Questions

Bookmark this page

arguing against rep exposure

4/6 points (graded)

Consider the following ADT:

```
// Mutable type representing Twitter users' followers.
public class FollowGraph {
    private final Map<String,Set<String>> followersOf;

    // Rep invariant:
    //    all Strings in followersOf are Twitter usernames
    //          (i.e., nonempty strings of letters, digits, underscores)
    //    no user follows themselves, i.e. x is not in followersOf.get(x)
    // Abstraction function:
    //    represents the follower graph where Twitter user x is followed by user y
    //       if and only if followersOf.get(x).contains(y)
    // Safety from rep exposure:
    //    All fields are private, and ..???..

    // Operations (specs and method bodies omitted to save space)
    public FollowGraph() { ... }
    public void addFollower(String user, String follower) { ... }
    public void removeFollower(String user, String follower) { ... }
    public Set<String> getFollowers(String user) { ... }
}
```

For each statement below: assuming the omitted method bodies are consistent with the statement, can we use the statement in place of `..???..` to make a persuasive safety-from-rep-exposure comment?

1. "Strings are immutable."

No — **Answer:** No

2. "`followersOf` is a mutable `Map` containing mutable `Set` objects, but `getFollowers()` makes a defensive copy of the `Set` it returns, and all other parameters and return values are immutable `String` or `void`."

Yes — ✔ **Answer:** Yes

3. "This class is mutable, so rep exposure isn't an issue."

No — ✔ **Answer:** No

4. "`followersOf` is a mutable Map, but it is never passed or returned from an operation."

No — **Answer:** No

5. "`FollowGraph()` does not expose the rep; `addFollower()` does not expose the rep; `removeFollower()` does not expose the rep; `getFollowers()` does not expose the rep."

No — ✔ **Answer:** No

6. "`String` is immutable, and the `Set` objects in the rep are made immutable by unmodifiable wrappers. The `Map` type is mutable, but that type is never passed or returned from an operation."

Yes ✔ **Answer:** Yes

Explanation

1. This is true, but insufficient, because the rep also contains mutable `Map` and `Set` objects.
2. This is a good argument. It considers each element of the rep (each private field, including all objects in the data structure that the field points to), and how each operation might affect it.
3. Immutability is not the only invariant that can be threatened by rep exposure. `FollowsGraph` has a rep invariant that can be threatened if a `Set` of followers is inadvertently shared with a client.
4. The `Map` is not the only mutable type in the rep.
5. Proof by repeated assertion is not an argument.
6. This is a good argument. It considers all the types in the rep, asks whether they are immutable or not, and whether there is a static guarantee (for `Map` and `String`) or a dynamic check (for `Set`) that protects them from rep exposure. One could ask which approach is more error-prone -- defensive copying, or making sure the sets stay unmodifiable internally -- but the safety argument is nevertheless sound.

Submit

Show Answer

Previous Next