Background

- In Object-Oriented Languages, an immutable object is one whose state cannot be changed once constructed.
- The state of an object O is a transitive closure of other objects reachable from O through fields.
- Immutable objects make designing applications (sequential, parallel, distributed) simpler; the programmers need not reason about side effects.
- Manual transformation of a client code to use a mutable class in an immutable way is time-consuming, tedious, and error-prone.
- Refactoring a client code to use a mutable class in an immutable way requires two steps.

Motivating Example

Goal

- Design a tool that detects whether it’s safe to adapt a client code to use a mutable class in an immutable way before transformation.
- If it’s safe, the tool should do the refactoring.
- “Safe” means that the behavior of the client code is preserved.

Fundamental Questions/Challenges

- What’s considered preserving the behavior of the client code?
- When is it unsafe to adapt a client code to use a mutable class in an immutable fashion?
- Determine what static analysis techniques to use to capture the unsafe spots in the client code.
- Determine how the tool should present the unsafe spots to the programmer.

Related Work/Interaction with Other Projects