Wesley Cox - coxw2
Grant Hughes - gyhughes
Joe Santino - jsantino

Use Cases

Use case 1) Developer needs to fix some non-obvious out-of-bounds exception

Actors

Developer, Tester

Preconditions

Error exists in code/logic involving arrays/lists

Triggers (what starts the use case)

Exception gets raised in some developer's unit test, error is non-obvious

Minimal/success guarantees (end condition)

If code is properly annotated, no index-out-bounds exceptions will be thrown after all warnings are dealt with.

List of steps to the success scenario

1. Look through method(s) and related object fields that threw the exception and properly annotate types as shown in the user manual.
2. Run our type checker
3. Fix source code such that warnings no longer appear

Failure end conditions

- Code is annotated incorrectly
- Warnings are suppressed incorrectly
- Source code uses unsupported features (reflection?)

Extensions/variations of the scenario

Tester runs into error and notifies developer

Use Case 2) Testing found silent behavior error, need to eliminate sources of error.

Actors

Developer, Tester

Preconditions

Error exists in code/logic involving arrays/lists

Triggers (what starts the use case)

Program misbehaves, no error is caught by the program itself

Minimal/success guarantees (end condition)

If code is properly annotated, all indexes will only be used to access the arrays/lists they are meant to access after all warnings are dealt with.

List of steps to the success scenario

1. Identify the pieces of code that are most likely the cause of the error.
2. If these pieces of code deal with arrays/lists, properly annotate types as shown in the user manual.
3. Run our type checker
4. Fix source code such that warnings no longer appear

Failure end conditions

See failure end conditions of use case 1

Extensions/variations of the scenario

Another module fails, error source unclear

Use Case 3) Project code needs to be reliable

Actors

Developer, Manager

Preconditions

A project specifies that some code absolutely must be accurate and reliable

Triggers (what starts the use case)

The need to ensure reliable code arises in some part of the project

Minimal/success guarantees (end condition)

If code is properly annotated, all indexes will only be used to access the arrays/lists they are meant to access and no index-out-of-bound errors will be thrown after all warnings are dealt with.

List of steps to the success scenario

1. As code is being written in these sensitive sections, properly annotate relevant types.
2. Upon compilation, run our type checker
3. Fix source code such that warnings no longer appear

Failure end conditions

See failure end conditions of use case 1

Extensions/variations of the scenario

A developer simply wants to have extra confidence in his code quality