

Please note that this survey requires at least a 13" monitor and cannot be completed on a mobile phone.

Consent Form

By completing this online questionnaire (i.e., clicking "Next"), you agree to participate in the study.

The study focuses on investigating the type and amount of information needed for debugging program failures. The study is structured in the form of an online questionnaire with 12 questions, which are split into three parts. Each part includes questions about the same piece of code. The entire survey is expected to take around 20-30 minutes of your time.

As a "thank you", we will randomly select 10 participants who finished the study to receive a \$30 Amazon gift card. Moreover, your participation in this study will help the academic and industrial community gain valuable insight into debugging approaches. The results will be fully anonymized and will be reported in an academic paper, which will be openly available to the community.

Participation is completely voluntary and you may withdraw from the study at any time before the final reports are made public.

We do not collect identifying information in this study. However, if you would like to be considered for the award, withdraw your data from the study at a later stage, and/or be notified when the results of the study are available, you should provide your email address as contact information. We will also appreciate it if you provide optional demographic information, which will only be shared in an aggregated form and will not be associated with any individual responses.

Thank you in advance for your time.

Your email address (optional):

Experience and Demographics

1. Which of the following describes you the best? (pick all that apply)

- Software developer or engineer working in industry
- Software tester working in industry
- Researcher working in industry (Research Staff Member, Research Fellow, Research Engineer)
- Researcher working in academia, non-student (Postdoctoral Fellow, Faculty Member)
- PhD student
- Master's student
- Undergraduate student
- Other

2. How many years of programming experience while in school/university do you have? (in any programming language)

- No experience
- Less than 1 year
- At least 1 but less than 3 years
- At least 3 but less than 5 years
- At least 5 but less than 10 years
- More than 10 years

3. How many years of programming experience outside of school/university do you have?

- No experience
- Less than 1 year
- At least 1 but less than 3 years
- At least 3 but less than 5 years
- At least 5 but less than 10 years
- More than 10 years

4. How would you rate your programming skill level?

- Novice: developed a few small programs
- Intermediate: developed a few large programs
- Advanced: developed several large software systems

5. What is your software development area? (e.g., web developer, full stack developer, embedded systems developer, ML data analyst)

6. How do you debug your code? (Pick all that apply)

- I do not debug my code
- Program logging (e.g., print)
- Assertions
- IDE debugger utilities (e.g., breakpoints and stepping)
- Other

7. What is your country of employment or studies?

8. What is your age?

- <25
- 25-34
- 35-44
- 45-54

- 55-64
- >64
- Prefer not to answer

9. What is your gender?

- Female
- Male
- Non-binary person
- Prefer not to answer

Part I

Part 1/3: Comparing code views

In this part of the survey, you will be given three different views, named A, B, and C. Each view contains two versions of a code snippet: the old version (V1), where an assertion passes, and a new version (V2), where the same assertion fails due to changes made in the code.

Each code snippet includes only a subset of statements deemed relevant to the failure. Views A, B, and C differ by the subset of statements their code snippet includes. You will be asked to rank the views based on their:

- **Completeness:** include all essential information needed to explain and debug the failure.
- **Conciseness:** do not include unnecessary information, unneeded to explain and debug the failure.

The goal is to identify views that are most helpful to explain and debug the failure.

Note: Clicking on the picture below opens its larger version.

View A	View B	View C
<pre> 1 public static void main(String[] args) { 2 int hourOffset = 2; 3 int minOffset = -2; 4 assert(4100000L == getOffset(hourOffset, minOffset)); 5 public int getOffset(int hourOffset, int minOffset) { 6 return offset(hourOffset * 60 + minOffset); 7 } 8 }</pre>	<pre> 1 public static void main(String[] args) { 2 int hourOffset = 2; 3 int minOffset = -2; 4 assert(4100000L == getOffset(hourOffset, minOffset)); 5 public int getOffset(int hourOffset, int minOffset) { 6 return offset(hourOffset * 60 + minOffset); 7 } 8 }</pre>	<pre> 1 public static void main(String[] args) { 2 int hourOffset = 2; 3 int minOffset = -2; 4 assert(4100000L == getOffset(hourOffset, minOffset)); 5 public int getOffset(int hourOffset, int minOffset) { 6 return offset(hourOffset * 60 + minOffset); 7 } 8 }</pre>
<small>Notations: Lines that correspond to each other in V1 and V2 are given the same line numbers. Numbered empty lines indicate statements that are either excluded in a particular view or are absent in a code version. Specifically, if a line is annotated by "delete", the statement is deleted in a version. Otherwise, it is excluded from the view. Changes between versions ("Add", "Update", "Delete") are shown as arrows between code snippets. Colored backgrounds highlight the differences between the views.</small>		

1. Please rank views A, B, and C (1 being the best; 3 being the worst). You can drag the view names into the box and then rank them internally. When ranking views, please consider:

- **Completeness:** include all essential information needed to explain and debug the failure.
- **Conciseness:** do not include unnecessary information, unneeded to explain and debug the failure.

Items	Ranking Views
View A	
View B	
View C	

2. Explain your ranking by describing the advantages and disadvantages of each view (given in their sequential order below).

View A: + + - - View B: + + - - View C: + + - -

3. Explain the failure in your own words. Specifically, please describe why changes made in this code resulted in the assertion failure.

--

Part II

Part 2/3: Comparing textual explanations

In this part of the survey, you are given three different **textual** explanations of the failure corresponding to the three code views in the previous part. You will need to rank these explanations based on their:

- **Completeness:** include all essential information needed to explain and debug the failure.
- **Conciseness:** do not include unnecessary information, unneeded to explain and debug the failure.

Note: Clicking on the picture below opens its larger version.

Explanation A	Explanation B	Explanation C
<p>The method <code>getOffset(hoursOffset, minsOffset)</code> calculates the offset in milliseconds given the hours and minutes offset. The assertion in line 4 checks if the method returns the value <code>-8100000</code>. Internally, the method computes the value of the variable <code>offset</code> based on its <code>hoursOffset</code> and <code>minsOffset</code> parameters.</p> <p>However,</p> <ol style="list-style-type: none"> In V1, line 10, the method reassigns <code>minsOffset</code> to the value of <code>hoursOffset*60 - Math.abs(minsOffset)</code>. In V2, the change in line 10, it reassigned <code>minsOffset</code> to the value of <code>hoursOffset*60 - minsOffset</code>, omitting the <code>abs</code> operation. <p>Then, in both versions, it calculates the value of <code>offset</code> as a sum of <code>hoursOffset</code> times its value by <code>secondsPerHour</code>, <code>minsOffset</code> times the multiplication of the duration of one minute by <code>secondsPerMin</code> (lines 13-17). In summary, the change in the initial value assigned to <code>minsOffset</code> in line 10 (i.e., the removal of the call to the <code>Math.abs</code> method for the value of <code>minsOffset</code>), leads to a difference in computing the final value of <code>offset</code> in V1 and V2.</p> <p>This difference causes the <code>offset</code> to differ from <code>-8100000</code> in line 4.</p>	<p>The method <code>getOffset(hoursOffset, minsOffset)</code> calculates the offset in milliseconds given the hours and minutes offset. The assertion in line 4 checks if the method returns the value <code>-8100000</code>. Internally, the method computes the value of the variable <code>offset</code> based on its <code>hoursOffset</code> and <code>minsOffset</code> parameters.</p> <p>However,</p> <ol style="list-style-type: none"> In V1, line 10, the method reassigns <code>minsOffset</code> to the value of <code>hoursOffset*60 - Math.abs(minsOffset)</code>. In V2, the change in line 10, it reassigned <code>minsOffset</code> to the value of <code>hoursOffset*60 - minsOffset</code>, omitting the <code>abs</code> operation. <p>Then, in both versions, it calculates the value of <code>offset</code> as a sum of <code>hoursOffset</code> times its value by <code>secondsPerHour</code>, <code>minsOffset</code> times the multiplication of the duration of one minute by <code>secondsPerMin</code> (lines 13-17). In summary, the change in the initial value assigned to <code>minsOffset</code> in line 10 (i.e., the removal of the call to the <code>Math.abs</code> method for the value of <code>minsOffset</code>), leads to a difference in computing the final value of <code>offset</code> in V1 and V2.</p> <p>This difference causes the <code>offset</code> to differ from <code>-8100000</code> in line 4.</p>	<p>The method <code>getOffset(hoursOffset, minsOffset)</code> calculates the offset in milliseconds given the hours and minutes offset. The assertion in line 4 checks if the method returns the value <code>-8100000</code>. Internally, the method computes the value of the variable <code>offset</code> based on its <code>hoursOffset</code> and <code>minsOffset</code> parameters, which are initialized in lines 2 and 3 to <code>-2</code> and <code>-15</code>, respectively.</p> <p>However,</p> <ol style="list-style-type: none"> In V1, line 10, the method reassigns <code>minsOffset</code> to the value of <code>hoursOffset*60 - Math.abs(minsOffset)</code>. In V2, the change in line 10, it reassigned <code>minsOffset</code> to the value of <code>hoursOffset*60 - minsOffset</code>, omitting the <code>abs</code> operation. <p>Then, in both versions, it calculates the value of <code>offset</code> as a function of the <code>minsOffset</code> value.</p> <p>In summary, the change in the value assigned to <code>minsOffset</code> in line 10 (i.e., the removal of the call to the <code>Math.abs</code> method for the value of <code>minsOffset</code>), leads to a difference in computing the final value of <code>offset</code> in V1 and V2 when the value of <code>minsOffset</code> is <code>-15</code>.</p> <p>This difference causes the function output to differ from <code>-8100000</code> in line 4.</p>

Notations:

Colored backgrounds highlight the differences between the views.

FYI: Views are given below again, for your reference.

View A	View B	View C
<pre> 1 public static void main(String[] args){ 2 int hoursOffset = -2; 3 int minsOffset = -15; 4 assert(-8100000, getOffset(hoursOffset, minsOffset)); 5 } 6 public int getOffset(int hoursOffset, int minsOffset){ 7 if (hoursOffset == 0 && minsOffset == 0) [false] 8 return DateTimeZone.UTC; 9 if (hoursOffset < 0) [true] 10 minsOffset = hoursOffset*60 - Math.abs(minsOffset); Update 11 else 12 minsOffset = hoursOffset*60 + minsOffset; 13 int offset = minsOffset; 14 int secondsPerMin = 60; 15 offset = offset * secondsPerMin; 16 int millisPerSecond = 1000; 17 offset = offset * millisPerSecond; 18 return offset; 19 }</pre>	<pre> 1 public static void main(String[] args){ 2 int hoursOffset = -2; 3 int minsOffset = -15; 4 assert(-8100000, getOffset(hoursOffset, minsOffset)); 5 } 6 public int getOffset(int hoursOffset, int minsOffset){ 7 if (hoursOffset == 0 && minsOffset == 0) [false] 8 return DateTimeZone.UTC; 9 if (hoursOffset < 0) [true] 10 minsOffset = hoursOffset*60 - minsOffset; 11 else 12 minsOffset = hoursOffset*60 + minsOffset; 13 int offset = minsOffset; 14 int secondsPerMin = 60; 15 offset = offset * secondsPerMin; 16 int millisPerSecond = 1000; 17 offset = offset * millisPerSecond; 18 return offset; 19 }</pre>	<pre> 1 public static void main(String[] args){ 2 int hoursOffset = -2; 3 int minsOffset = -15; 4 assert(-8100000, getOffset(hoursOffset, minsOffset)); 5 } 6 public int getOffset(int hoursOffset, int minsOffset){ 7 if (hoursOffset == 0 && minsOffset == 0) [false] 8 return DateTimeZone.UTC; 9 if (hoursOffset < 0) [true] 10 minsOffset = hoursOffset*60 - Math.abs(minsOffset); 11 else 12 minsOffset = hoursOffset*60 + minsOffset; 13 int offset = minsOffset; 14 int secondsPerMin = 60; 15 offset = offset * secondsPerMin; 16 int millisPerSecond = 1000; 17 offset = offset * millisPerSecond; 18 return offset; 19 }</pre>

4. Please focus on the **textual explanation** now (upper part of the picture). Please rank explanations A, B, and C (1 being the best; 3 being the worst). You can drag the explanation names into the box and then rank them internally.

- Items
- Explanation A
 - Explanation B
 - Explanation C

Ranking Explanations

5. Explain your ranking by describing the advantages and disadvantages of each explanation (given in their sequential order below).

Explanation A:	Explanation B:	Explanation C:
----------------	----------------	----------------

6. Which of the following do you prefer to see when understanding and debugging the failure?

- Code views
- Textual explanations
- Both

7. Explain your selection.

Part III

Part 3/3: Analyzing complete code snippets

In this part of the survey, you are given the complete code snippets of versions V1 and V2. You will be asked whether seeing the complete code helps your understanding of the failure.

Notations: The code is represented in a way similar to the code in Part 1. For simplicity, each “if” statement (e.g., in line 7) is annotated with a label showing whether the “if” condition is evaluated to *true* or *false*. Gray lines indicate statements that are not executed because they are encapsulated by an ‘if’ statement that evaluates to *false*. Like in Part 1, Changes between versions (“Add”, “Update”, “Delete”) are shown on arrows between the code snippets.

8. Please select statements you deem important for understanding and debugging the failure (in addition to the changed statements that are clearly important and, thus, already pre-selected below). Click on a statement to select it and click again to deselect it.

V1	V2
<pre> 1 public static void main(String[] args){ 2 int hoursOffset = -2; 3 int minsOffset = -15; 4 assert(-8100000, getOffset(hoursOffset, minsOffset)); 5 } 6 public int getOffset(int hoursOffset, int minsOffset){ 7 if (hoursOffset == 0 && minsOffset == 0) [false] 8 return DateTimeZone.UTC; 9 if (hoursOffset < 0) [true] 10 minsOffset = hoursOffset*60 - Math.abs(minsOffset); Update 11 else 12 minsOffset = hoursOffset*60 + minsOffset; 13 int offset = minsOffset; 14 int secondsPerMin = 60; 15 offset = offset * secondsPerMin; 16 int millisPerSecond = 1000; 17 offset = offset * millisPerSecond; 18 return offset; 19 }</pre>	<pre> 1 public static void main(String[] args){ 2 int hoursOffset = -2; 3 int minsOffset = -15; 4 assert(-8100000, getOffset(hoursOffset, minsOffset)); 5 } 6 public int getOffset(int hoursOffset, int minsOffset){ 7 if (hoursOffset == 0 && minsOffset == 0) [false] 8 return DateTimeZone.UTC; 9 if (hoursOffset < 0) [true] 10 minsOffset = hoursOffset*60 - minsOffset; 11 else 12 minsOffset = hoursOffset*60 + minsOffset; 13 int offset = minsOffset; 14 int secondsPerMin = 60; 15 offset = offset * secondsPerMin; 16 int millisPerSecond = 1000; 17 offset = offset * millisPerSecond; 18 return offset; 19 }</pre>

9. Explain your selection.

10. Has seeing the complete code snippet changed your understanding of the failure and why? Would you now

augment the explanation you gave in Part 1, Question 3? (you can navigate to your explanation by pressing the back button twice)



11. Please list any suggestions for how the views and textual explanations you liked the most can be improved even further.



12. Do you have any other comments related to this survey?

