Simplifying Access Modifiers Terminology

Taking into account the unique terminology used in various programming languages, we now present the formal definitions of the access modifiers that will be used in the questionnaire survey:

1. **Public:** Globally and externally visible, accessible from *any* part of the program.
2. **Non-Public:** Not globally visible, *limited* accessibility.
   - **Protected / Semi-visible:** Partially visible (to other Module / Package / Subclass / Class), *but* not globally.
   - **Private:** Visible *only* within the class / module / file; i.e., *inaccessible from outside* of the class / module / file.

Access modifier prefixes in certain programming languages (e.g. Python) are primarily based on **convention (C)** rather than strict enforcement by the language. This means that the prefixes used to indicate access modifiers are not mandated by the language itself.
The table provided below presents a comprehensive overview of the three access modifiers in different programming languages:

Programming Language	**Public**	**Non-Public**	
		Protected / Semi-visible	**Private**
C#	`public method/variable`	`protected / internal / protected internal / private protected method/variable`	`private method/variable`
C++	`public method/variable`	`protected method/variable`	`private method/variable`
Go	`Method/Variable (uppercase letter)`	`method/variable (lowercase letter)`	**`Not available`**
Java	`public method/variable`	`protected / package-private method/variable`	`private method/variable`
Javascript **(C)**	`[method/variable]`	`_[method/variable]`	`#[method/variable]`
Kotlin	`public method/variable`	`protected / internal method/variable`	`private method/variable`
Python **(C)**	`[method/variable]`	`_[method/variable]`	`__[method/variable]`
PHP	`public method/variable`	`protected method/variable`	`private method/variable`
Ruby	`public def method/variable`	`protected def method/variable`	`private def method/variable`
Rust	`pub method/variable`	`pub(crate) / pub(super) / trait method/variable`	`[method/variable]`
Typescript	`export / public method/variable`	`protected method/variable`	`private method/variable`

Industry Survey

* Indicates required question

Survey Questions

Please refer to the Simplifying Access Modifiers Terminology section in the Participant Information Sheet to get more information about the different access modifiers used in different programming languages covered by our questionnaire.

1. What is your main programming language? *

   Mark only one oval.

   - ◯ C#
   - ◯ C++
   - ◯ Go
   - ◯ Java
   - ◯ JavaScript
   - ◯ Kotlin
   - ◯ PHP
   - ◯ Python
   - ◯ Ruby
   - ◯ Rust
   - ◯ Typescript
   - ◯ Other: ______________________

2. If your main language **does not have access modifiers** (e.g., public, private, protected, etc.), do you follow any *conventions* to denote visibility of methods and instance variables? *

   (For example, some Python programmers elect to prefix private methods with an underscore.)

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ My main programming language has access modifiers
   - ◯ Other: ______________________

3. If you answered "yes", please tell us what conventions you follow.

   If you answered "no", please tell us your reasons as to why you do not follow any conventions.

   ____
   ____
   ____
   ____
   ____

Based on your main programming language chosen previously, we will ask specific questions about how do you do testing for public or non-public (e.g., private, protected) methods, following the syntax and conventions of your main programming language that you stated earlier in the questionnaire.

4. Does the code that you are usually testing involve **non-public** methods? *

   Mark only one oval.

   - ◯ Yes
   - ◯ No
   - ◯ Not sure

5. To what extent do you agree with the following statement? *

"In general, developers should write unit tests that only invoke *public methods*, avoiding direct calls to *non-public methods*."

Mark only one oval.

- Strongly disagree
- Disagree
- Agree
- Strongly agree
- Not sure

6. How often do you write tests that **directly invoke** non-public methods? *

Mark only one oval.

- Never
- Rarely
- Sometimes
- Often
- Always

7. How do you go about testing **non-public** methods? *

Mark only one oval per row.

	Not a feature in my language	Never	Rarely	Sometimes	Often	Mostly	Not sure
Via public method that invokes the non-public method							
Directly invoking the non-public method							
Using Reflection / Mocks							
Adding test code (e.g., print statements or assertions) in production code							
Temporarily changing non-public methods to public							
Changing the visibility of the method to public permanently							

8. Are there any other approaches or strategies that you use to test **non-public** methods?

9. Do you take a different approach for testing **different levels of visibility of non-public methods?** (For example, is your treatment of *private* methods different compared to *protected* methods?) *

Mark only one oval.

- Yes
- No

10. If you answered "yes", please tell us how and why.

11. Are there any guidelines, best practices, or specific rules you follow when testing **non-public** methods?

If so, please tell us about them here.

12. To what extent do you value the following aspects when writing unit tests? *

Mark only one oval per row.

	Not important	Somewhat important	Important	Very Important	No opinion
Coverage of production code	◯	◯	◯	◯	◯
Capturing the behaviour of the production code (through assertions)	◯	◯	◯	◯	◯
Ease of debugging following production code failures	◯	◯	◯	◯	◯
Robustness following refactoring of production code	◯	◯	◯	◯	◯
Sensitivity to behavioural changes of production code	◯	◯	◯	◯	◯
Realistic exercising of the unit by the test in a similar way to its usage in production	◯	◯	◯	◯	◯
Tests that instill confidence in the production code	◯	◯	◯	◯	◯
Writing concise unit tests to test the production code	◯	◯	◯	◯	◯

13. To what extent do you agree with the following statement? *

"Testing non-public methods leads to more tests failing spuriously when modifications are made to those methods."

Mark only one oval.

◯ Strongly Disagree

◯ Disagree

◯ Agree

◯ Strongly Agree

◯ Not sure

14. If you have anything else to say about your thoughts and/or processes when writing unit tests with respect to testing public and non-public methods, please let us know here:

15. Are there features you'd like to see in unit testing or mocking tools in the future to better accommodate the testing of non-public methods?

16. How many years of experience do you have in software development? *

Mark only one oval.

- 0–2
- 2–5
- 5–10
- 10–15
- More than 15

17. How many years of experience do you have in **writing unit tests**? *

Mark only one oval.

- 0–2
- 2–5
- 5–10
- 10–15
- More than 15

18. In which industry are you currently working in? *

Tick all that apply.

- Information Technology
- University / Education
- Electronics
- Enterprise / Business Software
- Fintech (Finance)
- Hospitality / Leisure Industry
- Healthcare
- Mass Media / Entertainment
- Public Sector / Government / Defense
- Retail Industry
- Sports Industry
- Transportation / Automotive
- Other: ______________________

19. Do you employ any of the following software development methodologies and/or programming practices? Please tick all that apply.

Tick all that apply.

- ☐ Test-driven development
- ☐ Behaviour-driven development
- ☐ Acceptance test-driven development
- ☐ Test-last development
- ☐ Feature-driven development
- ☐ Agile methodology, e.g. Scrum
- ☐ Waterfall method
- ☐ Other: ____________________