Basing equals on Identity, Not Everything

Compare every field in equals and an edited defect stops being recognized as the same defect. Base it on what defines identity -- usually a stable id -- not on what changes.

Automation Foundations · ashishautomationlabs.store

Writing `equals` raises an easy mistake: comparing more fields than you should. A `Defect` class overrides `equals` to compare every field -- `id`, `severity`, `status`, and `description`.

That looks thorough. It's actually a bug waiting to happen. Two defects are the same defect when their `id` matches. Comparing `description` too means the same defect with an edited description now counts as a _different_ defect -- but a `Defect` object with a stale `status` isn't really equal to the fresh one either, even with the same id.

The resolution is to separate two different questions. **"Is this the same defect?"** is about identity -- the id. **"Has this defect changed?"** is a different check you can write separately, if you need it at all. Bundle both into `equals` and every list and set will treat an edited defect as a brand-new one.

The verdict: base `equals` on the fields that define identity -- usually a stable id -- not on fields that change. Otherwise a defect that gets updated will look like a different defect to every collection that holds it, and your dedup logic quietly stops working.

Where this bites in real test code

This is exactly the trap a defect-tracking integration falls into. A test suite pulls open defects from a tracker, keeps them in a `Set<Defect>` to avoid filing duplicates, and then someone updates a defect's status from "Open" to "In Progress" partway through a run. If `equals` included `status`, the suite now sees that defect as a new one -- not because anything about _which defect it is_ changed, but because a field that was never meant to define identity did.

The same question applies to every type you'll write from here on: a `TestResult`, a `User`, a `Product`. Before you write `equals`, ask what makes two of these the same thing, not what happens to differ between two instances that clearly represent the same real-world record.