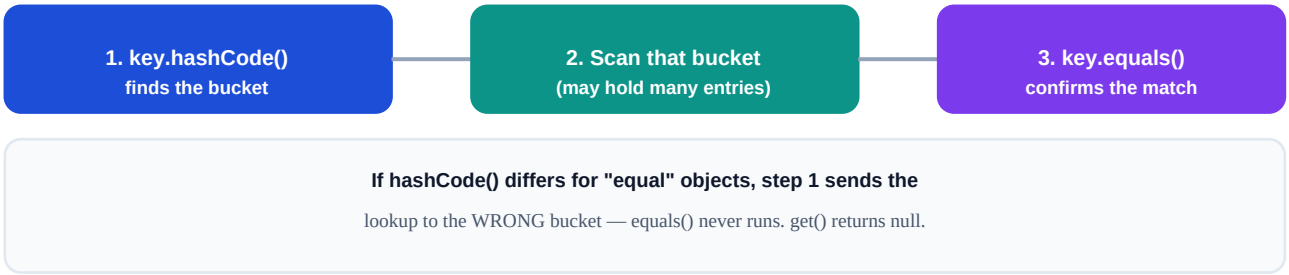
HashMap Key Contract Debugging Checklist

Diagnose silent lookup failures, duplicate-detection bugs, and null-value migration issues.

1. How a Lookup Resolves



If hashCode() differs for "equal" objects, step 1 sends the

lookup to the WRONG bucket — equals() never runs. get() returns null.

2. Null Support by Implementation

Implementation	Null Keys?	Null Values?
HashMap	One allowed	Allowed
ConcurrentHashMap	Not allowed	Not allowed
Hashtable	Not allowed	Not allowed

3. The Fix Pattern

```
@Override
public boolean equals(Object o) {
  if (this == o) return true;
  if (!(o instanceof ClaimKey)) return false;
  return claimId.equals(((ClaimKey) o).claimId);
}

@Override
public int hashCode() {
  return claimId.hashCode();
}
```

Both methods must be overridden together — overriding only one reintroduces the bug.

4. Triage Checklist

- ■ get() returns null for a key you're sure was put in? Check the key class for missing/mismatched equals() and hashCode() overrides.
- ■ Duplicate detection silently failing? The matching key class likely uses default identity-based hashCode().
- ■ NullPointerException right after switching HashMap → ConcurrentHashMap? Check for null values or keys the old map allowed.

■ Key fields mutated after insertion? A changed hashCode() makes the entry unreachable at its original bucket.

■ Relying on iteration order in code or a test assertion? HashMap doesn't guarantee one — switch to LinkedHashMap if order matters.

■ Multiple threads writing to the same HashMap? Replace it with ConcurrentHashMap, not a manually synchronized wrapper.

Rule of thumb: equals() and hashCode() are a pair, never override one without the other. Oracle's Object class contract requires equal objects to share a hash code — break that, and lookups fail silently instead of throwing an error.