

LiquidJava - Hands-On Study

This research aims to evaluate the usability and understanding of LiquidJava, an extension to the Java programming language, with the introduction of refinement types as a means for static program verification.

[REDACTED]

All responses to this questionnaire will be anonymous, and the treatment of the information obtained will be used only and exclusively in the context of the referred thesis.

Find the Bug - Plain Java

In this section, you will be asked to identify the bug present in the code. You have 7 minutes to finish each exercise. You can move to the next question without completing the current one if you don't think you can find the bug.

Start by cloning or downloading the repository:

[REDACTED]

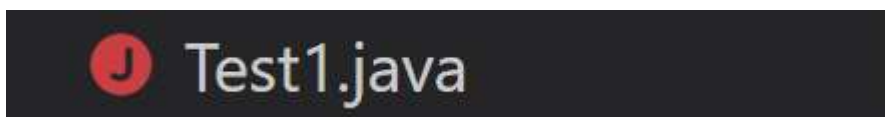
Please open the folder part1-plainJava/together# on Visual Studio Code.

The # corresponds to the number given by the interviewer.

To open the code:

Option 1 - Inside VSCode go to File -> Open Folder -> choose correct folder

Option 2 - Open the terminal, travel to the folder and type "code ."

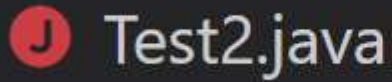


Open "Test1.java" and try to find the error.

1. Insert the line that contains the error.

Please insert the line of code (not just the number).

2. What do you suggest to fix it?



Open "Test2.java" and try to find the error.

3. Insert the line that contains the bug.

Please insert the line of code (not just the number).

4. What do you suggest to fix it?

Understand the
Refinements without
prior explanation

This section will show you multiple LiquidJava code snippets and ask you to write a correct and incorrect use of the code. Read the questions carefully.

Variable Refinement

```
@Refinement("-25 <= x && x <= 45")  
int x;
```

5. Implement, in Java, a CORRECT assignment for x.

6. Implement, in Java, an INCORRECT assignment for x.

Refinement in Function

```

@Refinement("_ >= 0")
public static double function1(
    @Refinement("a >= 0") double a,
    @Refinement("b >= a") double b){
    return (a + b)/2;
}

```

7. Implement a CORRECT invocation of function1.

8. Implement an INCORRECT invocation of function1.

Class Refinements

```

@StateSet({"sX", "sY", "sZ"})
public class MyObj {

    @StateRefinement(to="sY(this)")
    public MyObj() {}

    @StateRefinement(from="sX(this)", to="sZ(this)")
    public void pay(int account) { }

    @StateRefinement(from="sY(this)", to="sX(this)")
    public void select(int number) {}

    @StateRefinement(from="sX(this)", to="sY(this)")
    @StateRefinement(from="sZ(this)", to="sY(this)")
    public void show() { }
}

```

9. Create a MyObj object and implement a CORRECT sequence of invocations on the object.

Please use at least 3 invocations.

10. Create a MyObj object and implement an INCORRECT sequence of invocations on the object.

Please use at least 3 invocations.

LiquidJava Overview

Watch the video.

Go to [REDACTED]

Follow the instructions to install the LiquidJava Extension for VSCode.

You will need to have VSCode and Language Support for Java(TM) by Red Hat plugin installed.

LiquidJava Overview



11. How easy was it to install the extension?



1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Find the Bug -
Liquid Java

In this section, you will be asked to identify the bug present in LiquidJava code snippets.

Please open the folder part3-liquidJava/together# on Visual Studio Code

The # corresponds to the number given by the interviewer.

To open the code:

Option 1 - Inside VSCode go to File -> Open Folder -> choose correct folder

Option 2 - Open the terminal, travel to the folder and type "code ."



Open "Test1.java" and try to find the error.

12. Insert the line that contains the bug.

Please insert the line of code (not just the number).

13. What do you suggest to fix it?



Open "Test2.java" and try to find the error.

14. Insert the line that contains the bug.

Please insert the line's code.

15. What do you suggest to fix it?

Annotate a Java Program with Refinement Types

Open the folder part4/exercises-add-refinements on VSCode.
In this section, you will be asked to annotate a Java program
with Refinement Types.

Some examples of the annotations can be found in

[REDACTED]

For each exercise:

- An implementation in Java is already provided. If you find errors in the implementation, fix them.
- A correct and incorrect use is given to test the refinement.
- After completing the exercise, comment the error and proceed to the next exercise.

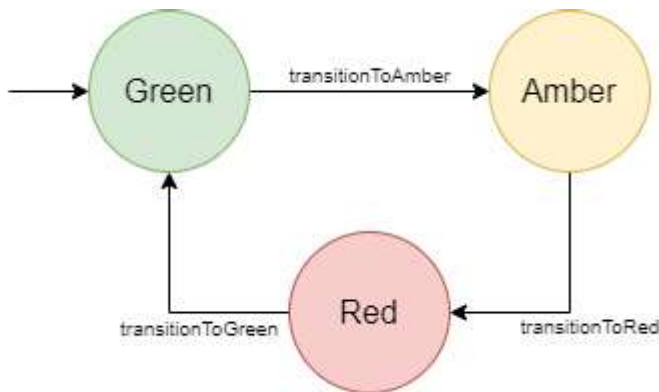
16. Add refinements to currentMonth variable in Variable.java file, according to the comment above the variable.

Copy the Variable.java code to this box.

17. Add refinements to the method `inRange(...)` in `Method.java`, according to the javadoc.

Copy the `Method.java` code to this box.

Annotate the class `TrafficLight`, that uses `rgb` values (between 0 and 255) to define the color of the light, and follows the protocol defined by the following image



18. Add the annotations to `TrafficLight.java`. The `TestTLWrong.java` must have an incorrect behaviour and `TestTLCorrect.java` must have a correct behaviour.

Copy the `TrafficLight.java` code to this box.

19. How easy was it to add the Refinement Types annotations?

[Redacted]

	1	2	3	4	5	
Very Difficult	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very Easy

Final Overview

20. What did you enjoy the most while using LiquidJava?

21. What did you dislike the most while using LiquidJava?

22. Would you use LiquidJava in your projects?

[Redacted]

☐ Yes

☐ No

23. If you have any other comments or suggestions leave them here.
