The Craftsman: 4

Robert C. Martin
12 July 2002

Dear Diary,

Last night I stared out the window for hours watching the stars drift through the spectrum. I felt conflicted about the work I did with Jerry yesterday. I learned a lot from working with Jerry on the prime generator, but I don't think I impressed him very much. And, frankly, I wasn't all that impressed with him. he spent a lot of time polishing a piece of code that worked just fine.

Today Jerry came to me with a new exercise. He asked me to write a program that calculates the prime factors of an integer. He said he'd work with me from the start. So the two of us sat down and began to program.

I was pretty sure I knew how to do this. We had written the prime generator yesterday. Finding prime factors is just a matter of walking through a list of primes and seeing if any are factors of the given integer. So I grabbed the keyboard and began to write code. After about half an hour of writing and testing I had produced the following.

```
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class PrimeFactorizer {
  public static void main(String[] args) {
    int[] factors = findFactors(Integer.parseInt(args[0]));
    for (int i = 0; i < factors.length; i++)
      System.out.println(factors[i]);
  }

  public static int[] findFactors(int multiple) {
    List factors = new LinkedList();
    int[] primes = PrimeGenerator.generatePrimes((int) Math.sqrt(multiple));
    for (int i = 0; i < primes.length; i++)
      for (; multiple % primes[i] == 0; multiple /= primes[i])
        factors.add(new Integer(primes[i]));
    return createFactorArray(factors);
  }

  private static int[] createFactorArray(List factors) {
    int factorArray[] = new int[factors.size()];
    int j = 0;
    for (Iterator fi = factors.iterator(); fi.hasNext();) {
      Integer factor = (Integer) fi.next();
      factorArray[j++] = factor.intValue();
    }
    return factorArray;
  }
}
```

I tested the program by running the main program with several different arguments. They all seemed to work. Running it with 100 gave me 2, 2, 5, and 5. Running it with 32767 gave me 7, 31, and 151. Running it with 32768 gave me fifteen twos.

Jerry just sat there and watched me. He didn't say a word. This made me nervous, but I kept on massaging and testing the code until I was happy with it. Then I started writing the unit tests.

"What are you doing?" asked Jerry.

"The program works, so I'm writing the unit tests." I replied.

"Why do you need unit tests if the program already works?" he said?

I hadn't thought of it that way. I just knew that you were supposed to write unit tests. I ventured a guess: "So that other programmers can see that it works?"

Jerry looked at me for about thirty seconds. Then he shook his head and said: "What are they teaching you guys in school nowadays?"

I started to answer, but he stopped me with a look.

"OK", he said, "delete what you've done. I'll show you how we do things around here."

I wasn't prepared for this. He wanted me to delete what I had just spent thirty minutes creating. I just sat there in disbelief.

Finally, Jerry said: "Go ahead, delete it."

"But it works." I said.

"So what?" Said Jerry.

I was starting to get testy. "There's nothing wrong with it!" I asserted.

"Really." he grumbled ; and he grabbed the keyboard and deleted my code.

I was dumbfounded. No, I was furious. He had just reached over and deleted my work. For a minute I stopped caring about the prestige of being an apprentice of Mr. C. What good was that apprenticeship if it meant I had to work with brutes like Jerry? These, and other less complimentary, thoughts raced behind my eyes as I glared at him.

"Ah. I see that upset you." Jerry said calmly.

I sputtered, but couldn't say anything intelligent.

"Look." Jerry said, clearly trying to calm me down. "Don't become vested in your code. This was just thirty minutes worth of work. It's not that big a deal. You need to be ready to throw away a lot more code than that if you want to become any kind of a programmer. Often the best thing you can do with a batch of code is throw it out."

"But that's such a waste!" I blurted.

"Do you think the value of a program is in the code?" he asked. "It's not. The value of a program is in your head."

He looked at me for a second, and then went on. "Have you ever accidentally deleted something you were working on? Something that took a few days of effort?"

"Once, at school." I said. "A disk crashed and the latest backup was two days old.

He winced and nodded knowingly. Then he asked: "How long did it take you to recreate what you had lost?"

"I was pretty familiar with it, so it only took me about half a day to recreate it."

"So you didn't really lose two days worth of work."

I didn't care for his logic. I couldn't refute it, but I didn't like it. It had *felt* like I had lost two days worth of work!

"Did you notice whether the new code was better or worse than the code you lost?" he asked.

"Oh, it was much better." I said, regretting my words the instant I said them. "I was able to use a much better structure the second time."

He smiled. "So for an extra 25% effort, you wound up with a better solution."

His logic was annoying me. I shook my head and nearly shouted: "Are you suggesting that we *always*

throw away our code when we are done?”

To my astonishment he nodded his head and said: “Almost. I’m suggesting that throwing away code is a valid and useful operation. I’m suggesting that you should not view it as a loss. I’m suggesting that you not get vested in your code.”