Code Reviews

Brian Joo, Trevor Parke, Nathan Bunge

What is a code review?

Definition: Code review is a process where one or more people examine and evaluate a software codebase to find and fix errors, improve quality, and ensure compliance with coding standards.

- Peer review to judge the quality of code.
- A mandatory step in the process of software development to ensure communication between teams and to improve the code before it is shipped

Why are code reviews important?

- In the real world, 99 percent of the time you will be writing code with others on a project.
- Improves code quality.
- Other benefits.
  - Discover bugs earlier
  - Maintain compliance
  - Enhance security
- In a healthy environment, code reviews can also be a learning experience.
  - People don’t write the same code. Because of this, they can bring new ideas to improve what you have.
  - People can notify you if there is a potential error that you did not see or test for.
- Enhancing security: Code reviews can help identify security vulnerabilities and ensure that the code is secure.

Disadvantages of code reviews

- Time-Consuming: Code reviews can be time-consuming, especially for large or complex codebases.
- Subjectivity: Code reviews can be subjective, and reviewers may have different opinions on what constitutes good or bad code.
- Negative Feedback: Code reviews can sometimes result in negative feedback
- Over-Reliance on Reviews: Some teams may become overly reliant on code reviews as a way to catch errors or improve quality.

Some best practices

- Set clear objectives and expectations for the review process.
- Choose the right reviewers based on their expertise and knowledge.
- Use a checklist or guidelines to ensure that all important aspects of the code are reviewed.
- Provide constructive feedback that is specific, actionable, and respectful.
- Follow up on feedback to ensure that issues are resolved and improvements are made.
- Be professional!

Different areas to checklist

- Design principle
  - Is the code reusable.
  - Are we following the High cohesion and low coupling train of thought.
  - Is the code properly designed?
- Documentation and maintainability
  - Is the code well documented?
  - Is the code readable and not need too many comments?
  - Is the code unnecessarily complex?
- Testing
  - Do we have test cases?
  - Do we have boundary cases?
- Performance
  - Time complexity
  - Space complexity

C# Checklist

- Disposing of **Unmanaged Resources**
- Proper implementation of **Exception Handling**
- Consistent **naming**
- **Methods** have less than 20 lines of code
- Minimal **nesting for/if** statements
- Use **constants** and **readonly** wherever applicable.
- All **unused usings** need to be removed
- **null check** needs to be performed wherever applicable
- Extract a **method** if the same piece of code is being used more than once
- Use **anonymous** typing if code is going to be used only once.
- Use proper **access specifiers**
- Use **interfaces** wherever needed to maintain decoupling.
- Use a **Stringbuilder** instead of **string** if multiple concatenations are required, to save heap memory
- No unreachable code
- Avoid **type casting** and **type conversions** as much as possible

Different types of code reviews

1. Manual Code Review: Developers review code manually, either individually or in groups, to identify issues and provide feedback.
2. Pair Programming: Two developers work together on a single computer to write code, review each other's work, and provide feedback in real-time.
3. Over-the-shoulder reviews
4. Tool-assisted Code Review: Developers use specialized tools to analyze code for errors, bugs, and security vulnerabilities.
5. Email pass-around

Manual Code Review

- The most traditional code review where whole teams or subgroups of the development team get together with admins and manually go through code and discuss. This is usually done in conjunction with the other types of code review.
- Pros:
  - The most thorough and complete kind of review
  - Best for driving further development by guiding future code
  - Time for teams to touch base and discuss
- Cons:
  - Can take a long time, block up a lot of people at once

Pair Programming

- Requires 2 developers to work in real time.
- One is the writer, the other is the reviewer. Swap roles often.
- Pros
    - Can be done remote.
    - Immediate feedback.
    - Fosters good communication between programmers.
- Cons
    - Can be overused.
    - Difficult to measure.
    - Takes more work hours to produce results

Over-the-shoulder reviews

- Similar to pair programming, there are 2 parties. The author and the reviewer.
- The reviewers role/responsibility is to ask questions and make suggestions.
- The author can either take the advice immediately or write it down and do later; however, in the end, it's up to the author to make the choice.
- Pro
    - Can be done remote.
    - Faster than pair programming.
- Con
    - Reviewer is detached from code
    - Speed of progress is entirely on the author.

Email pass-around

- Used for a small amount of code or a minor issue.
- Author sends a piece of code to different reviewers via email.
- Pros
    - Easy implementation and completion
    - Facilitates remote, asynchronous reviews
- Cons
    - Time consuming to gather files
    - Difficult to follow conversations

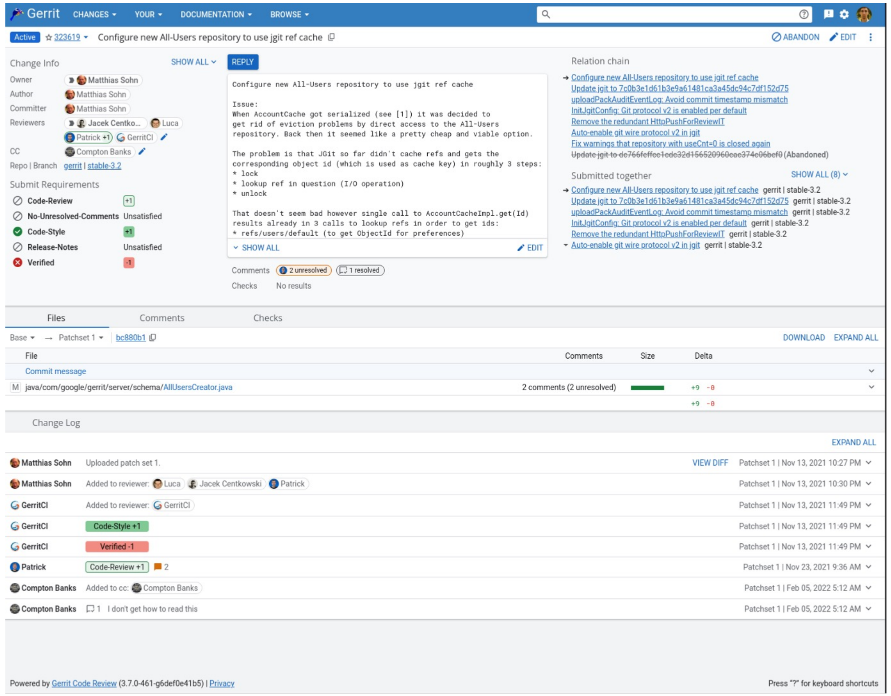
Tool-assisted reviews

- Uses non human help to decide if code is “good”.
- Tools can be extremely fast and compute many different factors.
- It is important to not solely rely on tools, but to combine it with other types of reviews.
- Pros
  - Easier to gather metrics
  - More time for the developer to do other tasks.
- Cons
  - Can be expensive depending on the tool

Types of Code Review Tools

Gerrit:

- Git based, web hosted code review
- Adds a step to the process of pushing to a remote source
- Code will not be merged until it has gone through review
- Works closely with the usual workflow
- `git push origin HEAD:refs/for/<branch_name>`
- remote: http://gerrithost/#/c/RecipeBook/+/702 Change to a proper, yeast based pizza dough.



Active ☆ 323619 Configure new All-Users repository to use jgit ref cache

Change Info

Owner	Matthias Sohn
Author	Matthias Sohn
Committer	Matthias Sohn
Reviewers	Jacek Centko... Luca Patrick +1 GerritCI
CC	Compton Banks
Repo | Branch	gerrit | stable-3.2

Submit Requirements

Code-Review	+1
No-Unresolved-Comments	Unsatisfied
Code-Style	+1
Release-Notes	Unsatisfied
Verified	-1

```
Configure new All-Users repository to use jgit ref cache

Issue:
When AccountCache got serialized (see [1]) it was decided to
get rid of eviction problems by direct access to the All-Users
repository. Back then it seemed like a pretty cheap and viable option.

The problem is that JGit so far didn't cache refs and gets the
corresponding object id (which is used as cache key) in roughly 3 steps:
* lock
* lookup ref in question (I/O operation)
* unlock

That doesn't seem bad however single call to AccountCacheImpl.get(Id)
results already in 3 calls to lookup refs in order to get ids:
* refs/users/default (to get ObjectId for preferences)
```

Comments 2 unresolved 1 resolved
Checks No results

Relation chain

→ Configure new All-Users repository to use jgit ref cache
Update jgit to 7c0b3e1d61b3e9a61481ca3a45dc94c7df152d75
uploadPackAuditEventLog: Avoid commit timestamp mismatch
InitJgitConfig: Git protocol v2 is enabled per default
Remove the redundant HttpPushForReviewIT
Auto-enable git wire protocol v2 in jgit
Fix warnings that repository with useCnt=0 is closed again
~~Update jgit to dc766feffce1cdc32d156520960cac374c06bef0~~ (Abandoned)

Submitted together SHOW ALL (8)

→ Configure new All-Users repository to use jgit ref cache gerrit | stable-3.2
Update jgit to 7c0b3e1d61b3e9a61481ca3a45dc94c7df152d75 gerrit | stable-3.2
uploadPackAuditEventLog: Avoid commit timestamp mismatch gerrit | stable-3.2
InitJgitConfig: Git protocol v2 is enabled per default gerrit | stable-3.2
Remove the redundant HttpPushForReviewIT gerrit | stable-3.2
Auto-enable git wire protocol v2 in jgit gerrit | stable-3.2

Files Comments Checks

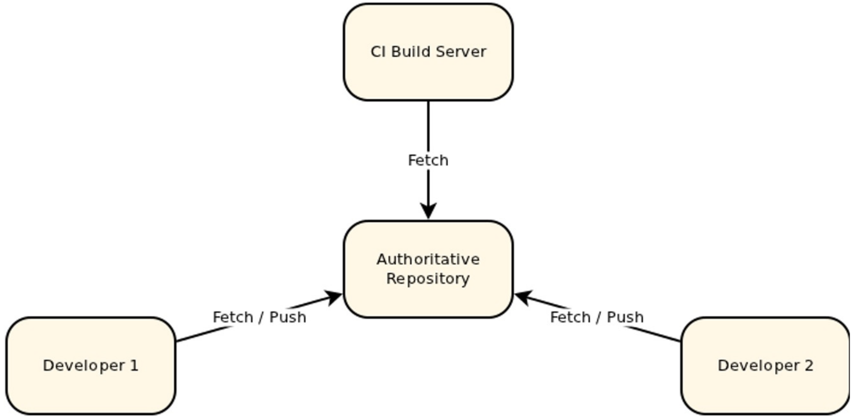
Base → Patchset 1 bc880b1 DOWNLOAD EXPAND ALL

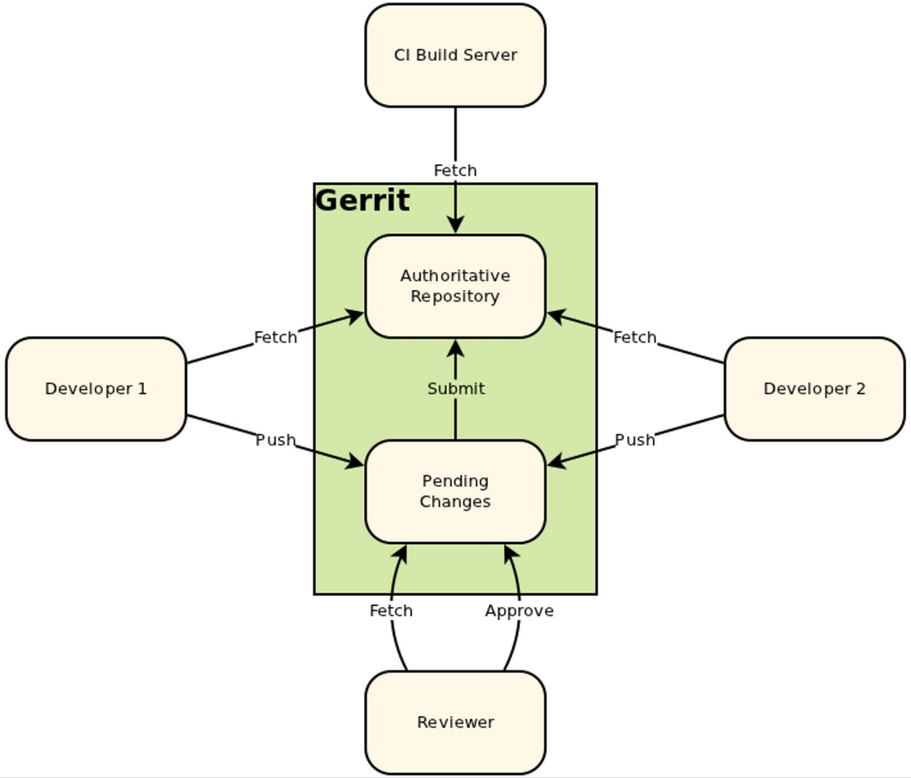
File	Comments	Size	Delta
Commit message			
M java/com/google/gerrit/server/schema/AllUsersCreator.java	2 comments (2 unresolved)		+9 -0
			+9 -0

Change Log EXPAND ALL

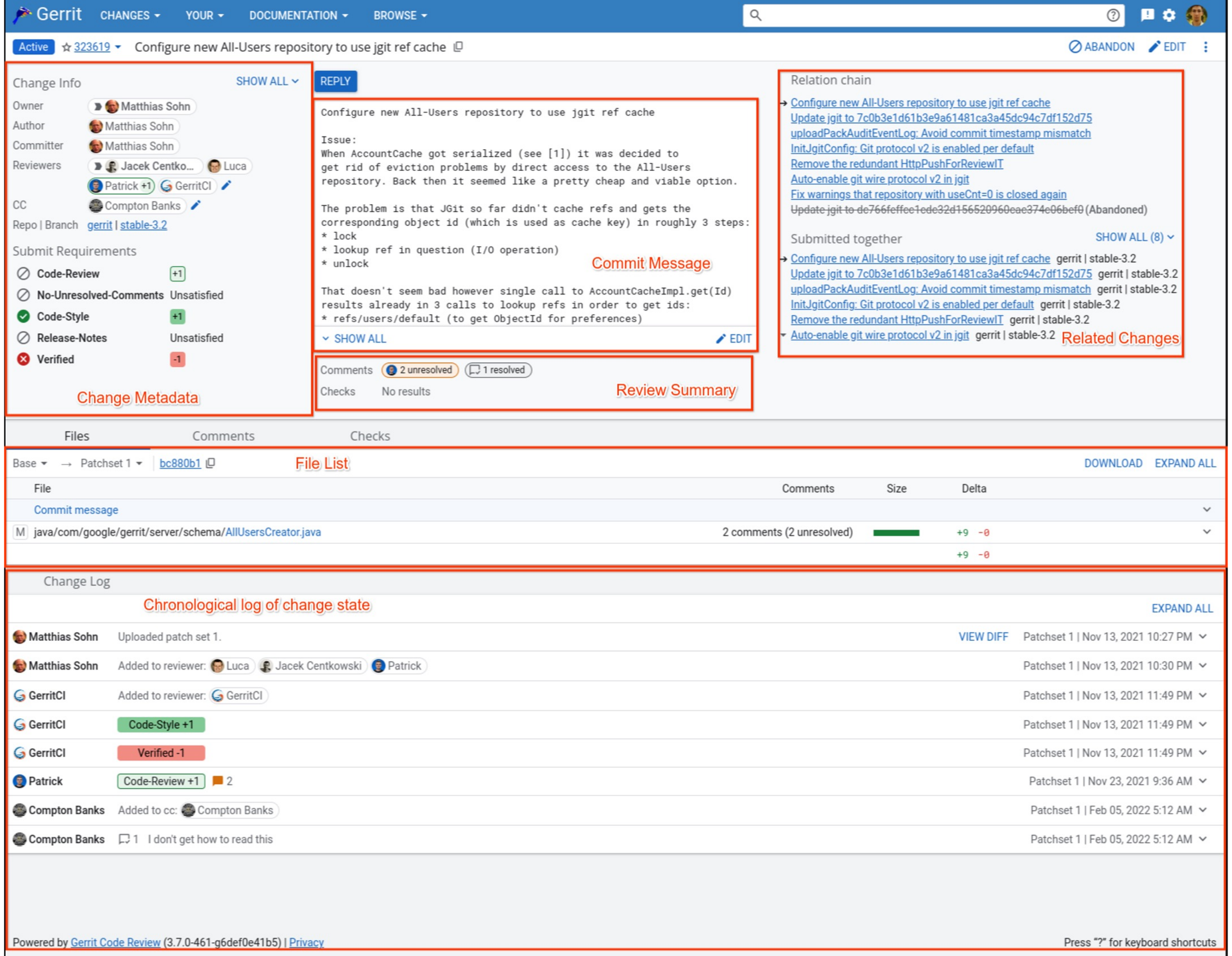
Matthias Sohn	Uploaded patch set 1. VIEW DIFF	Patchset 1 | Nov 13, 2021 10:27 PM
Matthias Sohn	Added to reviewer: Luca Jacek Centkowski Patrick	Patchset 1 | Nov 13, 2021 10:30 PM
GerritCI	Added to reviewer: GerritCI	Patchset 1 | Nov 13, 2021 11:49 PM
GerritCI	Code-Style +1	Patchset 1 | Nov 13, 2021 11:49 PM
GerritCI	Verified -1	Patchset 1 | Nov 13, 2021 11:49 PM
Patrick	Code-Review +1 2	Patchset 1 | Nov 23, 2021 9:36 AM
Compton Banks	Added to cc: Compton Banks	Patchset 1 | Feb 05, 2022 5:12 AM
Compton Banks	1 I don't get how to read this	Patchset 1 | Feb 05, 2022 5:12 AM

Powered by Gerrit Code Review (3.7.0-461-g6def0e41b5) | Privacy   Press "?" for keyboard shortcuts





Gerrit Central Repository



Gerrit CHANGES YOUR DOCUMENTATION BROWSE

Active ☆ 323619 Configure new All-Users repository to use jgit ref cache ABANDON EDIT

Change Info SHOW ALL

Owner	Matthias Sohn
Author	Matthias Sohn
Committer	Matthias Sohn
Reviewers	Jacek Centko... Luca Patrick +1 GerritCI
CC	Compton Banks
Repo | Branch	gerrit | stable-3.2

Submit Requirements

Code-Review	+1
No-Unresolved-Comments	Unsatisfied
Code-Style	+1
Release-Notes	Unsatisfied
Verified	-1

REPLY

```
Configure new All-Users repository to use jgit ref cache

Issue:
When AccountCache got serialized (see [1]) it was decided to
get rid of eviction problems by direct access to the All-Users
repository. Back then it seemed like a pretty cheap and viable option.

The problem is that JGit so far didn't cache refs and gets the
corresponding object id (which is used as cache key) in roughly 3 steps:
* lock
* lookup ref in question (I/O operation)
* unlock

That doesn't seem bad however single call to AccountCacheImpl.get(Id)
results already in 3 calls to lookup refs in order to get ids:
* refs/users/default (to get ObjectId for preferences)
```

SHOW ALL EDIT

Comments 2 unresolved 1 resolved

Checks No results

Relation chain

→ Configure new All-Users repository to use jgit ref cache
Update jgit to 7c0b3e1d61b3e9a61481ca3a45dc94c7df152d75
uploadPackAuditEventLog: Avoid commit timestamp mismatch
InitJgitConfig: Git protocol v2 is enabled per default
Remove the redundant HttpPushForReviewIT
Auto-enable git wire protocol v2 in jgit
Fix warnings that repository with useCnt=0 is closed again
~~Update jgit to dc766feffce1ede32d156520960cae374c06bef0~~ (Abandoned)

Submitted together SHOW ALL (8)

→ Configure new All-Users repository to use jgit ref cache gerrit | stable-3.2
Update jgit to 7c0b3e1d61b3e9a61481ca3a45dc94c7df152d75 gerrit | stable-3.2
uploadPackAuditEventLog: Avoid commit timestamp mismatch gerrit | stable-3.2
InitJgitConfig: Git protocol v2 is enabled per default gerrit | stable-3.2
Remove the redundant HttpPushForReviewIT gerrit | stable-3.2
Auto-enable git wire protocol v2 in jgit gerrit | stable-3.2

Files Comments Checks

Base → Patchset 1 bc880b1 DOWNLOAD EXPAND ALL

File	Comments	Size	Delta
Commit message			
M java/com/google/gerrit/server/schema/AllUsersCreator.java	2 comments (2 unresolved)		+9 -0
			+9 -0

Change Log EXPAND ALL

Matthias Sohn	Uploaded patch set 1. VIEW DIFF	Patchset 1 | Nov 13, 2021 10:27 PM
Matthias Sohn	Added to reviewer: Luca Jacek Centkowski Patrick	Patchset 1 | Nov 13, 2021 10:30 PM
GerritCI	Added to reviewer: GerritCI	Patchset 1 | Nov 13, 2021 11:49 PM
GerritCI	Code-Style +1	Patchset 1 | Nov 13, 2021 11:49 PM
GerritCI	Verified -1	Patchset 1 | Nov 13, 2021 11:49 PM
Patrick	Code-Review +1 2	Patchset 1 | Nov 23, 2021 9:36 AM
Compton Banks	Added to cc: Compton Banks	Patchset 1 | Feb 05, 2022 5:12 AM
Compton Banks	1 I don't get how to read this	Patchset 1 | Feb 05, 2022 5:12 AM

Powered by Gerrit Code Review (3.7.0-461-g6def0e41b5) | Privacy Press "?" for keyboard shortcuts

Code Review through git

An alternative way to perform code reviews is to use pull requests, e.g.:

- Create a branch for the feature/bug you are working on
- Once you are done, commit your work and push it to your repository
- Create a pull request from your branch
- Describe what the work is about
- Select reviewers
- Address the reviewers comments until the code is accepted
- The reviewer/verifier can then merge the code to the master

And of course even simpler is to just comment on the code.

Expectations, Outcomes, and Challenges of Modern Code Review - A Case Study

- Researchers wanted to answer 3 main questions

1) What are the motivations and expectations for modern code review? Do they change from managers to developers and testers?
2) What are the actual outcomes of modern code review? Do they match the expectations?
3) What are the main challenges experienced when performing modern code reviews relative to the expectations and outcomes?

- To conduct the code reviews, a Program called CodeFlow was implemented.

Question 1)What are the motivations and expectations for modern code review? Do they change from managers to developers and testers?

- The manager’s main goal was to get the assignment/task done on time.
- The developer’s main goal was to improve their skills
- The Testers main goal was to find the defects in the code
- However, despite these differences, there was a consensus among the three parties. By code reviewing, it would improve the quality of the code and help find bugs. This would hit all 3 parameters for all parties. As by code reviewing, it would allow the team to push on time, to learn from each other, and lesson the load on the testers.

Question 2) What are the actual outcomes of modern code review? Do they match the expectations?

- Pros
  - Better code quality
  - Transfer of knowledge
  - Increased productivity
  - Increased team-work
- Cons
  - Workload management
- Overall, the researchers found that code reviews are not a “get out of jail” card for companies/teams. It ultimately depends on how code reviews are used.

Question 3) What are the main challenges experienced when performing modern code reviews relative to the expectations and outcomes?

- TIme
  - Code reviews can take time, and it is important not to spend so much time on it that no other progress is made.
- Workload
  - As all members on the team have their own responsibility. It is important not to spend too much time on code reviews, to allow all members to complete their tasks.
- Communication
  - While code reviews can increase the level of communication, the code review itself can be impacted by the level/quality of the communication.
  - It is also important to remember to be honest but not critical. It was found being too critical could lower the member’s morale and drive to work.

Characteristics of Useful Code Reviews: An Empirical Study at Microsoft

- Primary goal of code reviews
  - change is free from defects
  - follows team conventions
  - solves a problem in a reasonable way
  - Result is high quality or higher than it was before
- Microsoft follows a 3 step process for better code reviews
  - 1) Ask the actual programmers what is useful
  - 2) Try any changes recommended in step 1
  - 3) Compare the results from the first 2 steps with the comments from 5 Microsoft projects(roughly 1.5 million comments).

The 3 steps

1. What are the characteristics of code review comments that are perceived as useful?
2. What methods and features are needed to automatically classify review comments into useful and not useful?
3. What factors have a relationship with the density of useful code review comments?

Result

- After conducting the interviews, the researchers found that code reviews
    - improve the maintainability of code
    - Identifies defects
    - Increase in knowledge and team awareness
- Somewhat Useful comments
    - “nit-picking issues”
        - Indentation, identifier naming, and typos
- ‘Not Useful’ comments
    - Praise
    - What needs to be done in the future

Cont..

- Like in the previous reading, code reviews are not a end all be all solution.
    - Many different factors
        - Types of comments
        - Experience
        - Team’s attention to detail
        - Level of collaboration

QUESTIONS?

Sources

https://microsoft.github.io/code-with-engineering-playbook/code-reviews/recipes/csharp/

https://docs.google.com/presentation/d/1C73UgQdzZDw0gzpaEqIC6SPujZJhqamyqO1XOHjH-uk/edit#slide=id.g4d6c16487b_1_2753

https://smartbear.com/blog/pros-and-cons-of-code-review-methods-infographic/

http://gerrit.appinventor.mit.edu/Documentation/intro-quick.html

Now for a demo

Code Review Practice

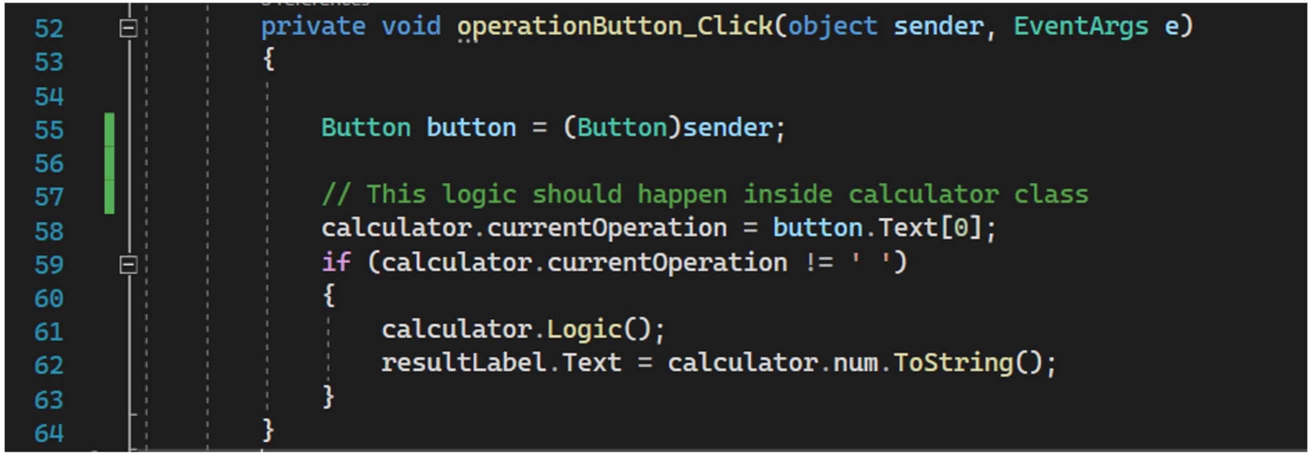
Code:https://github.com/NathanBunge/calculator/tree/calculator_logic

Checklist: https://github.com/mgreiler/code-review-checklist

- Does this code change accomplish what it is supposed to do?
- Is the code at the right abstraction level?
- Is the code modular enough?
- Is a framework, API, library, or service used that should not be used?
- Can a better solution be found in terms of maintainability, readability, performance, or security?
- Is error handling done the correct way?
- Should any logging or debugging information be added or removed?
- Can the readability of the code be improved by different function, method or variable names?
- Which parts were confusing to you and why?
- Can the readability of the code be improved by smaller methods?

Code Review Practice

Is this at the right level of abstraction?



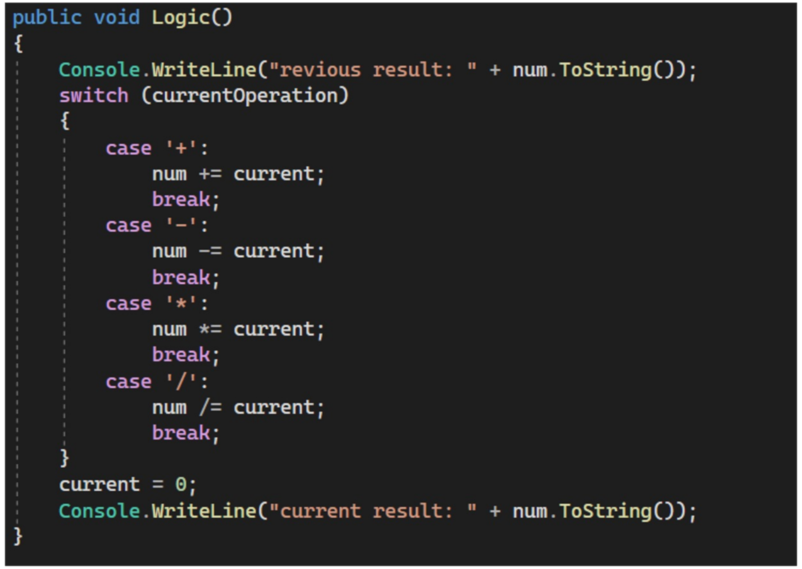
```
        private void operationButton_Click(object sender, EventArgs e)
        {

            Button button = (Button)sender;

            // This logic should happen inside calculator class
            calculator.currentOperation = button.Text[0];
            if (calculator.currentOperation != ' ')
            {
                calculator.Logic();
                resultLabel.Text = calculator.num.ToString();
            }
        }
```

You should not be changing currentOperation from the form.

Code Review Practice

Can a better solution be found in terms of readability?

Operations are code coded. Make them constants for better readability. Or modular like we did in class.

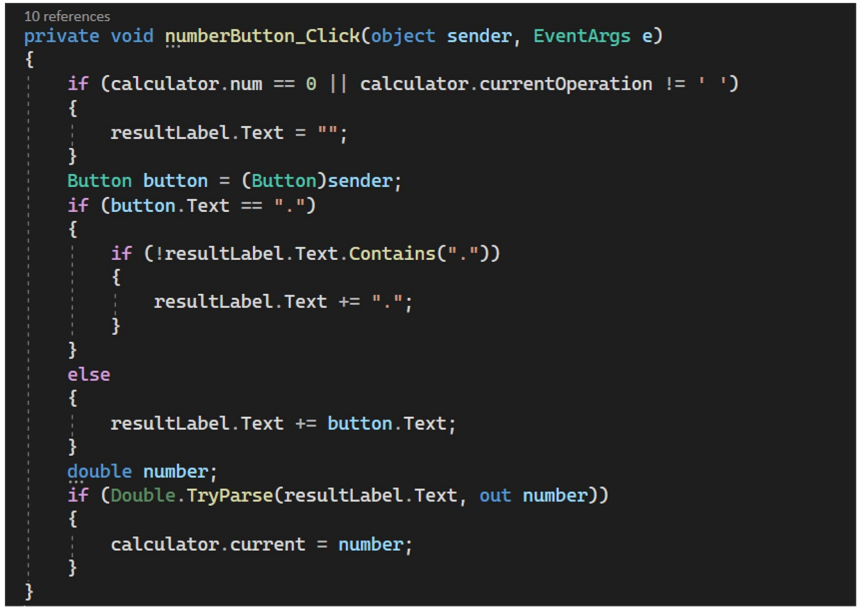


```
public void Logic()
{
    Console.WriteLine("revious result: " + num.ToString());
    switch (currentOperation)
    {
        case '+':
            num += current;
            break;
        case '-':
            num -= current;
            break;
        case '*':
            num *= current;
            break;
        case '/':
            num /= current;
            break;
    }
    current = 0;
    Console.WriteLine("current result: " + num.ToString());
}
```

Code Review Practice

Can a better solution be found in terms of performance?

You could change TryParse to only run when you are performing a calculation, instead of running on every new digit entered.

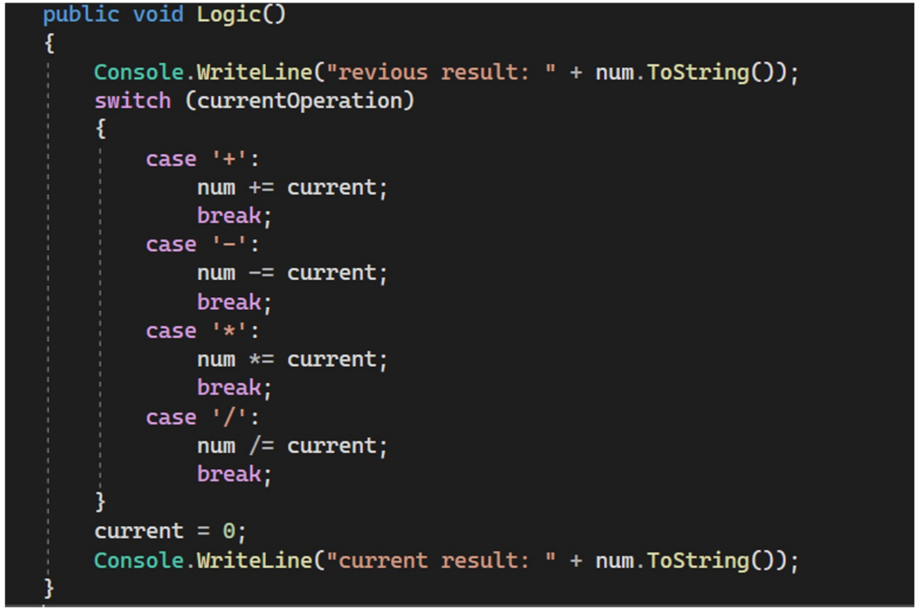


```
10 references
private void numberButton_Click(object sender, EventArgs e)
{
    if (calculator.num == 0 || calculator.currentOperation != ' ')
    {
        resultLabel.Text = "";
    }
    Button button = (Button)sender;
    if (button.Text == ".")
    {
        if (!resultLabel.Text.Contains("."))
        {
            resultLabel.Text += ".";
        }
    }
    else
    {
        resultLabel.Text += button.Text;
    }
    double number;
    if (Double.TryParse(resultLabel.Text, out number))
    {
        calculator.current = number;
    }
}
```

Code Review Practice

Is error handling done the correct way?

At the bare minimum you need a divide by zero exception. This is essential for a calculator program.

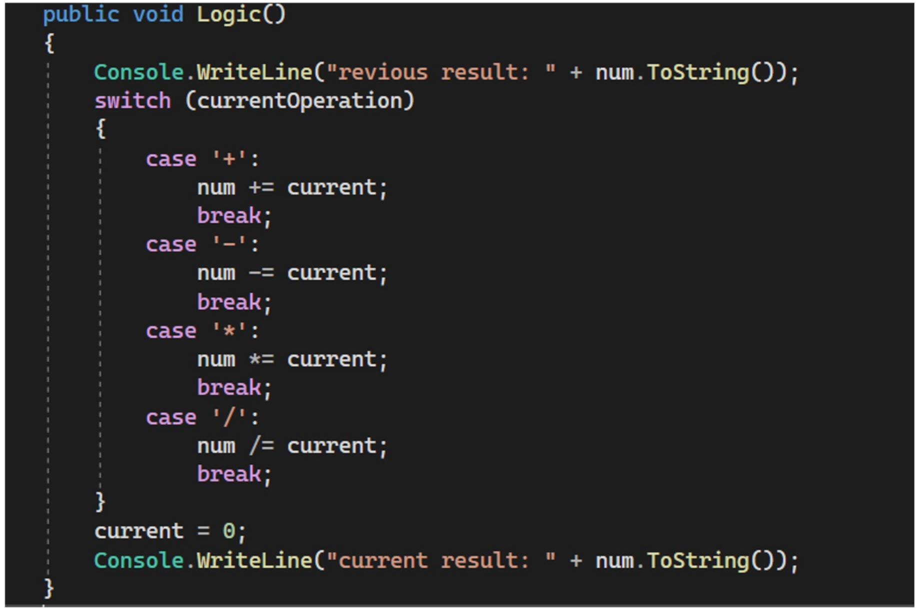


```
public void Logic()
{
    Console.WriteLine("revious result: " + num.ToString());
    switch (currentOperation)
    {
        case '+':
            num += current;
            break;
        case '-':
            num -= current;
            break;
        case '*':
            num *= current;
            break;
        case '/':
            num /= current;
            break;
    }
    current = 0;
    Console.WriteLine("current result: " + num.ToString());
}
```

Code Review Practice

Can the readability of the code be improved by different function, method or variable names?

- Logic() should be renamed to Calculate() or Operate()
- “num” could be changed to “total” or “result”
- “current” could be changed to “currentNum” or “operand”



```
public void Logic()
{
    Console.WriteLine("revious result: " + num.ToString());
    switch (currentOperation)
    {
        case '+':
            num += current;
            break;
        case '-':
            num -= current;
            break;
        case '*':
            num *= current;
            break;
        case '/':
            num /= current;
            break;
    }
    current = 0;
    Console.WriteLine("current result: " + num.ToString());
}
```

Code Review Practice

Others:

- Needs Test cases
- Calculator should have more methods that split up each job
- There are unused libraries that are including that should be deleted
- This works with single number calculations, but not with multiple numbers all on the same line