# **LiquidJava:** Improving the Usability of Liquid Types for Reliable Software

Catarina Gamboa[1,2], Alcides Fonseca[2], Jonathan Aldrich[1]

[1] S3D, School of Computer Science, Carnegie Mellon University
[2] LASIGE, Faculty of Sciences of the University of Lisbon

## 1 MOTIVATION

Software errors and vulnerabilities are usually introduced by developers while writing code and can have dire consequences.
Strong type systems are one of the most used strategies to add guarantees about the code behavior but they are not very expressive.

```
int p = "string"; //Error
```

## 2 APPROACH

Liquid types extend a language with predicates allowing more domain-specific information. However they are yet to become widely used.

### Liquid Types

```
@Refinement("0 <= p && p <= 100")
int p;
p = 20; //Correct
p = 200; //Error
```

The predicates are automatically verified before executing the code.

### User-Centered Design

## 3 LIQUID JAVA

Refinements designed with developer's feedback

IDE integration

Usability study

- LiquidJava helps finding more bugs
- Refinements are easy to interpret
- Useful for protocols and lesser known-classes;

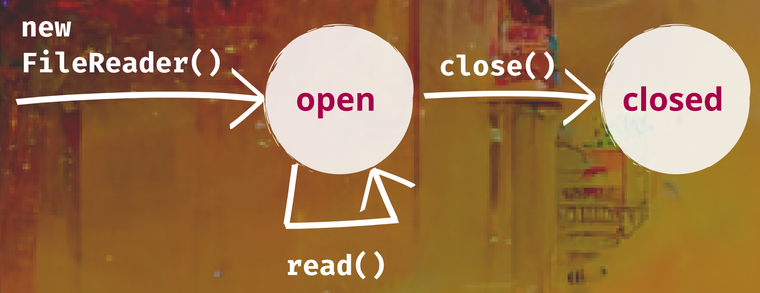
4 TASKS

30

- Easy to annotate programs;
- Partial refinements might mislead developers

Liquid Java enables refinements of

- variables
- methods pre- and post- conditions
- classes using state machines and ghost properties




```
@StateSet("open", "closed")
class FileReader{
  @StateRefinement(to="open(this)")
  public FileReader(){}

  @StateRefinement(from="open(this)",
                   to="open(this)")
  public void read(){}

  @StateRefinement(from="open(this)",
                   to="close(this)")
  public void close(){}
}
FileReader f = new FileReader();
f.read();
f.close();
f.read();//Error
```

Visit website for more examples

## 4 WORK PLAN

### Development

Formalize Type System

Enhance verification features (e.g., aliasing, loop verification)

### Testing & Repair

Use liquid types to automatically cover edge cases in testing suites

Generate patches with the help of liquid types

### Debugging

Improve debugging experience using task analysis to understand how developers debug using liquid types and improve design


IDE integration

**Acknowlegments**

This work was supported by Fundação para a Ciência e Tecnologia (FCT) in the LASIGE Research Unit under the ref. (UIDB/00408/2020 and UIDP/00408/2020), and the CMU-Portugal Dual Degree PhD Program given to Catarina Gamboa, and by the CMU--Portugal project CAMELOT, (LISBOA-01-0247-FEDER-045915), and the RAP project under the reference (EXPL/CCI-COM/1306/2021),