Agile Software In Process

A Software Book With A Story

Steven Solomon

Agile Software In Process

A software book with a story

Steven Solomon

This book is for sale at http://leanpub.com/agilesoftware

This version was published on 2020-03-07

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Steven Solomon by spreading the word about this book on Twitter!

The suggested hashtag for this book is #TimAndMikePair.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#TimAndMikePair

To my parents, who have supported me in more ways than I can recall.

Contents

A Stumble (draft)

It is a clear summer evening. The majestic blue and green fireworks are thrown across the night sky, while Mike looks down at his buzzing phone. He sees a text from John, “Everything is broken!”

The winding road out of the picnic area is lined with cars, forcing Mike to drive in the center of the road. His jeep’s headlights revealing their colors for a brief moment. His heart is pounding, as adrenaline courses through his veins. He swerves left and right in an attempt to avoid potholes, nearly hitting the parked cars. Grabbing the wheel with his left hand, he fumbles to dial John’s number with the right.

“What is wrong,” Mike asks with haste.

John answers, “Everything. The new subscription feature, the batch job, and Frank is livid that customers are being overcharged.”

“What,” exclaims Mike. He pushes down harder on the gas pedal, hoping to arrive back at the lodge sooner. The stress causes his mind to drift back to the aroma of grilled meats and vegetables filling the air. The calm of drinking a beer on the hill overlooking the treeline, with the city far in the distance. The sun beginning its descent. On the way it paints shades of orange, blue, and purple across the summer sky. The people with their picnic blankets enjoying the firework show.

“Mike!? Frank was just able to sign up without putting his credit card in on his phone. Not to mention, our help center is overrun with screaming customer’s calls,” reports John.

The jeep reaches the end of the campground road, with the extra space, Mike is on the highway in seconds. “I can’t believe this is happening,” he thinks to himself. His mind reviewing the flairs of bright pink and gold, which made a sizzling sound as they sparked, and zig-zaged downward. The fireworks began to perform at a drum beat, the rhythm of ba ba bum, ba ba bum, screech, pop. Ba ba bum, ba ba bum, screech, pop.

“Dang,” screams Mike. He looks back frantically noticing he just ran a red light. He eases of the gas, and shouts, “Let me call you back once I get signed in.” He ends the call, and tosses his phone on the front passenger seat.

Mike pulls up the gravel driveway of the lodge, and swings his jeep into the first spot, slamming on the breaks. He turns the car off as fast as he can, and rips the e-brake up. Mike leaps from the car, running up the wooden stairs. At the door, he fumbles for his keys, sweat dripping down his face. Once in the door, he grabs the handrails of the main stairs, as he skips two or three stairs at a time. He turns into the first room the left of the landing. He lets himself fall into a old desk chair, pulling his laptop from a backpack he places it on the desk, and impatiently smashing the power button.

Cursing the slow computer, he anticipates the login screen to appearing. He quickly types in his password and slams enter. “Come on. come on,” he mutters to himself, watching the loading screen.

Once his desktop loads, he opens the Putty program and establishes a remote connection to the production server.

While it connects, he wishes that the other two engineers had not quit last week. Pulling out his phone he calls John back.

“Mike,” he shouts, “Where have you been? Since you hung up it has charged customers two more times.”

“Sorry, I needed to focus on driving.” Mike says in exasperation. “Let me just connect to the server.”

“Alright, but hurry up. Frank is blowing up my phone, and now there is an email thread with all the executives talking about this issue,” says John.

“Okay, okay. I know we don’t want customers to get charged again. Just give me a second.”

As Mike final connects to the server, and opens the cron jobs:

```
Last login: Fri July  6 15:02:43 on ttys001
You have new mail.
> crontab -e
```

He reminded himself to be careful, noting the location of the batch billing for the website, and the monthly sales reports, that get generated on this machine.

```
# THIS IS IMPORTANT! DON'T DELETE THE BILLING JOB
* 23 * * * /IMPORTANT/batch-billing.sh 'visa' 'en-GB' 'en-US'
* 23 * * * /IMPORTANT/monthly-sales.sh
0 * * * * /subscriptions/charge.sh
* 23 0 * * /subscriptions/report.sh
```

He locates and removes the line that executes the new subscription batch jobs. For good measure, he also removes the job that creates the report that is sent to his bosses email. “No need to add fuel to the fire,” Mike thinks to himself.

```
# THIS IS IMPORTANT! DON'T DELETE THE BILLING JOB
* 23 * * * /IMPORTANT/batch-billing.sh 'visa' 'en-GB' 'en-US'
* 23 * * * /IMPORTANT/monthly-sales.sh
~~0 * * * * /subscriptions/charge.sh~~
~~* 23 0 * * /subscriptions/report.sh~~
```

Mike saves the crontab file. “Alright, the billing job is stopped,” he informs John.

“The batch-billing is down too,” shouts John, in confusion.

Mike clarifies, “No, I mean the subscription billing job. The one that has been charging everyone.”

"Oh, geez. Mike, you just gave me a heart attack. We don't need any more problems right now. Frank is already pissed," scolds John. "Now, what about that subscription signup widget, can you hide that?"

"Yeah, let me change shut off the feature flag," says Mike.

Mike opens up the `setenv.sh` for the Tomcat server.

```
export CONNECTION_STRING="......."
export THREADS="5"

export PUBLIC_KEY="....."
export LOGS_PATH="logs/splunk"

export FAKE_CARD_NUMBER="4111111111111111"
export FAKE_CARD_EXPIRY="02/80"

export SUBSCRIPTION="1"
export SUBSCRIPTION_WIDGET="1"
export SUBSCRIPTION_CHECKOUT_OPTION="1"

export DEFAULT_LANGUAGE="en-GB"
export FAKE_CARD_SEC_CODE="123"
```

Mike switches all of the subscription feature flags to off.

```
export CONNECTION_STRING="......."
export THREADS="5"

export PUBLIC_KEY="....."
export LOGS_PATH="logs/splunk"

export FAKE_CARD_NUMBER="4111111111111111"
export FAKE_CARD_EXPIRY="02/80"

~~export SUBSCRIPTION="1"~~
~~export SUBSCRIPTION_WIDGET="1"~~
~~export SUBSCRIPTION_CHECKOUT_OPTION="1"~~
export SUBSCRIPTION="0"
export SUBSCRIPTION_WIDGET="0"
export SUBSCRIPTION_CHECKOUT_OPTION="0"

export DEFAULT_LANGUAGE="en-GB"
export FAKE_CARD_SEC_CODE="123"
```

Upon saving the file, Mike asks, “Can you check the site?”

“Alright,” says John. “Dang! No the subscription banner is still there. Yeah and the checkout widget too.”

“Ahh, it didn’t pick up the environment variable change.” Mike google searches `setenv.sh not picking up changes` Mike falls silent as he reads about how Tomcat uses the sentenv.sh script. “Oh snap,” says Mike. “It says I need to restart the server for it too reload the variables.”

“You have to take down the entire site,” shouts John. “Are you serious?”

“I don’t want to, its how it works John,” Mike says defensively.

“Well, if you are going to do it hurry up,” snaps John.

Mike stops the server.

```
> catalina.sh stop
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /java-home/jre
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tom\
cat-juli.jar
```

Then Mike starts it again.

```
> catalina.sh run &
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /java-home/jre
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tom\
cat-juli.jar
Tomcat started.
```

“It’s been restarted John,” reports Mike.

“Okay, I’m refreshing.” John lets out a loud sigh of relief, “Awesome, the feature is gone. Thanks Mike, I am going to let Frank and the executives know.”

After the call ends, Mike sits back in the uncomfortable wooden chair. “Thankfully that is over,” he says to himself, as he disconnects from the server. Mike stares off into space contemplating how this went so poorly. He wonders, “What will happen on Monday. Will Frank fire me? I am the only engineer left. Maybe, he needs me, at least until they hire someone else.”

Outside the sound of rock music and gravel crunching fill the air, as his friends fly up the driveway in their car. They laugh and yell, as they enter the house. Mike stays in his room, and moves to

the old matres. He spends the rest of the evening staring at the ceiling, worrying about his job. His friends continue to yell and listen to music out until 2 a.m., but he just lays there.

The weekend passed in a haze. As he drove back to the city on Sunday night, he couldn't stop worrying. Monday at the office came and went, with no word from Frank. Frank didn't even come in, which made Mike even more worried. "He must be really angry," Mike thought to himself. Tuesday passed in the same way. Then Wednesday, he could barely concentrate on his work. Eventually, it was the next weekend. The week had passed and not a soul had spoken to him about the major issue, not even an email was sent. He wondered, "Am I going to be okay?"

On Monday morning, when he entered the office, Frank's office door was wide open. He took a peek inside, and notices that there was nothing in the room. Just an empty desk and shelves. Frank was gone without a trace. Rushing to his desk he checked his email. Some sales messages but nothing about the outage or Frank's departure. Then he saw a new email come through, effective immediately Jen Richards is now Product Owner for the website and mobile application team.

"Mike," said Jen, causing Mike to leap in surprise. His heart was racing.

"Jen," he replied, his voice trembling with shock.

She smiled slightly, saying "We need to talk about the subscription feature."

Losing the path: The team tries pairing (draft)

It is 8 am, in the Big Ticket Book office, the clacking sounds of key presses echo throughout the vast room. Mike sits at his desk in the dark. The bright blue screen reflects on his glasses. He types wildly, as mumbles to himself.

“Yes,” he whispers as the new widget that he has been working on springs to life.

His celebration causes him to ignore the sound of shoes moving up the office hallway. Suddenly, he is totally blinded by a bright flash of white blurring blue dots. Raising his arms in a futile attempt to shield his eyes, Mike lets out a loud grown.

“Oh, good morning Mike,” says a Tim, a tall blond man whose hand is still resting on the light switch. Looking perplexed by Mike’s reaction, he asks, “Why are you in the dark?”

“Ugh, just finishing up that shopping cart change from yesterday.”

Tim flashes his disgruntled associate a skeptical look as he sits down at his workstation. “I have some emails to catch up on but I’ll be with you shortly,” he says.

Mike nods as he refocuses on his computer screen.

The morning passes in silence between the two coworkers. Tim sets up some coffee meetings, for later in the afternoon. He muses how great it will be to catch up with the people he hasn’t seen in a few years. It feels like a lifetime since he first joined the company.

Mike hacks away, barely moving except for his lightening fast fingers. Every so often, he groans when his shopping cart widget breaks.

Around 1pm, a tall curly haired woman steps off the elevator. Her heels snap on the hardwood floor as she approaches the developer’s workspace. While quickly responding to a few memos, she grins at her updated email signature that includes her new title, Senior Product Owner. She enters the room and instinctively approaches Mike, passing by Tim without noticing him.

“Good afternoon, Mike,” she says.

“Uh, hey Jenn,” he rapidly replies turning around his chair to face her.

“I just got off the phone with the CFO. He wants to increase our prices for Audio books…”

Mike cuts her off “…Why?…”.

“…Because we are losing out on revenue for that LOB. Our competitors are already charging 3.5 times the base price.”

“Uh, changing requirements are the…” Mike begins to whine before Tim quickly pipes in.

“Good morning Jen,” Tim says, as he steps forward to shake her hand.

A big smile spreads across her face, “Morning Tim, I am very glad that you are back. How have you been?”

“Great, still getting settled. How is everything with you?”

“Awesome, well actually, I was just telling Mike that we need to increase the price of Audio books that are New Releases. It is something our competitors have beat us to the punch on.”

“Sounds really important, how can I help?”

Mike rolls his eyes.

“It is definitely crucial! Could you work with Mike to get this sorted ASAP?”

“We can get that started but we will have to halt work on the shopping cart change that Mike has been working on.”

Mike begins to raise his hand in disapproval but Jen nods in agreement and says, “That is way less important anyway.”

“Cool, will you be around if we have any questions?”

“Yeah –” she looks down at her watch, “Dang it, actually I am in meetings the rest of the day. In fact I’m late for one now. Gotta run! Maybe in the morning?” Jen speedily waves goodbye, not waiting for an answer.

Mike lets out a sigh and turns back to his workstation.

“Hey Mike,” says Tim.

“Yeah?”

“What do you say we try that pair programming that I was talking about before?”

Mike spins around to face Tim with his face scrunched up. “Why?”

“Well, I wanted to get some experience with the price calculation part of our system and you were already working with it for the shopping cart feature.”

Mike starts to sigh, “Oh alright, we can do that for a…” Tim pulls up a chair before the sentence escapes Mike’s lips.

Mike puts the shopping cart work on a feature branch.

```
$ git co -b shopping-cart-change
```

He then adds all of the files and commits

```
$ git add .
$ git commit -m "stopping for now"
```

Before he can hit enter, Tim interjects, “Would you mind putting a ‘wip’ marker as a prefix for that commit, so that we know it is not complete?”.

Mike shrugs, “Sure, I guess.”

```
$ git commit -m "wip: stopping for now"
```

Again Tim interjects before Mike can hit enter, “What exactly changed?”

“Well, I’m now grouping duplicate books in the cart so that users see a count multiplier.”

“You mind including that in the commit message?”

Mike looks sideways at Tim, he asks, “Why does that matter?”

“It would be nice to know inside the commit message ‘the why’ behind the changes.”

Mike thinks for a moment, scratching his beard, frozen like a statue. “Yeah, I guess that makes sense.” He updates the commit message.

```
$ git commit -m "wip: grouping duplicate books in cart"
```

This time, he rapidly hits enter and pushes the code up to the remote, making sure to leave no time for comment.

```
$ git push origin head
```

Tim just looks at Mike, wondering if he ran the tests but thinks better of bringing up any more suggestions. Mike switches back to the master branch.

```
$ git co -
```

Then he looks to Tim, “Where should we start?”

“Let’s try to find that new release in the codebase.”

Mike agrees and begins to search for the classes named “NewRelease”, but nothing shows up. He tries “Release”, yet there is still nothing.

“I have an idea,” says Tim, grabbing the keyboard to searches for “type” with case matching off.

The pair discovers the Book class in the matching results. As Tim opens the file, Mike remarks, “That looks promising.” Tim scrolls through the Book implementation, he finds the calculatePrice method. Tim pensively looks over the code.

Mike emotes, “What th–”

The pair freezes as the size of the mess they just found washes over them. Tim says, “It’s nice that there is a type field, but it’s not used”

Book.java

```
30   public String calculatePrice() {
31     double totalPrice = basePrice.doubleValue();
32
33     double m = 1.5d;
34
35     String[] splitString = name.split(" ");
36     if ("new release".equals(splitString[splitString.length - 1].toLowerCase()))
37       m += .5d;
38
39     totalPrice = totalPrice * m;
40
41     String[] parts = name.split(" ");
42     int i = 0;
43     for (int k = 0; k < parts.length; k++) {
44       if ("classic".equals(parts[k].toLowerCase())) {
45         i = k; break;
46       }
47     }
48
49     double sum = 0;
50     sum += m * .78;
51
52     if (type == 2)
53       return String.format("$%s", basePrice);
54
55     if (parts.length - 1 == i) {
56       totalPrice = popularity * totalPrice;
57     }
58
59     if (popularity != null)
60       sum += 10d * popularity;
61
62     return new DecimalFormat("$#.00").format(
63       totalPrice
64     );
65   }
```

“This is crap!” exclaims Mike. “I want to know who did this!” Mike switches to the terminal and begins searching the git history for the Book class.

```
$ git log --follow src/main/java/com/music/app/Book.java
```

Tim responds with a stern voice, “Let’s not seek to blame our coworkers. If we feel the code could be better, it’s our job to introduce that improvement.”

Mike squeaks “But why…”

“Nah man, let’s take some ownership.”

“What do you mean?”

“Let’s clean up the code, it is an example of ‘making the change easy, then make the easy change’.”

“Ugh, fine – What’s first?” Mike retorts as palpable frustration fills the air.

Tim grabs a notepad, “Okay, this is what we are going to do. First we are going to change this code to use the type from the database. Then we are going to extract classes for each type.” Pausing briefly to brush the hair out of his face. “Last, let’s make the change to the calculation in the New Release class, so it doesn’t affect the other types.” Tim places the pad and pen between them, so that they both could see it.

“That’s a waste of time.”l

“Why do you say that?”

“Well, we know what we have to do. We should just change the code.”

“A second ago, you said it was crap?”

“Sure, but I know we could get it working.”

“Don’t you want to make it easier to work on?”

“No, I just want to get back to the shopping cart feature.”

Tim looks shocked, “Mike, our job as engineers isn’t just writing code. It is also about clearing the way to make code easier to change.”

Mike lets out a deep sigh of protest, which Tim ignores. “Alright, pull up the tests for the Book class. Let’s spend a few minutes verifying that the tests define the existing behavior,” says Tim. “You see each Book type is in the test name?”

“Yeah.”

BookTest.java

```
1  package com.book.app;
2
3  import org.junit.Test;
4
5  import java.math.BigDecimal;
6
7  import static org.assertj.core.api.Java6Assertions.assertThat;
8
9  public class BookTest {
10   @Test
11   public void whenBookNewRelease_itsPriceIsTwoTimesBasePrice() {
12     Book book = new Book(
13       1L,
14       "Detective Novel: New Release",
15       3,
16       "123456",
17       new BigDecimal("1"),
18       null
19     );
20
21     assertThat(book.calculatePrice()).isEqualTo("$2.00");
22   }
23
24   @Test
25   public void whenBookNewRelease_itsPriceIsTwoTimesBasePrice_2() {
26     Book book = new Book(
27       1L,
28       "Movie Prop Photos: New Release",
29       3,
30       "345345",
31       new BigDecimal("2"),
32       null
33     );
34
35     assertThat(book.calculatePrice()).isEqualTo("$4.00");
36   }
37
38   @Test
39   public void whenClassic_hasSevenRating_itsPriceIsOneAndAHalfTimesBasePricePerPage(\
40 ) {
41     Book book = new Book(
42       1L,
```

```
43        "Dracula: Classic",
44        1,
45        "545463",
46        new BigDecimal(".25"),
47        7
48      );
49
50      assertThat(book.calculatePrice()).isEqualTo("$2.62");
51    }
52
53    @Test
54    public void whenClassic_has10PopRating_itsPriceIsOneAndAHalfTimesBasePricePerPage(\
55  ) {
56      Book book = new Book(
57        1L,
58        "Catcher in the Rye: Classic",
59        1,
60        "237237",
61        new BigDecimal(".25"),
62        10
63      );
64
65      assertThat(book.calculatePrice()).isEqualTo("$3.75");
66    }
67
68    @Test
69    public void whenRegularMovie_itsPriceIsBasePrice() {
70      Book book = new Book(
71        1L,
72        "No one buys this",
73        2,
74        "987654",
75        new BigDecimal("2.99"),
76        10
77      );
78
79      assertThat(book.calculatePrice()).isEqualTo("$2.99");
80    }
81  }
```

"Let's clean up the tests up," says Tim.

Raising his eyebrow in confusion, Mike replies, "They look pretty clean to me."

"Those magic numbers are tripping me up. I want to extract some explanatory variables."

"Magic numbers?"

"Yeah, a primitive data type that has a special meaning. Use the extract field refactoring built into IntelliJ. Highlight the magical `3` on line 16 and use `option + command + F` to introduce a field."

"Wow!" Mike's jaw drops as he completes the key sequence.

"Now type in name that tells us what it is doing"

Mike types "NEW_RELEASE" then hits enter. The editor prompts him to replace other instances.

"Yeah," says Tim.

Mike taps enter and looks over the automated refactoring that just took place.

```
9  public class BookTest {
10
11   private final int NEW_RELEASE = 3;
12
13   @Test
14   public void whenBookNewRelease_itsPriceIsTwoTimesBasePrice() {
15     Book book = new Book(
16       1L,
17       "Detective Novel: New Release",
18 ~~~~~~3,~
19
20       NEW_RELEASE,
21       "123456",
22       new BigDecimal("1"),
23       null
24     );
25
26     assertThat(book.calculatePrice()).isEqualTo("$2.00");
```

"That is great," says Tim, "Now, let's do the same for the other two."

Mike does the same extract field refactoring converting the magic number `1` to `Classic`, and `2` to `Standard`:

```
9  public class BookTest {
10
11   private final int NEW_RELEASE = 3;
12   private final int CLASSIC = 1;
13   private final int STANDARD = 2;
```

Mike thinks out loud, “I know `Standard` isn’t a great name.”

“It’s good enough for now. We can talk to Jen later about how the business refers to it.” Tim writes another item on the todo list. `Find name for` Standard `Books`. “Do you mind a brief break,” asks Tim standing up from his chair. “Say, 10 minutes?”

“Yeah sure.” Mike pulls out his phone to check Twitter, holding it inches from his face.

After a brief phone call, Tim returns to the desk. “Sorry about that.”

“Okay, let’s commit this change,” says Mike.

“Oh dang, we never ran the tests. What a Shame.”

Mike executes the tests for the Book class, they all pass.

“Alright, move those magic numbers into the Book class.”

Mike cuts the fields they extracted and pastes them in the Book class.

“Wait a tick. Put those back.”

“Are you sure,” Mike asks with hesitation.

“Trust me,” promises Tim.

Mike puts the code back.

“Cool, now make the constants static.”

```
17 public class Book {
18 ~~  private final int NEW_RELEASE = 3;~~
19 ~~  private final int CLASSIC = 1;~~
20 ~~  private final int STANDARD = 2;~~
21   static final int NEW_RELEASE = 3;
22   static final int CLASSIC = 1;
23   static final int STANDARD = 2;
24
25   @Id
26   private Long id;
```

“Now if you put your cursor on the constant and hit `Ctrl + t` it brings up a refactoring menu. Select ‘Move’, then enter the name `Book`. Now refactor.”

```
17 public class Book {
18   private final int NEW_RELEASE = 3;
19   private final int CLASSIC = 1;
20   private final int STANDARD = 2;
21
22   @Id
23   private Long id;
```

Mike is floored. He can't believe that it was so easy. He exclaims, "The test even references the change!"

Book Test

```
11   @Test
12   public void whenBookNewRelease_itsPriceIsTwoTimesBasePrice() {
13     Book book = new Book(
14       1L,
15       "Detective Novel: New Release",
16       Book.NEW_RELEASE,
17       "123456",
18       new BigDecimal("1"),
19       null
```

"What's next? Should we create an enum for each type," asks Mike.

Tim thinks for a moment and then answers, "No need, the only benefit at the moment would to communicate that types are mutually exclusive. We already communicated the fact because a Book can only have one type. How about we start using those types in `calculatePrice`?"

"I'm on it." Mike replaces the check for new releases.

Adding type condition

```
37     double m = 1.5d;
38
39     String[] splitString = name.split(" ");
40     if ("new release".equals(splitString[splitString.length - 1].toLowerCase()))
41       m += .5d;
42
43     if (type == NEW_RELEASE)
44       m += .5d;
45
46     totalPrice = totalPrice * m;
```

"It looks really good Mike, but you want a Yoda condition in there."

"Uh, what?"

"It is when the condition is backwards just like how Yoda talks."

Mike scratches his beard trying to figure out what Tim means.

Tim places the cursor on the == and taps `alt + enter`, then selects `Flip ==`.

```
37      double m = 1.5d;
38
39      if (NEW_RELEASE == type)
40        m += .5d;
```

Mike is shocked. He calculates, "But isn't that like saying blue is the sky, or red is the car? Or in this case, New Release is the type."

"True, it is less readable. However, the benefit is that it prevents accidental assignment."

"Ah, I see. If it is ever changed to a single equals sign, it won't re-assign the variable."

"Exactly," says Tim. He asks permission before continuing to type. Mike gestures in agreement.

Tim replaces the hardcoded `2` on with `Standard` and reruns the tests.

```
52      double sum = 0;
53      sum += m * .78;
54
55      if (STANDARD == type) {
56  ~~~~~~if (type == 2) {~~~
57        return String.format("$%s", basePrice);
58      }
59
60      if (parts.length - 1 == i) {
```

Once the tests pass, Tim pauses, "You see that?"

"See what?"

"Look at line 52 through 63"

```
50      }
51
52      double sum = 0;
53      sum += m * .78;
54
55      if (STANDARD == type) {
56  ~~      if (type == 2) {~~
57        return String.format("$%s", basePrice);
58      }
59
60      if (parts.length - 1 == i) {
61        totalPrice = popularity * totalPrice;
```

“You mean the `sum` variable? What about it?”

“It is disused.”

“Dis-what?”

“It was used in the past but no longer. So, we can delete it.”

“No! What if we need it,” exclaims Mike.

“If it is important, we can always bring it back. We have it in source control.” says Tim calmly.

“Fair enough,” replies Mike. “Sorry for the over-reaction.”

“No worries,” says Tim as he deletes the dead code. Then he reruns the test. “Ah, they still pass.”

```
50      }
51
52  ~~  double sum = 0;~~
53  ~~  sum += m * .78;~~
54
55      if (STANDARD == type)
56        return String.format("$%s", basePrice);
57
58      if (parts.length - 1 == i) {
59        totalPrice = popularity * totalPrice;
60      }
61
62  ~~  if (popularity != null)~~
63  ~~    sum += 10d * popularity;~~
64
65      return new DecimalFormat("$#.00").format(
66        totalPrice
67      );
```

Tim continues by replacing the last condition, for the `Classic` Books. As a result, he gets to delete the wonky looking for loop.

```
39      if (NEW_RELEASE == type)
40        m += .5d;
41
42      totalPrice = totalPrice * m;
43
44  ~~~~String[] parts = name.split(" ");~~
45  ~~~~int i = 0;~~
46  ~~~~for (int k = 0; k < parts.length; k++) {~~
47  ~~~~~~if ("classic".equals(parts[k].toLowerCase())) {~~
48  ~~~~~~~~i = k; break;~~
49  ~~~~~~}~~
50  ~~~~}~~
51
52      if (STANDARD == type)
53        return String.format("$%s", basePrice);
54
55  ~~~~if (parts.length - 1 == i) {~~
56  ~~~~~~totalPrice = popularity * totalPrice;~~
57  ~~~~}~~
58
59      if (CLASSIC == type) {
60        totalPrice = popularity * totalPrice;
61      }
62
63      return new DecimalFormat("$#.00").format(
64        totalPrice
65      );
```

The pair pauses momentarily to admire their work.

“It’s looking really nice,” says Mike.

```
34   public String calculatePrice() {
35     double totalPrice = basePrice.doubleValue();
36
37     double m = 1.5d;
38
39     if (NEW_RELEASE == type)
40       m += .5d;
41
42     totalPrice = totalPrice * m;
43
44     if (STANDARD == type)
45       return String.format("$%s", basePrice);
46
47     if (CLASSIC == type) {
48       totalPrice = popularity * totalPrice;
49     }
50
51     return new DecimalFormat("$#.00").format(
52       totalPrice
53     );
54   }
```

Pointing with the mouse Tim says, “You see these variables, `totalPrice` and `m`, are no where near the code that uses them?”

“Yup!”

“We should make each condition really explicit.”

“Done,” Mike says as he copies the variables to make the `if-else-chain` crystal clear.

“That’s a pretty big step to take.”

```
34   public String calculatePrice() {
35     double totalPrice = basePrice.doubleValue();
36
37     double m = 1.5d;
38
39     if (NEW_RELEASE == type)
40       m += .5d;
41
42     totalPrice = totalPrice * m;
43
44     if (STANDARD == type)
45       return String.format("$%s", basePrice);
```

```
46
47    if (CLASSIC == type) {
48      totalPrice = popularity * totalPrice;
49    }
50
51    return new DecimalFormat("$#.00").format(
52      totalPrice
53    );
54
55
56    double m = 1.5d;
57    double totalPrice = 0;
58    totalPrice = totalPrice * m;
59
60    if (STANDARD == type) {
61      return String.format("$%s", basePrice);
62    } else {
63      return new DecimalFormat("$#.00").format(
64        totalPrice
65      );
66    }
67  }
```

Mike runs the tests. The bar turns red, and he immediately begins flailing. He quickly moves variables around the file, re-running the tests, rapid fire. Move, fail; move, fail.

“Slow down, Mike,” Tim cautions.

Mike copy pastes code, run the tests, and they fail. He tweaks code, run them without pausing, and they fail again.

“Okay, hold on. There is no need to do that,” Tim says while asking permission to drive. “If you just made the tests fail, don’t reason about it too much. You know the last passing code was moments ago.” Tim simply undoes the changes. “Okay, let’s take smaller steps. First I want to handle the simplest case, which is the `Standard` type.” Tim moves the standard condition to the top of the `calculatePrice`. “For good measure, I am going to add curly braces as well.”

```
34    public String calculatePrice() {
35      if (STANDARD == type) {
36        return String.format("$%s", basePrice);
37      }
38
39      double totalPrice = basePrice.doubleValue();
40
41      double m = 1.5d;
42
43      if (NEW_RELEASE == type)
44        m += .5d;
45
46      totalPrice = totalPrice * m;
47
48  ~~    if (STANDARD == type)~~
49  ~~      return String.format("$%s", basePrice);~~
```

The tests still pass.

"Dang!" remarks Tim, "The `Classic` and `New Release` logic is really tied together." He blows the hair out of his eyes. "Okay, let's duplicate the multiplication and re-assignment of `totalPrice`."

"Wait, duplicate?" Mike raises his eyebrows.

"Yeah, it is too hard to see what the actual calculation right now. We need to make it clearer."

"I thought you were one of those 'Clean Code' guys! Why would you want to create duplication? Doesn't that fly in the face of that Do Not Repeat Yourself (DRY) principle you were talking about the other day?"

"Nope."

"No…" Mike leans forward, waiting for a response.

"DRY has nothing to do with duplicate code."

Mike stares blankly, his eyes glazing over.

"It is about not having duplicate ideas."

Mike's eyes widen.

"Think about it this way: Could you have an entire conversation about `New Releases` without mentioning `Classics`?"

"Yeah sure."

"Why is that?"

"Since the business rules can change at different times, they don't necessarily affect each other." Mike states flatly.

“Definitely. Now think about having that conversation with Jen after you both spent all morning laying out how `New Releases` work.”

“Where are you going with this?”

“How surprised would she be if after that discussion you changed `New Releases` but broke `Classics`?”

“She would flip!”

“Why?”

“Because the two ideas have nothing to do with –” Mike trails off. The look of realization suddenly washes over his face.

“Now you got it.” chuckles Tim.

“Alright, alright. You sold me. How do we move forward?”

“First add an `else` clause under the `Standard` type condition and put all remaining code inside it.”

```
34   public String calculatePrice() {
35     if (STANDARD == type)
36       return String.format("$%s", basePrice);
37     else {
38       double totalPrice = basePrice.doubleValue();
39
40       double m = 1.5d;
41
42       if (NEW_RELEASE == type)
43         m += .5d;
44
45       totalPrice = totalPrice * m;
46
47       if (CLASSIC == type) {
48         totalPrice = popularity * totalPrice;
49       }
50
51       return new DecimalFormat("$#.00").format(
52         totalPrice
53       );
```

“Now that the else clause is in place let’s duplicate the code under a `else-if` clause for `New Release`.”

```
34   public String calculatePrice() {
35     if (STANDARD == type)
36       return String.format("$%s", basePrice);
37     else if (NEW_RELEASE == type) {
38       double totalPrice = basePrice.doubleValue();
39
40       double m = 1.5d;
41
42       if (NEW_RELEASE == type)
43         m += .5d;
44
45       totalPrice = totalPrice * m;
46
47       if (CLASSIC == type) {
48         totalPrice = popularity * totalPrice;
49       }
50
51       return new DecimalFormat("$#.00").format(
52         totalPrice
53       );
54     }
55     else {
56       double totalPrice = basePrice.doubleValue();
57
58       double m = 1.5d;
59
60       if (NEW_RELEASE == type)
61         m += .5d;
62
63       totalPrice = totalPrice * m;
64
65       if (CLASSIC == type) {
66         totalPrice = popularity * totalPrice;
67       }
68
69       return new DecimalFormat("$#.00").format(
70         totalPrice
71       );
72     }
73   }
```

The tests still pass.

“Now that there are two separate paths for each type we can delete the code that is unreachable.”

```
34    public String calculatePrice() {
35      if (STANDARD == type)
36        return String.format("$%s", basePrice);
37      else if (NEW_RELEASE == type) {
38        double totalPrice = basePrice.doubleValue();
39
40        double m = 1.5d;
41
42        if (NEW_RELEASE == type)
43          m += .5d;
44
45        totalPrice = totalPrice * m;
46
47        if (CLASSIC == type) {
48          totalPrice = popularity * totalPrice;
49        }
50
51        return new DecimalFormat("$#.00").format(
52          totalPrice
53        );
54      }
55      else {
56        double totalPrice = basePrice.doubleValue();
57
58        double m = 1.5d;
59
60        if (NEW_RELEASE == type)
61          m += .5d;
62
63        totalPrice = totalPrice * m;
64
65        if (CLASSIC == type) {
66          totalPrice = popularity * totalPrice;
67        }
68
69        return new DecimalFormat("$#.00").format(
70          totalPrice
71        );
72      }
73    }
```

“What’s more, we can even remove conditions that are always true.”

```
34    public String calculatePrice() {
35      if (STANDARD == type)
36        return String.format("$%s", basePrice);
37      else if (NEW_RELEASE == type) {
38        double totalPrice = basePrice.doubleValue();
39
40        double m = 1.5d;
41
42  ~~~~~~  if (NEW_RELEASE == type)~~
43          m += .5d;
44
45        totalPrice = totalPrice * m;
46
47        return new DecimalFormat("$#.00").format(
48          totalPrice
49        );
50      }
51      else {
52        double totalPrice = basePrice.doubleValue();
53
54        double m = 1.5d;
55
56        totalPrice = totalPrice * m;
57
58  ~~~~~~  if (CLASSIC == type) {~~
59          totalPrice = popularity * totalPrice;
60  ~~~~~~  }~~
61
62        return new DecimalFormat("$#.00").format(
63          totalPrice
64        );
65      }
66    }
67  }
```

“It’s still really hard to look at,” remarks Mike.

“You are right. So that’s the reason why we are going to do one more refactoring. Let’s inline the `m` variable.”

```
34    public String calculatePrice() {
35      if (STANDARD == type)
36        return String.format("$%s", basePrice);
37      else if (NEW_RELEASE == type) {
38        double totalPrice = basePrice.doubleValue();
39
40  ~~~~~~double m = 1.5d;~~
41
42  ~~~~~~m += .5d;~~
43
44        totalPrice = totalPrice * 2.d;
45
46        return new DecimalFormat("$#.00").format(
47          totalPrice
48        );
49      } else {
50        double totalPrice = basePrice.doubleValue();
```

“Let’s inline `totalPrice` as well.”

```
34    public String calculatePrice() {
35      if (STANDARD == type)
36        return String.format("$%s", basePrice);
37      else if (NEW_RELEASE == type) {
38  ~~~~~~double totalPrice = basePrice.doubleValue();~~
39
40  ~~~~~~totalPrice = totalPrice * 2.d;~~
41
42        return new DecimalFormat("$#.00").format(
43  ~~~~~~~~totalPrice~~
44          basePrice.doubleValue() * 2.d
45        );
46      } else {
47  ~~~~~~double totalPrice = basePrice.doubleValue();~~
48
49  ~~~~~~totalPrice = totalPrice * 1.5d;~~
```

“There you have it, refactoring complete!” Tim throws his hands up in celebration.

```
34   public String calculatePrice() {
35     if (STANDARD == type)
36       return String.format("$%s", basePrice);
37     else if (NEW_RELEASE == type) {
38       return new DecimalFormat("$#.00").format(
39         basePrice.doubleValue() * 2.d
40       );
41     } else {
42       return new DecimalFormat("$#.00").format(
43         popularity * basePrice.doubleValue() * 1.5d
44       );
45     }
46   }
```

“Now all we have to do is change `2d` to `3.5d` and then we can ship this change. We will deal with those `doubles` later. They have got to go.”

“What about those subclasses for each type of `Book`?”

“We should wait until we have some time to talk with Jen about the domain.”

“Alright,” Mike says. He changes the value from `2d` to `3.5d`.

They both review the changes one last time.

“Now we just wait for it to get to production so we can let Jen know the change is done.” says Tim.

“Wait for?”

“Yeah for the build to finish on our Continuous Integration Box.”

“That doesn’t exist,” barks Mike. “I have to do production deployments by hand. I build the JAR and put it on the server.”

Tim opens his mouth and eyes wide in terror. “You do production deployments by hand?”

“Well yeah, how else do you do them,” asks Mike.

“Okay, we will talk about that later.” Tim pulls out a pack of sticky notes and a black marker. He writes on three individual stickies:

- Add Money Object
- Talk to Jen about Domain Terms
- Build CI Pipeline

Mike commits the code, pushes to the remote git repository, and uses some magic to get the JAR up to production. Once the change is up, he sends Jen an email reporting that the change is live. Tim wheels over to his desk, where he sends Jen an invite for a meeting the next morning at 10:30am.

Both engineers work alone the rest of the day. When 5 o'clock arrives, each cordially waves to the other goodbye. Mike is still typing away when Tim leaves.

As Tim stands waiting for the elevator, he thinks, "Man, there is a lot to do."