Testing on the Toilet Presents... Healthy Code on the Commode

Eliminate YAGNI Smells

The majority of software development costs are due to maintenance. **One way to reduce maintenance costs is to *implement something only when you actually need it,* a.k.a. the *"You Aren't Gonna Need It"* (YAGNI) design principle.** How do you spot unnecessary code? Follow your nose!

A *code smell* is a code pattern that usually indicates a design flaw. For example, creating a base class or interface with only one subclass may indicate a speculation that more subclasses will be needed in the future. Instead, **practice *incremental development and design***: don't add the second subclass until it is actually needed.

The following C++ code has many YAGNI smells:

```
class Mammal { ...
  virtual Status Sleep(bool hibernate) = 0;
};
class Human : public Mammal { ...
  virtual Status Sleep(bool hibernate) {
    age += hibernate ? kSevenMonths : kSevenHours;
    return OK;
  }
};
```

Maintainers are burdened with understanding, documenting, and testing both classes when only one is really needed. Code must handle the case when `hibernate` is true, even though all callers pass `false`, as well as the case when `Sleep` returns an error, even though that never happens. This results in unnecessary code that never executes. **Eliminating those smells simplifies the code**:

```
class Human { ...
  void Sleep() { age += kSevenHours; }
};
```

Here are some other YAGNI smells:

- Code that has never been executed other than by tests (a.k.a. code that is *dead on arrival*)
- Classes designed to be subclassed (have virtual methods and/or protected members) that are not actually subclassed
- Public or protected methods or fields that could be private
- Parameters, variables, or flags that always have the same value

Thankfully, **YAGNI smells, and code smells in general, are often easy to spot by looking for simple patterns and are easy to eliminate using simple refactorings**.

Are you thinking of adding code that won't be used today? Trust me, *you aren't gonna need it!*

More information, discussion, and archives:
testing.googleblog.com