Questions

Bookmark this page

risky #2

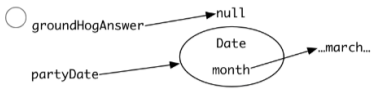
1/1 point (graded)

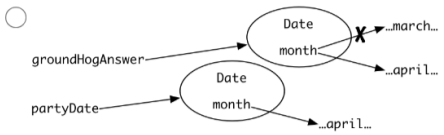
```
/** @return the first day of spring this year */
public static Date startOfSpring() {
    if (groundhogAnswer == null) groundhogAnswer = askGroundhog();
    return groundhogAnswer;
}
private static Date groundhogAnswer = null;
```

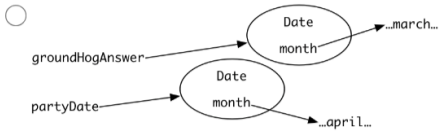
```
// somewhere else in the code...
public static void partyPlanning() {
    // let's have a party one month after spring starts!
    Date partyDate = startOfSpring();
    partyDate.setMonth(partyDate.getMonth() + 1);
    // ... uh-oh. what just happened?
}
```

We don't know how `Date` stores the month, so we'll represent that with the abstract values `...march...` and `...april...` in an imagined `month` field of `Date`.

Which of these snapshot diagrams shows the bug?







✔

Explanation

The problem arises because the cached `groundhogAnswer` points to the same mutable `Date` that is updated from March to April.

Submit | Show Answer

Answers are displayed within the problem

understanding risky example #2

1/1 point (graded)

`partyPlanning` has unwittingly changed the start of spring because `partyDate` and `groundhogAnswer` happen to point to the same mutable `Date` object.

Worse, this bug will probably *not* be discovered in `partyPlanning()` or `startOfSpring()` right away. Instead, it will be some innocent piece of code that subsequently calls `startOfSpring()`, gets the wrong date back, and goes on to compute its own wrong answer.

Globalization?
Is one problem here that `groundhogAnswer` is a global variable?

- ( ) It's global, that's OK
- ( ) It's global, that's terrible
- (•) It's not global ✔

Submit

Show Answer

✔ Correct (1/1 point)

arithmonthmatic

1/1 point (graded)

A second bug
The code has another potential bug in how it adds to the month.

Take a look at the Java API documentation for `Date.setMonth`.

For what result of `partyDate.getMonth()` could there be a problem?

11 ✔ **Answer:** 11

Explanation
According to the specs of `getMonth` and `setMonth`, 0 represents January and 11 represents December.
So if `partyDate` is in December, `getMonth` returns 11, and the code makes an invalid call to `setMonth(12)`.

Submit

Show Answer

Answers are displayed within the problem

NoSuchMonthException

0/1 point (graded)

The documentation for `Date.setMonth` says: `month – the month value between 0-11`.

Based on that statement and what you've read so far...

Which of the following *might* happen when this month-adding bug is triggered? Check all that apply.

- [x] The method call will do nothing ✔
- [x] The method call will actually do the thing we wanted ✔
- [x] The method call will cause the Date object to become invalid and report incorrect values ✔
- [ ] The method call will throw a checked exception
- [x] The method call will throw an unchecked exception ✔

- [x] The method call will set our computer clock to 9/9/99 ✔
- [ ] The method call will cause other Date objects to become invalid ✔
- [ ] The method call will never return ✔

Explanation
If that restriction on `month` is a precondition, when we call `setMonth(12)`, we have violated the precondition. The function is free to do *whatever it wants*... except that the method declares no checked exceptions, so the compiler will not allow that option.
All the other options are possible even if several of them are very unlikely.

Submit

Show Answer

Answers are displayed within the problem

SuchTerribleSpecificationsException

1/1 point (graded)

Elsewhere in the documentation for `Date`, it says: "arguments given to methods [...] need not fall within the indicated ranges; for example, a date may be specified as January 32 and is interpreted as meaning February 1".

What looks like a precondition... isn't!

Which of these is an argument against this feature of `Date`?

- ( ) Don't repeat yourself (DRY)
- (•) Fail fast
- ( ) Groundhog algorithm
- ( ) Exceptions for special results
- ( ) Preconditions restrict the client

✔

Explanation
Some clients might appreciate the flexibility of this approach, but other clients will have bugs that `Date` makes it harder to find. If the spec says values need to be in particular ranges, it would be better to fail fast when they're not.

Submit

Show Answer

Answers are displayed within the problem

Previous Next