Verifying Java Programs with VeriFast

A Simple Introduction to VeriFast and vfide

Aiden Boyce
CS 810-E
13 May 2025

Video Tutorial: https://youtu.be/01eJ5SAMY4g
Tutorial Files: https://github.com/aiden-boyce/verifast-java-tutorial

How to Download VeriFast

Step 1) Download the latest release on GitHub for your OS.

- https://github.com/verifast/verifast/releases

Step 2) Extract the compressed file.

Step 3) Place the uncompressed directory wherever you'd like.

Step 4) Done.

Verifying Programs with VeriFast's IDE: vfide

Step 1) Open your `VeriFast` folder.

Step 2) Navigate to the `./bin` directory.

Step 3) Run `vfide.exe`

Step 4) Download the tutorial files and follow the next steps below.

- https://github.com/aiden-boyce/verifast-java-tutorial

Step 5) Open the directory with your program and select the program.

Step 6) Verify the program by clicking `Verify` (▷) or verify a function by clicking `Verify Function` (⚙).

Step 7) Done.

Go to the next page to learn about the basic annotations supported by VeriFast.

What are the VeriFast Annotations?

Annotations

- A special comment that is used to provide formal specifications to VeriFast.
- They describe what the program should do, so VeriFast can prove its correctness.
- Single-line Annotation:
    - `//@ ...`
- Multi-line Annotation:
    - `/*@ ... @*/`

Method Contracts

- Defines what must be true before and after a method is executed.
    - Preconditions:
        - The resources or conditions that are expected to be true before the method is executed.
        - `//@ requires true;`
    - Postconditions:
        - What resources are returned after the method is executed.
        - `//@ ensures x <= max;`

Assertions

- Assert what must hold true at a specific point
- `//@ assert x <= max;`

Predicates

- Abstraction used to represent permissions, objects, and logical conditions.
- Can be reused
- `//@ predicate valid_act(int b) = this.balance |-> b;`
- Can be opened and closed to modify an object.
    - `//@ open valid_act(b);`
    - `//@ close valid_act(b + amount);`

Invariants

- Defines conditions that must hold true throughout the entire loop: before the loop is executed, during every iteration, and after it ends.
- `//@ invariant 0 <= i && i <= n;`