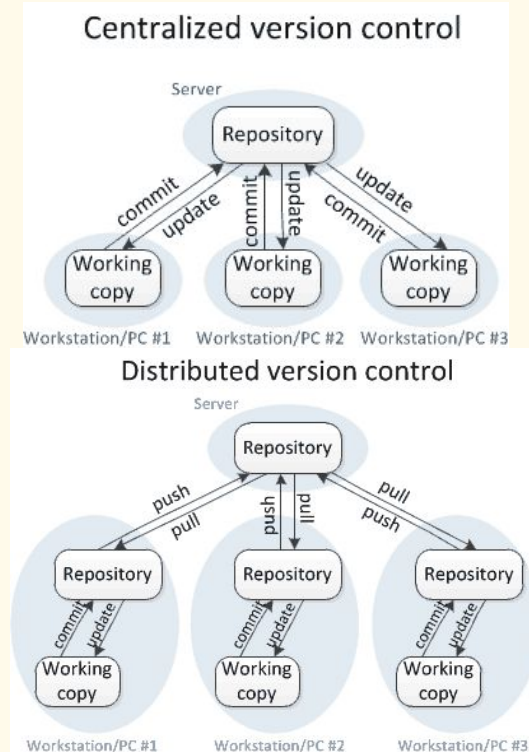
Repositories and Working Copies

- The repository is a database of changes.
- The working copy is your copy of the files
 - This is where you do your work.
- The repository tracks edits over time, and dictates what the “official” version contains.
 - You update that version by committing local changes to the repository.
 - Incorporate changes by updating the working copy.



Centralized and Distributed VCS

- In centralized VCS, the repository is stored on a central server.
- In decentralized VCS, all working copies are paired with a copy of the repository.
 - A central repository is used to coordinate and synchronize the local repositories.
 - **Git** is a decentralized system.



Basic Workflow

- To establish a working copy, clone the repository.
 - Downloads current version of all files to your machine.
 - **git clone (URL, ex: `https://github.com/User/Project.git`)**
- Make changes to files.
- Tell the VCS what you have changed.
 - This is called staging.
 - **git add (file or directory name, ex: `MyFile.java`)**

Committing Changes

- A commit takes a snapshot of the project.
 - The commit command takes the set of changes and writes them to the repository.
 - **git commit -m “Message”**
- Commits form a history of snapshots that can be revisited at any time.
- The commit must be pushed to all other users.
 - **git push origin master**

Commit Best Practices

- Commits are tagged with an explanatory message.
 - **git commit -m “Fixes issue #1337 by adding a configuration flag”**
- Use descriptive commit messages.
 - These messages indicate the purpose of the change and are used to find the right commit.
- If commit relates to a issue, refer to the issue ID.
- Describe what the commit does - use verbs.

Commit Best Practices

- Each commit should have a single purpose and should completely implement that purpose.
 - One feature change, bug fix, redesigned class.
 - Makes it easier to locate changes related to a particular feature or bug fix.
 - Allows isolated analysis of changes.
 - Don't fix a fault at the same time you introduce new functionality.
- What if you fix an issue before finishing a change?
 - git commit can selectively commit files.

Commit Best Practices

- Avoid indiscriminate commits.
 - Specify specific files to commit.
 - Do not commit all changed files unless you intend to.
 - Do not include temporary debugging changes.
- To avoid committing more than intended:
 - **git status** lists all modified files.
 - **git diff** shows specific changes made.
 - **git commit file1 file2 -m “Message”** only commits the named files.

Commit Best Practices

- VCS is intended for the files people edit.
- Do not commit generated files.
 - .class, .o files. Temporary or compiled artifacts (.pdf).
 - Users can regenerate them.
 - Generated files are prone to conflict.
 - Generated files tend to be binary files.
 - VCS cannot identify differences between versions of binary files.
- A **.gitignore file** lists filenames to ignore.

Incorporating Changes

- An update operation applies the changes made in all commits since your last update.
 - You commit, they update.
 - **git pull** incorporates all commits made by other users.
 - **Commit** moves changes between the working copy and the local repository. **Push** moves changes between local and central repositories.
 - **Pull** moves changes from the central repository to the local repository (and applies them to the working copy).

Update Best Practices

- Incorporate changes frequently.
 - Run **git pull** before making any changes.
 - If someone has made a change before you start to edit, it is a waste of time to make changes then have to resolve conflicts.
 - Run **git pull** before **git push**.
 - This allows you to resolve conflicts quickly.
- Also push changes frequently!
 - Once you have completed work, share those changes with all copies of the repository.

Conflicts

- A conflict occurs if two users make simultaneous, different changes to the same line of a file.
 - Manual intervention is needed to resolve a conflict.
- Changes 1 & 2 are simultaneous if:
 - User A makes Change 1 before performing the update that brings in Change 2.
 - User B makes Change 2 before performing the update that brings in Change 1.
- Frequent push and pull help avoid conflicts.

Conflicts

- Remember that VCS tools are line-based.
- Be careful with adjusting indentation.
 - Check for accidental indentation changes.
- Avoid excessively long lines.
 - Longer lines are more likely to cause conflicts.
 - More people will edit that line.
 - The more characters, the harder it is to determine the exact changes when viewing VCS history.
 - Shorter lines are easier to read when editing a file.

Summarizing Basic Workflow

- **git clone** (on initialization)
- **git pull** (on new change)
- Make local edits.
 - **git diff**, **git status** to examine changes made
- **git add**
- **git commit**
- **git pull** (before you push)
- **git push**

Not Just for Source Code

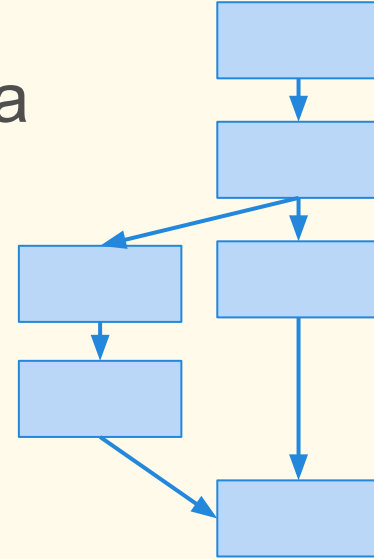
- VCS can also be used for requirements, design, and other documentation.
- Update and commit workflow can be used for any type of file.
 - Binary files (.pdf, .doc, images) are complicated as there is no way to identify specific changes to files.
 - VCS stores newest version.
 - Machine-readable files have no issues (.txt, .csv).

Branches

- The history of commits forms a tree.
- The core version is called the **master** branch.
 - The master branch should be relatively stable, tested, and free of faults (as far as we know).
- How can we add new features or redesign elements of the system while keeping the master branch stable?
 - VCS allows the creation of new **branches**.

Branches

- Creating a branch takes current commit of a branch as the start of a new timeline.
 - Current commit to any existing branch.
 - Commits to the new branch are not applied to the original branch.
 - New commits to the original branch are not applied to the new branch.
- The new branch can be merged into the original.
 - May require handling conflicts.



Branches

- These are known as feature branches.
 - Insulated from master branch, so we can make changes without impacting stability for other developers.
 - Once complete, we can merge the working change into the master branch
- Allows experimentation without breaking anything or impeding others.
 - Many projects have a stable master branch, a development branch, and feature branches.
 - Enables control over development.

Branches

- To create or switch to a branch:
 - **git checkout -b <new branch name> [<base branch name>]**
 - [<base branch name>] is optional, default value is master.
- To push changes:
 - **git push origin <new branch name>**
 - **git push origin master** commits to the **master** branch.
- **git pull** incorporates changes to all branches.
- **git commit** commits to the “checked out” branch.

Forks

- A **fork** is a copy of a complete repository.
 - Contains all past commits and branches.
 - New commits to the original repository not applied to fork.
 - New commits to fork are not applied to the original.
 - Commits can be selectively merged into the original.
- Used when a team wants to create an independent spinoff and may not plan to merge their work.
 - Often used in open-source by amateurs to extend a project they are not a member of.

Merging Branches or Forks

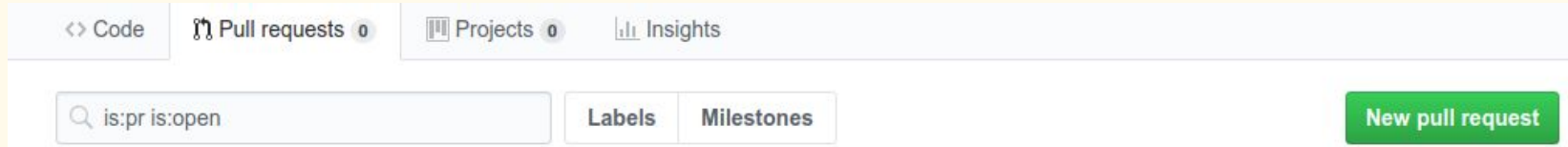
- Pull requests are used to tell others about changes you would like to merge into a branch.
 - Once sent, interested parties can review changes, discuss them, and push additional follow-up commits before agreeing to merge.
 - Performed through the web interface (i.e., Github).
- Once conflicts are resolved, target branch will be updated by applying commits from source branch.

Pull Requests

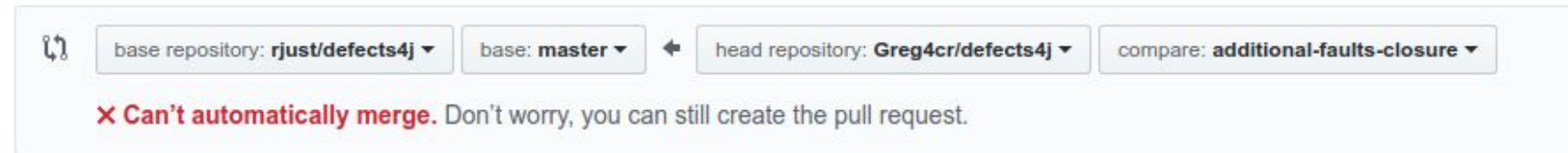
- Common tool in open-source development.
 - Fork the project to create a “personal copy” or create a branch in the core repository.
 - Incorporate your changes.
 - Issue a pull request to the core repository/master branch.
 - Same mechanism is used for both fork-to-original and branch-to-branch (i.e., new branch-to-master) merging.
 - Project maintainers discuss the pull request and assess whether it fits the project (and does not break anything).
 - Project maintainers agree to incorporate changes.

Pull Requests

- To create pull request, you must have changes committed to your branch.
- Go to the core repository page, click the “Pull requests” tab, and click "New pull request" button.

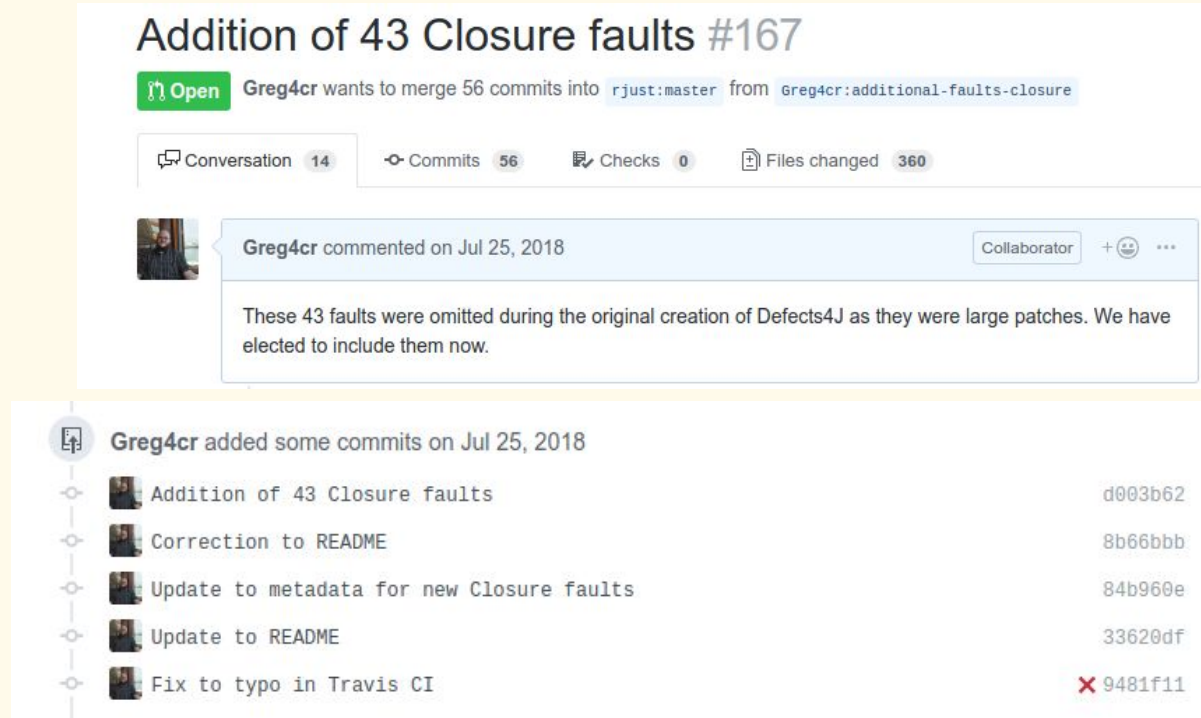


- Pick the source branch using the "head repository" and “compare” dropdowns. Pick the target branch using the “base repository” and “base” dropdowns.








Pull Requests


- Fill out a title and description of the changes.
- Commits that will be applied to the target are listed.




Addition of 43 Closure faults #167











 **Open** Greg4cr wants to merge 56 commits into `rjust:master` from `Greg4cr:additional-faults-closure`

 Conversation **14**  Commits **56**  Checks **0**  Files changed **360**

 **Greg4cr** commented on Jul 25, 2018 Collaborator + 🗨️ ⋮

These 43 faults were omitted during the original creation of Defects4J as they were large patches. We have elected to include them now.


 **Greg4cr** added some commits on Jul 25, 2018





  Addition of 43 Closure faults	d003b62
  Correction to README	8b66bbb
  Update to metadata for new Closure faults	84b960e
  Update to README	33620df
  Fix to typo in Travis CI	✗ 9481f11

Pull Requests

- Contributors may discuss or review the request.
- Conflicts may need to be resolved.
- When ready, click the “Merge pull request” button.

Add more commits by pushing to the **additional-faults-closure** branch on **Greg4cr/defects4j**.



-  **Review requested** [Show all reviewers](#)
Review has been requested on this pull request. It is not required to merge. [Learn more](#).
-  **Some checks haven't completed yet** [Hide all checks](#)
1 pending check
 -  **continuous-integration/travis-ci/pr** Pending — The Travis CI build is in progress [Details](#)
-  **This branch has conflicts that must be resolved** [Resolve conflicts](#)
Use the [web editor](#) or the [command line](#) to resolve conflicts.
Conflicting files
README.md

[Merge pull request](#) or [view command line instructions](#).

Issue Tracking

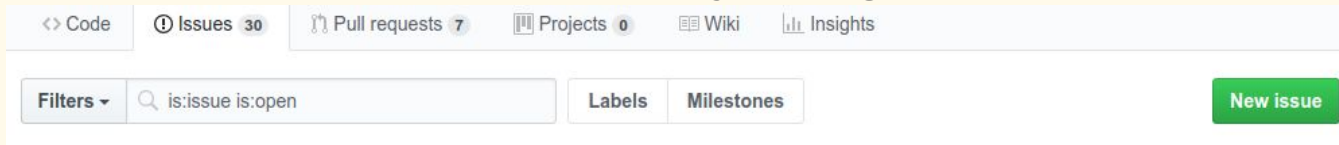
- Allows developers and users to report and document software issues.
 - Used to record information about the issue and the status of fixing that issue.
 - Also often used to request new features and track progress on implementation of these features.
- Often available as part of project management software (i.e., Gitlab).

Issue Tracking

- **What happened:** The erroneous behavior.
- **How to reproduce the issue:** Steps (and often test cases) that will demonstrate the behavior.
- **The severity of the issue:** A rating of how badly this affects users (and how widespread it is).
- **When the issue was reported:** Allows tracking of age of issues, and measuring fix time.
- **Responsibility:** Who reported the issue and who has been tasked with fixing the issue.


Filing an Issue

- Go to the “Issues” tab on the project page, and click “New issue.”



- Fill out a title and description.
 - Be sure to include both what happened and steps to reproduce.

JsonElement#getAsCharacter() #1355

 Closed

iinc opened this issue on Jul 24, 2018 · 1 comment



iinc commented on Jul 24, 2018

JsonElement
public char getAsCharacter()
“convenience method to get this element as a primitive character value.”

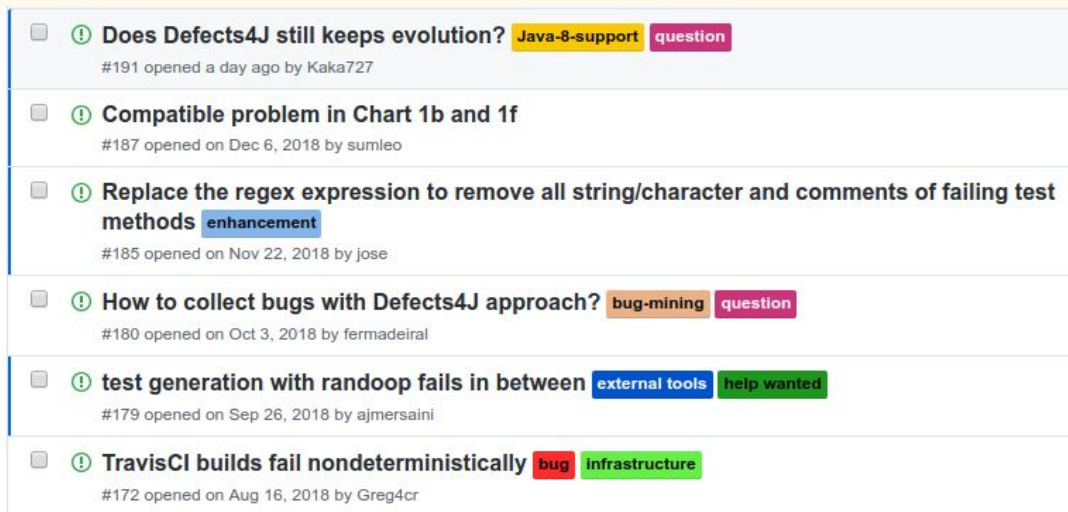
I expected calling getAsCharacter would convert an integer to its character value. Instead, it returns the first character of the toString().

```
JsonPrimitive jp = new JsonPrimitive(100);  
System.out.println(jp.getAsCharacter());      Output: 1  
System.out.println((char) jp.getAsInt());     Output: d
```

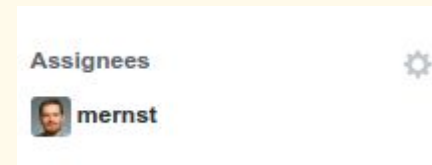
Perhaps the documentation should be more descriptive.

Filing an Issue

- Issues can be labeled.
 - Common: “bug”, “enhancement”



- Users can be assigned to fix the issue.



Filing an Issue


- Refer to report IDs in commit messages and pull requests addressing such issues.
- Allows traceability, showing progress in addressing issues.

Fix JPMS module setup (fixes #1315) (#1402)

* Fix JPMS module setup (fixes #1315)

* Re-added cause to AssertionErrors

Fix JPMS module setup (fixes #1315) #1402

 **Merged** inder123 merged 2 commits into `google:master` from `unknown repository` on Oct 18, 2018

Move module-info.java to /src/main/java #1315

 **Closed** cayhorstmann opened this issue on May 9, 2018 · 8 comments

Issue Reporting Advice

- Use descriptive subjects.
 - Issue title is used to get an idea of the contents at a glance. Convey what exactly is wrong in the title.
 - Tracker might have hundreds of issues. Important to let developers search information efficiently.
- Focus on the facts.
 - Do not speculate on the cause of an issue if you can avoid it. Give exact facts rather than guessing.
 - If you have an educated guess, you can provide it. Just be careful, as these often are wrong.

Issue Reporting Advice

- Describe what you expected to happen when you used the system. Some issues are subjective.
 - Describe what actually happened. Include output text, a screenshot, a URL if you can.
 - “I can’t log in” versus “I’m at (URL) and I’m trying to log in using my username, (username). But the system is saying that my username can’t be found, and it’s taking me to (URL).”
- Check whether an issue was already reported.
- Include OS and hardware information.

Summary

- Version control can be used to control how code and other artifacts evolve.
 - A history of snapshots is maintained.
 - Mistakes can be easily resolved by rolling back.
 - Allows multiple developers to collaborate.
 - Branches and forks allow experimental development while maintaining a stable version.



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY