Preface

This story does end happily, with data types flowing smoothly across the screen, almost nodding at me like old trusted friends. There was a moment, however, when it wasn't at all that enjoyable; in fact, everything looked downright depressing.

This is what the author posted on the CodeRanch forum on May 20, 2016 (edited slightly for better readability):

> A few days ago I failed my 1Z0-808 at 62%.
>
> [This should] serve as a warning to those who dream about becoming Oracle-certified in a snap. Alright, here it goes:
>
> I started preparing in the earnest six months ago. The first thing I did was quitting my job, quite literally.
>
> After burning the proverbial bridges behind me I watched… no, that's a wrong word; I worked painstakingly through the entire video course, a.k.a. *Live Lessons*, on 1Z0-803 by Simon Roberts. Was spending on that ten to twelve hours daily. Took me three weeks; no holidays, no weekends, no nothing… Why this long? Because I was hitting *Pause* every few seconds to write notes, think over what Simon was talking about and, most importantly, put all of it out in code. By using Notepad++. Yes, that's right. For the first two months I wasn't letting myself fire up an IDE.
>
> Soon running my tiny 'javalets' directly from the CLI became so tedious that I hacked the Registry and wrote a minimalist launcher to run the code by right-clicking on the .java file in the Explorer window. This trick alone allowed me to write and test well over a hundred javalets a day, all the while learning new, even more mind-boggling rules from the JLS.
>
> By my rough estimation, by the mid-January, 2016, I had thought up, tested, twisted, tortured, tormented, and otherwise tweaked with over 3,000 javalets while listening to and watching Simon. (Mind you, not because watching him makes you do all those unspeakable things, no. He did an outstanding work. The best there is. I'm going to write an extensive review about his video course, although not right now. Right now I don't have the right. Right…)
>
> And then I hit the books. Oh boy, did I hit 'em… Here's the list:
>
> – the ubiquitous *OCA: Oracle Certified Associate Java SE 8 Programmer I Study Guide. Exam 1Z0-808* by Jeanne Boyarsky & Scott Selikoff (loved it; positively loved it! granted, it's ridden with typos but still makes a great read; in fact, THE greatest);
>
> – *OCA Java SE 7* by Mala Gupta (liked it, especially all those tightly packed Review Notes; btw, her new book, this time on 1Z0-808, will be released very soon);
>
> – *Head First Java* by Kathy Sierra & Bert Bates (dropped it after a couple of weeks; reason: the delivery grated on my nerves; in my eyes, it'll make a perfect present to one's worst enemy but that's just me; anyhow, all my comments are highly subjective);
>
> – *Thinking in Java* by Bruce Eckel (6th ed.) (not everything, of course; I didn't touch collections, inner classes, threads, and other advanced topics that are not on the exam; I am awed by this book; a few pieces of code resemble poems, even; then again, it's just me);
>
> – few chapters from *Effective Java* by Joshua Bloch (am going to read it ten more times; pure pleasure);
>
> – *Java 8. Pocket Guide* by Robert & Patricia Liguori (I always keep this book within my reach and consult with it dozens of times daily);

- *Java in a Nutshell* by Benjamin Evans & David Flanagan (6th ed., which covers JSE8) (I think I did find a typo in their code but the book itself is undeniably excellent; helped me to look at certain concepts from a different angle; would love to translate it if I only had a chance…);
- *Exercices en Java* par Claude Delannoy (4e éd., couvre JSE8 aussi) (I certainly enjoyed it; would've gladly recommended it to any of my friends);
- *La programmation objet en Java* par Michel Divay (apparently, he teaches this very course at the Univ.of Rennes; the book itself is as dry as moon dust and bo-oring but it has one major advantage: it makes use of simple UML diags, which I find helpful when solving problems);
- and three more books in Russian, which are too obscure to talk about here.

Naturally, just reading books won't cut it so I did exercises. As Jeanne Boyarsky loves to repeat, "Practice!" And so I practiced. It looked like this: read Chapter 1, do the exercises to that Chapter, do not check answers but read Chapter 2 instead, come back and repeat the exercises to Chapter 1, compare your own answers (to see if they are the same, otherwise it would mean that you are unsure about certain points) and finally check them against the key. Loop in the same fashion to finish the book. Cover all the books that have exercises. Then do it again and again until your score becomes at least 90% at every single pass.

Boyarsky & Selikoff's book alone contains 345 exercises if we count in the three mock up exams; I did the whole cycle three times (and read the book itself four times, from cover to cover; now I'm reading it for the fifth time; I did quit my job, remember?). Mala Gupta's book gives a lot of practice, as well…

I also enrolled for a four-week-long course on Java 8 that they teach at Bauman's Technical University here, in downtown Moscow. Three hours of brain wracking practice in class, then solving all those home assignments until four o'clock in the morning. For twenty days.

Attended JPoint, a Java conference for students organized by the local JUG. Was reading all Java-related magazines I could get my hands on in all the languages I'm familiar with. Sat for hours cracking Java Interview questions that can be found on so many sites in India…

The idea was to saturate myself with Java code, until I puke blood or drop dead. Or both. Didn't happen, though.

This epic picture will not be complete without me squinting at my screen while working through emulated exams; I had plenty of those, too:

- two 60-question-long practice exams that are bundled with *OCA SE7 Programmer I & II* by Kathy Sierra, et al.;
- over 600 questions in the Enthuware Question Bank;
- Kaplan Self-Test with its 180 questions;
- as for the testing software that comes with the B&S book, I stopped using it after seeing how many errors it contains; preferred pen and paper instead.

Kathy Sierra's quizzes are tough. Enthuware's even harder – and thoroughly enjoyable. The explanations alone are worth the buck; they are first rate, hands down.

As for Kaplan's… I wish I could sue that sorry bunch. Their so-called 'self-test' (for which I paid over $80 and which is even advertised on the Pearson Vue[2] site as a part of the official Oracle Certification Bundle, of all things) is the worst piece of software workmanship that I have ever seen in my entire life.

In case someone is curious why I resent them that much, here are the reasons; there are two of them… hold it! three![3]

[2] And on Oracle University site, too…

[3] Come to think of it, I can add one more: Kaplan SelfTest website misleads the prospective buyers into thinking that these mock-up tests mimick "the actual test environment perfectly" (I took this expression right from the Testimonials section on their landing page.) Mildly put, it isn't true. In fact, none of the simulated 1Z0-808 prepware suits reproduce the look and feel of the *actual* test envionment – which puzzles me to no end. After all, it's just GUI; no changes to the business logic are needed to recreate the atmosphere of the real exam…

Firstly, the desktop version's GUI is full of visual bugs that literally clutter the screen (seriously!) and it is only the web-based version that works as advertised. Secondly, their flash cards contain errors and ambiguities in the explanations, and finally, the questions are simply… well, too simple. I took their mock-up exam in the so-called Certification Mode and got 96%. Without breaking a sweat, with tons of time to spare, without even reviewing my answers…

Now, about Enthuware. My scores were 77% on Foundation, 74-79-75-81-74-80 on Standard Tests 1 thru 6, and 78% on the Last Day Test. The Enthuware developers have a sort of a running scoreboard on their site; it says that the average Standard Tests score from thousands of users is 74.8% while the actual exam score is 88%. My avscore was 77.2… so I thought I had this exam, this damned 1Z0-808 thingy in my pocket… Over 800 hours of reading, plus coding, plus cracking puzzles, plus almost 9000 lines of review notes that weren't merely copied&pasted from anywhere but were indeed my own brainchildren… Notes to myself. Remember this, remember that… "Watch out for the wrappers' constructors! they accept either underlying primitives or Strings!"… "Be on guard for trim()! StringBuilder doesn't have it!"… "switch accepts compile-time constants only!"… Stuff like that. Even earned myself the 'dry eye' syndrome because of watching too many Simon Roberts videos, so now I'm dragging my sorry derrière around wiping tears from the corner of my right eye every few seconds. Like a fool. All those efforts… down the drain…

How do you call it? this strange, hollow feeling when you know, with a cold, calm, even serene and contented certainty know that you gave everything humanly possible – and yet it wasn't enough?

Do not worry, I am about to finish. I think I know what my problem was and still is. Am going to test this hypothesis next time, around June 17. Will report both the results and my findings.

Thanks for listening.
Signing off for now.

And then, on June 26, 2016…

This is a simple follow-up on my previous post:
I finally passed (98%); now to the 1Z0-809!

There was also one more post that I placed on amazon.com as a review for *OCA: Oracle Certified Associate Java SE 8 Programmer I Study Guide. Exam 1Z0-808* by Jeanne Boyarsky & Scott Selikoff:

Six weeks ago I failed my 1Z0-808 exam at 62%. Was too naïve, cocky and self-confident to the point of arrogance – and the exam creators taught me a lesson. Which I did learn because the day before yesterday I retook the exam and passed it with the 98% score. My study kit included three tools each of which, IMO, is a must for all those who need solid, certification-grade Java skills. Here's the list:

– this study guide, which will give you every single bit of the theoretical and practical knowledge required to pass the exam with flying colors;
– a video course on preparing for the Oracle JSE exams by Simon Roberts (the very person who developed the initial concept of the questions a couple of decades ago; his video course was *the* foundation upon which I built and expanded my knowledge base further by reading this book);
– and, finally, the 1Z0-808 Test Studio by Enthuware, with its 600+ questions, which will probe, drill, grind, boil, and pig-roast you on all possible aspects and nuances. When you are done with it, you will be seeing both Java and yourself in a completely new light.

Clench your teeth, bear the pain – a lot of it is coming your way, especially if you are new to Java as I was – but stick to these three tools, and you *will* get your certification.

Unfortunately, none of these fine instruments can give you the second key to the success: they all discuss in detail purely technical matters while the way you work through and with the questions during the actual exam is equally important. At what speed should you go through the test? Since many of the questions are booby-trapped, is it advisable to second-guess yourself and attempt to re-solve the problem some other way? When you immediately see the "correct" answer, should you check every other option? Or is it better to move on to the next question right away because this way you'll save time, and it is *time* that is the most precious resource during the exam... What about 'cherry picking'? What about the 'Review Later' check-marks? Just how restrained or generous should you be with them? Is it safe to trust your gut instincts rather than do things by the book, checking and re-checking every single LOC methodically? As I found out, and at a great cost, too, some of the answers to these and many other questions are, in fact, counterintuitive and even paradoxical.

> Tactics, guys, I'm talking tactics here. All those study guides, video lessons, exam simulators, and testing suits crafted by talented people are a part of your winning strategy but – sorry for sounding like Captain Obvious – it's the execution that gets the job done. If you think this psychological cr*p is something that belongs to the 70s and that you'll never have use for it, be my guest. Just go ahead, I wish you all the luck. You'll need it. As for me, I had to learn this lesson the hardest way possible, for which I am actually grateful. The initial failure forced me to scrap and then rebuild my entire approach, devise a fool-proof plan of attack, invent shorthand notations for cracking loops, diagramming the objects' lifecycle dynamics, expressing inheritance, polymorphic behavior, etc. and even to formulate dozens of concise, easy-to-remember rules all the while saturating myself with the Java code until my eyes start hurting whenever I see an erroneous snippet. No study guide that I know of, no software and no Java coders' forum teach such skills, which I find quite puzzling because they are simply indispensable. To the extent that I am going to write a small book on the whole experience[4]…
>
> Now, if I may, a direct advice: code. Code as much and as often as you can, immediately test all your ideas or doubts while reading, listening or simply thinking about what you've been learning AND DO NOT use an IDE for the first few weeks as the very minimum. Stick to your Notepad++ instead and run everything from the CLI. This will reveal your soft spots and the typical mistakes you tend to make.
>
> Still reading? Hmm. Well, stop then and go get this study guide by Jeanne Boyarsky and Scott Selikoff, the test suite by Enthuware and the video course by Simon Roberts. Best of luck! I actually envy you: you still have your chance to score the perfect 100% :)

You know the saying, "*Practice makes perfect?*" It isn't true. Only perfect practice makes perfect. To know syntax rules, principles of overriding and so on isn't enough; I witnessed it firsthand. You need to learn how to apply them under stress, when everything works against you.

The pages that follow contain every single trick, approach, or technique I had been using while both preparing for and taking the actual exam. I will tell you *all of it*, down to the most boring and even embarrassing detail – because tactics *is* about details.

There is one problem, though. I did sign the NDA[5] before taking the exam, so I can't possibly give you a laundry list of correct answers to the real questions; often you'll have to figure them out on your own by *inferring*: the very concept that we will meet again when working with generics and lambdas. But do not worry: I'll be at your side all along, dropping hints now and then, and generally helping as much as humanly possible – and permissible by law.

Let me show you an example right away. Why am I reusing my old posts in this section of the book? Because this approach illustrates the principle that you will – you *will*! – encounter in one of the textual questions[6] on the real exam. Can you name it, this principle? It is one of the benefits of polymorphism.

Hint: the preceding paragraph mentions the correct answer in the open. And here's another hint, which is more like a reminder: the Test Studio by Enthuware has it all. *All the answers*. You just have to dig for them a little deeper.

[4] This very book you are reading.

[5] Non-Disclosure Agreement; we sign these whenever we are about to sit an Oracle cert exam.

[6] Half a dozen of questions will have no code in them; they are included to test your knowledge of the most basic concepts of the object-oriented programming (OOP).

PART I
Know Your Enemy

- *Young man! Don't you know that the Department of Public Health warns about dangers of smoking?*
- *It's OK. I'm a programmer, after all.*
- *?!*
- *We spit on warnings and care about errors only.*

Chapter I.0 Toe Suckers and Guinea Pigs

Once nailed, this certification is supposed to become our closest ally – if not a friend. Right now, however, it seems more like an uphill battle. This being said, let’s start by taking a closer look at the adversary’s stronghold.

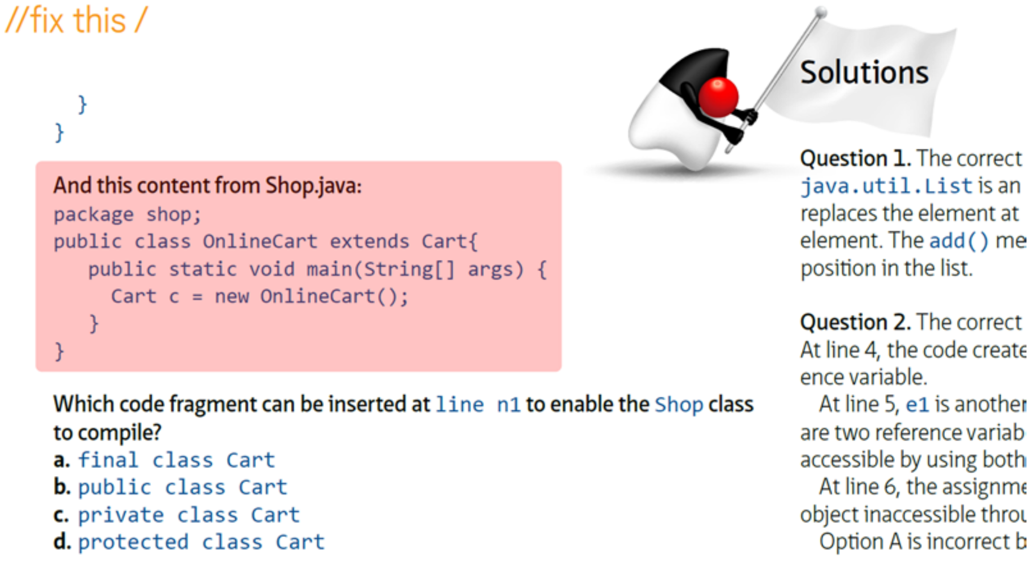
As you probably remember, B&S[7] contains a brief description of the history of OCA exams and what they are for. How about using another angle to better appreciate what’s in store for us?

This is what was published in the *Fix It* section of *Java Magazine*, July-August 2015 (pp.14-15):

> ([…] The purpose of this certification [1Z0-808 – *I.S.*] is to enable beginners to demonstrate their knowledge of fundamental Java concepts. The certification is designed for beginners in programming and/or those with a nonprogramming background who have basic mathematical, logical, and analytical problem-solving skills and who want to begin to learn the Java programming language; and for novice programmers and those programmers who prefer to start learning the Java programming language at an introductory level.
>
> After successfully completing this certification, candidates can confidently utilize their knowledge on Java datatypes, operators, statements, arrays, lists, and exception-handling techniques.)
>
> Naturally, these questions have small embedded traps that spring open if you rush through without considering exactly what the code is doing. […]

Got the picture? In my eyes, it basically says the OCAJSE8 exam – with all its "small" traps – is meant for toddlers if not toe-sucking infants.

Ready? then asks the column in a silver-haired mentor’s tone – and produces this (Q3, 2nd part):



//fix this /

```
  }
}
```

And this content from Shop.java:

```
package shop;
public class OnlineCart extends Cart{
   public static void main(String[] args) {
     Cart c = new OnlineCart();
   }
}
```

Which code fragment can be inserted at line n1 to enable the Shop class to compile?

a. final class Cart
b. public class Cart
c. private class Cart
d. protected class Cart

Question 1. The correct java.util.List is an replaces the element at element. The add() me position in the list.

Question 2. The correct At line 4, the code create ence variable.
At line 5, e1 is another are two reference variab accessible by using both
At line 6, the assignme object inaccessible throu
Option A is incorrect b

[7] Throughout this book, B&S stands for the OCA 1Z0-808 Study Guide by Jeanne Boyarsky and Scott Selikoff. Its OCP sibling, the study guide for 1Z0-809, will be referred to as B&S809.

Take a good look at the highlighted area; hopefully, you can spot the problem right away. If not, re-read the quiz and its LOCs – unhurriedly and *subvocalizing*; this is how you should work through the code whenever you suspect it might be booby-trapped. What are the telltale signs that can arouse such a suspicion in you, how do you apply this technique in different scenarios – all this will be illustrated later; right now I am drawing your attention to something else entirely.

Alright, by now you should have it. Being `public`, **OnlineCart** class cannot possibly be declared in the file **Shop.java**, which makes all those options **a** through **d** simply irrelevant. In other words, while lying a bear trap for us toddlers, the quiz creator[8] shot herself in the leg.

Now, I'm neither saying nor implying she is insufficiently qualified; there's no doubt in my mind she is thoroughly professional: after all, her employer is Oracle Univ., and this credential alone is enough for me. But this example is as real as it gets, and by using it I mean to say that there are so many rules, nuances and subtleties to Java that even the most knowledgeable person might slip occasionally. Unfortunately for us, the exam itself is not that understanding and neither it is forgiving. Paraphrasing Forrest Gump, 1Z0-808 is like a box of chocolates: *ya never know what ya gonna get*. But twenty[9] strikes out of seventy seven, and you are done for.

The exam is delivered from a remote server; it doesn't live on the local machine. (I actually asked the exam administrator in my test center about this and she confirmed that it is so.) The machine was a peculiar looking flat-case desktop PC, deep-space black with no markings, it had a lockable CD/DVD-drive and no USB connectors on the front panel. The rear panel was well out of reach; I didn't managed to get a look at it. To be honest, I didn't even try…

The machine was connected to a small bluish box that looked like a regular Ethernet hub to me. The test room had one more computer, a twin to mine; it was also connected to the hub.

The most unusual feature of the whole setup was the monitor: it was square. Not the usual 4:3, 16:10 or 16:9 aspect ratio, no. It was 1:1. I've only seen such things twice in my entire life, and both times it was on the OCA exam.

Some test centers run their exams on Mac but in my case the OS was Windows-based; probably, some flavor of MS Windows Server because its flash screen had the Windows logo. According to my observations, the access to the exam is protected by a double-authentication protocol.

It seems that each exam is unique in the sense that the questions are most probably taken from a larger bank in a quasi-random order. The available answers (a.k.a. options) among which we have to choose the correct one(s) are shuffled each time the exam subset is being generated.

By 'quasi-random' I mean the following: it looks like[10] that the exam is made up of three major sections:

- a Core section, which contains questions that appear always, on each exam, for every candidate regardless of the date or location; in other words, these questions aren't randomly chosen from the question bank and will be present in every case;

- an Advanced section whose questions test our knowledge on a higher level of comprehension; close to Tough / Very Tough classification by Enthuware; these are the questions that are chosen randomly;

[8] Sushma Jagannath of Oracle University.

[9] Later I'll show you how I arrived at this number. Then again, my logic can be totally bogus. You still have your own head.

[10] The only way to ascertain this is to ask Oracle – and those guys ain't talking.

- a ‘Guinea Pig’ section, which is made up of the questions that Oracle tests on us. As you know after reading B&S, these questions are not graded. At all. Unfortunately, there is no way to tell if your current question is of the ungraded variety, otherwise it would mean that you can spit it in the face, kick its butt and then skip it altogether – and get away with it…

I think this general layout is more or less correct because I met core questions both times, the tough ones were different, and there were some average looking problems that I met only once. I have this feeling that the core questions make up 70 to 80% of the whole set. I can be wrong, though.

When looking up your score, you’ll see a list of the objectives that you have missed; after the second sitting my list contained a single item – "Create and overload constructors; including impact on default constructors" – and since my score was 98%, it appears that a single objective is worth 2%. On the other hand, the failed attempt (62%) gave me a list with 21 items; now you know why I think that the game is over after twenty strikes.

This math, however, does not mean much because we can answer wrongly for a completely different reason that has nothing to do with the objective that we are being tested for. ‘Spooked-by-a-shadow’ sort of thing.

An illustration. Do you remember the **TargetAudience** example? Admittedly, that was a shamelessly contrived piece of code, but still, how many objectives does it incorporate? Let’s count:

- ❑ applying access modifiers such as `public` and `protected`;
- ❑ applying non-access modifiers such as `final`;
- ❑ applying `this` keyword;
- ❑ initing[11] of fields in constructors;
- ❑ using method overloading;
- ❑ working with methods defined in some of the core Java API classes.

At least six, and if we count in `package` and `import` declarations, general class design principles, rules for valid identifiers… Too many. While the code actually tests us on the proper use of constructors, it also throws a lot of dust into our eyes.

Apparently, designing test quizzes is an art in itself, and that’s precisely why Oracle has this ‘Guinea Pig’ section on the exam.

Let’s have another illustration. The same quiz block that appeared in Java Magazine (July-Aug 2015), contained one more somewhat ambiguously formulated question. Please look it up on your own; it concerns garbage collection, which is on the list of the exam objectives, too. Try to figure out what’s wrong with the wording, then refer to the **Quizzical Quiz Questions** in the *Letters to the Editor* column (p.7) in the Nov-Dec 2015 issue (External Resources).

Very well; so Oracle tests new questions on us toe suckers and guinea pigs all in one – and we don’t even get credit for this – but why do they need new questions in the first place? I suspect it’s a defensive mechanism against braindumps. No more on this subject you shall hear from me.

Now, back to the test rig. It may sound silly at first but I strongly advise you to take a long, hard look at the keyboard. See if it has a **Sleep** / **Hibernate** / **Power Off** key on it. Now imagine what happens when you accidentally hit it during the exam.

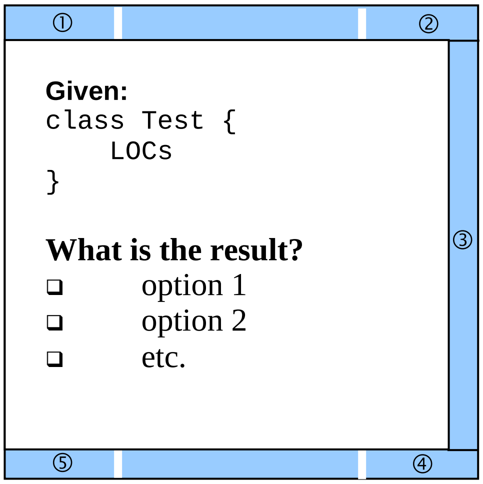
[11] Please refer to the Abbreviations and Acronyms section.

Worst case scenario: game over. Like, totally. The session with remote server has been abruptly terminated, and you're out for good. Best case scenario: the connection can be restored, and you are able to continue right from the point of premature termination.

In fact, when I asked the exam admin about this, she replied that the best case scenario is exactly what would happen. But she wasn't at all sure about the timer…

My keyboard did not have potentially suicidal keys on it. And neither did mouse[12].

Good. Now, to the battlefield, that is, the screen. As I've told you, it was square, and most of the time the picture looked pretty much like this:



Given:

```
class Test {
    LOCs
}
```

What is the result?

❑ option 1
❑ option 2
❑ etc.

Legend:

① – Exam taker's name
Oracle Testing ID

② – Countdown timer, 02:30:00 ➔ 0:00:00
☐ 'Flag for Review' checkbox

③ – Vertical scrollbar

④ – **Review** / **Previous** / **Next** buttons

⑤ – **Help** button

It wasn't the usual, maximized window that takes up almost entire screen leaving accessible just the Taskbar; no, there was no Taskbar, not even a collapsed one – which means no Start button. As for the Ctrl+Alt+Del combo to access the Task Manager with its plethora of interesting options, I think it would have worked but I didn't try it. Neither did I hit Win+R or other hot-key combinations to see if it would be theoretically possible to gain access to the command line, PowerShell, ports, and so on. Most probably, would not have worked; too obvious.

I also think they have disabled the Clipboard; same goes for the PrintScreen key (at least that's how I would have gone about it if someone asked me to harden the exam environment against casual attacks.) Apart from taking the buffer out of the picture, they apparently took care of the selection event handler because the displayed text itself is not selectable.

Pearson Vue suggests that in order to familiarize themselves with the look and feel of the exam software, candidates should visit the following link: http://www.pearsonvue.com/demo/. To be honest, I don't think it is terribly important in our case because the 1Z0-808 GUI is quite simple and straightforward. It's up to you.

Do watch, however, 'What to expect in a Pearson VUE test center' video; it is close to what I have observed (https://home.pearsonvue.com/test-taker/security.aspx). One comment, though: they are checking now our watches. When I asked the administrator if it is because the modern wearable gizmos have built-in cameras and whatnots, she just smiled sweetly and said nothing. Draw your own conclusions.

[12] No, I didn't check the mouse pad – because there was none! Otherwise… ☺

Before taking the exam, you are supposed to sign some papers including the NDA and Oracle Certification Terms & Conditions; you'll see this legal stuff again, this time right on the screen. As soon as you click 'I accept', the clock starts ticking.

Now, pay close attention to what I'm saying: **this trick**[13] **alone can save you up to TEN minutes**… and oh boy, if you only knew how precious becomes time during the exam… but you'll need to be prepared to benefit from it fully.

You see, there are TWO clocks on the exam. As I've just said, the first one commences its countdown when you click 'I accept'. After that you'll have 10 minutes to calm down, get a grip of yourself, concentrate, switch to full combat mode and/or read the Instructions (like clicking Help brings up the Help screen, clicking Next forwards to the next question while clicking Previous… I think, you got the picture.) So, you can skim all this self-explanatory stuff and then do whatever takes your fancy during those 10 minutes – which should be writing down some really important things that will help you crack the questions.

Now, what are they, these 'really important things'? Do you remember the following little passage from B&S (page 363)?

> If there's a particular facet of the Java language that you have difficulty remembering, try memorizing it before the exam and write it down as soon as the exam starts.[…] For example, you may have trouble remembering the list of acceptable data types in `switch` statements. If so, we recommend you memorize that information before the exam and write it down as soon as the exam starts for use in various questions.

This is it – with one correction: NOT as soon as the exam starts but *before* it. I repeat, you'll have as much as 10 minutes to do it. But do remember: you are supposed to click Start within those 10 minutes – and only then the SECOND timer will begin its 2h30m countdown…

B&S mentions the data types for `switch`. IMO, there are two more items that you might be writing down before the actual exam: the table of modifiers and an extract from the operator precedence chart. Maybe even the list of the most 'popular' ChEs + RTEs plus five major cases when an NPE gets thrown. Again, it's up to you. Anyway, we'll talk about all these in Part 2, Exam Objectives, or *Pater Noster*.

Perhaps, the only really helpful bit of info in the Instructions is the fact that you can strike out wrong answers right on the screen. Now, B&S809 says that

> You can right-click questions to cross out answers. This lets you mark answers as incorrect as you go so that you have less to think about as you read. It also helps you remember what you've eliminated when you go back to questions.

True, true – but again, with one minor correction: to do so you'll need to keep the Ctrl key pressed. So, Ctrl+right-clicking will eliminate a particular choice. Remember me mentioning a **Sleep** / **Hibernate** / **Power Off** button? Now you know you'll may need to use the keyboard after all, so watch out.

<u>Another time-saving tip</u>: to cross out incorrect answers do NOT click on the ○ or □ symbols in the list of the available options, it won't work, you'll be just wasting precious seconds; click instead right on the words / code. This particular option will immediately gray out and become stricken-through. The state is persistent, so when reviewing your answers, you'll see those grayed out lines of text/code as you have left them.

[13] Yeah, it is a bit sneaky, I admit that much; on the other hand, they themselves say it in the open: we have prepared curved balls plus a minefield for you. So I think it's just fitting; after all, in love and war, etc…

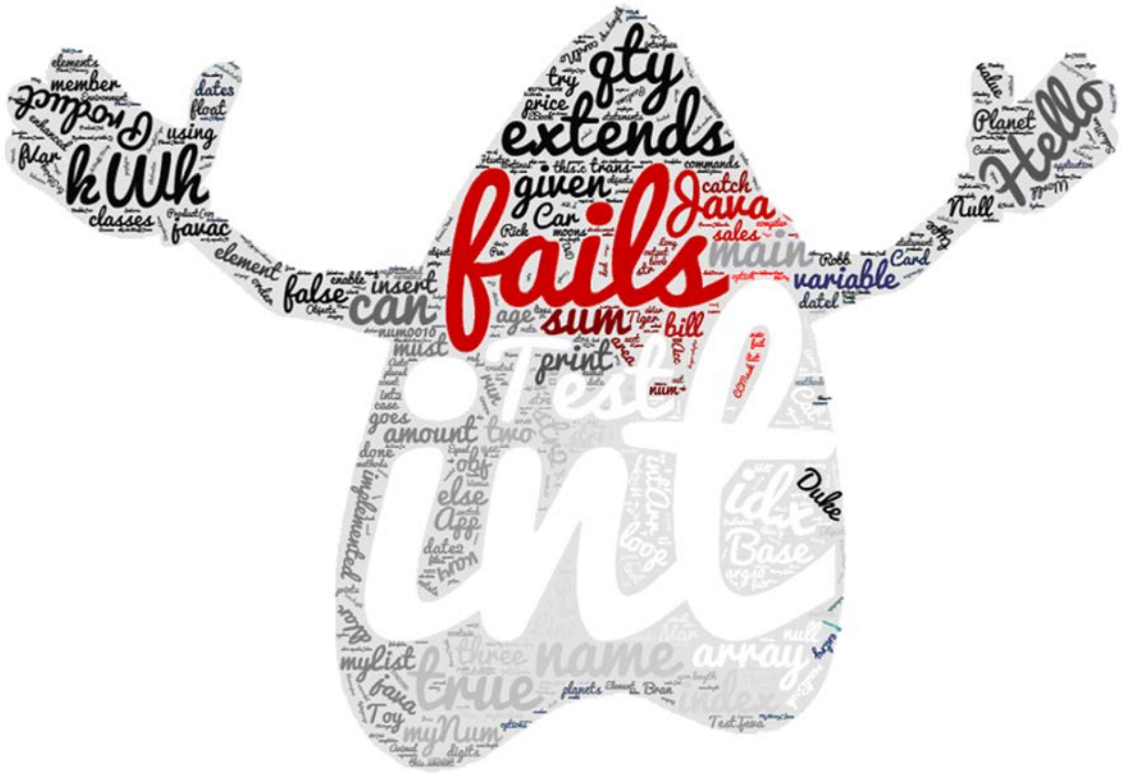
When you finish answering the questions, a table will appear on the screen containing a list of all those questions that you have marked for Review plus, supposedly, those questions that you have forgotten to answer (or ticked off insufficient number of times, like two instead of three, etc.) I say ‘supposedly’, because I did not test this function. On the other hand, the Instructions did promise it. In any case, do whatever it takes but do not leave any question unanswered or answered incompletely. Make double sure of it, the penalty is just too harsh.

One more thing of equal importance. You will need to take all the precautions so as not let anything disrupt your concentration. Both B&S and B&S809 contain many really helpful, practical suggestions in this department, like dressing in layers, using earplugs and so on. All of it solid, all of it true. Here I’d like to share with you a few of my own observations:

- If the test center permits bringing in drinks, do not use caffeinated beverages: they’ll most probably make you jumpy and even more thirsty (which did happen to me.) Bring in just plain water in a fully transparent plastic bottle with no markings (basically, just strip off the label).
- Have on you a couple of glucose tablets in case of hypoglycemia because of all this nervousness, otherwise you risk becoming listless.
- Watch out for uncapped markers. In all probability, you will be given a couple of felt tip pens (I’ve got one, blue, on the first sitting, and two, green and red, on the second.) Well, it turned out that because of all the excitement I forgot to cap my single pen – and its tip dried up. *Oops!* can write no more. So I dipped the pen into my coffee, and the trick worked. But I lost almost ten seconds – and got seriously distracted, it broke my concentration even more…
- Count the steps between the office and the test room. Sounds mysterious but tell me, can you guess why I didn’t call the exam admin by waiving at the CCTV camera with my dried-up pen? Because I counted the time it took us to move from the office (where they checked my IDs and I signed all those legal papers and so on) to the test room: almost 50 seconds. Add here the time it would have taken the admin to understand the meaning of my wild grimaces, find a fresh pen and bring it in. Best case scenario – close to a couple of minutes, wouldn’t you say? While I solved my silly problem in ten seconds. It didn’t help pass the exam, though…

Alright, back to our little story. Well, during the first sitting I hadn’t realized that I could have benefited from extra 10 minutes, so after clicking ‘I accept’ I spent thirty or so seconds scrolling through Instructions and then, right before clicking ‘Start’, I just sighed heavily and murmured ‘Hit me’.

And so it did.



Entire 1Z0-808 at a glance. What does it tell you?

Chapter

I.1 Taking the Heat

Immediately one thing became apparent – and I was utterly unprepared for it. This was of *immense* importance, but even more serious were consequences. Naturally, you may be of a completely different opinion; I am simply sharing my experience with you, that's all.

Now, what is it, this hellishly important thing? Take a deep breath, then read slowly:

THE CODE IS PRETTY-TYPED

All of it. Every frigging LOC[14]. And what are the consequences of this momentous discovery?

Since the code is properly formatted, we can stop counting curly braces… Guys! *We can stop counting those pesky things and waste time on them*!

I dunno, maybe it's just me but before the exam I took for granted that one of the possible traps could be a missing or misplaced brace. Besides, a string of them may get in the way obscuring the meaning, the clarity of code. "Get used to those horrible braces", warns the study guide by Kathy Sierra and Bert Bates referring, for example, to this curly brace parade on the same line: }}}}}. Or take B&S (p.33):

> Identifying blocks needs to be second nature for the exam. The good news is that there are lots of code examples to practice on. You can look at any code example in this book on any topic and match up braces.

But now all the initialization blocks, all the method and constructor bodies and so on can be grasped at a glance. I suspect that the Oracle cert team simply got tired of all the complaints (in a couple of days after successfully passing the exam you'll get an email asking you to participate in a survey) and finally gave in making all code pretty-typed.

I, however, couldn't believe my eyes and continued, stubbornly and stupidly, match the braces just *wasting time*. A lot of it. On the second sitting it was altogether different: I simply took them all in at a glance, just making sure that all the method bodies, initers, etc. are properly enclosed. But no counting this time! That was a *major* time saver.

Alas, this is perhaps the only piece of good news about the code being pretty-typed on the exam because when you properly format your LOCs, with most of the closing braces on fresh lines, with indentations and all, the listing becomes *lo-ong*.

Combined with the fact that the font size is rather big (I think it is around 18 points), it means that many questions – not to mention answer options! – cannot fit on a single screen thus forcing you to scroll up and down incessantly and desperately. Damn! That threw me out of synch[15] with the code, I was out of the groove – and got real nervous.

And what happens when you are nervous? Some people (me included) start second-guessing themselves. Doubt sneaks in and fogs your mind, you become tense, edgy… The vicious circle at its worst.

[14] This, however, I fully realized only near the end of the exam.

[15] To tell the truth, I'm fairly surprised that none of the software-based mock up exams that I'm familiar with (such as Kaplan's, Enthuware, etc.) simulate this particular feature. After all, it is harder to crack problems when you can't see *entire* code and *all* available options…

So I committed another hideous error: after checking off the option(s) I thought to be correct, I began to re-read the code looking for previously unnoticed traps. Each and every time.

That was most foolish of me. But wait, there was another deadly mistake: being full of doubts, I started marking up *too many* questions for Review…

Stop here for a sec and think what reviews are meant for. To check your answers in case you missed something, right? Like re-running calculations for nested loops, for example, just to see if you got everything correctly the first time. Or to take another look at the code for which you couldn't decide between two options, something like that[16]. And when you mark almost every other question… What would *that* mean? It would mean that you'll be practically taking the same exam once again. Still within the allotted time. Which is already running out because of multiple re-readings…

When I went through all the questions, I had only 15 minutes left, and the Review list contained some thirty items. And I didn't even remember which were the most difficult ones; those that really needed my attention…

Another psychological trap I let myself in is that I was expecting nasty tricks in every single question. The thing is, many questions don't even try to deceive you; they are straight as an arrow. Tell me, can you spot anything even remotely suspicious about this one (something to this tune *is* on the real exam):

Given in the file HiThere.java**:**

```
class HiThere {
    public static void main(String[] args) {
        System.out.print("Hi there" + args[0]);
    }
}
```

Which set of commands prints Hi there! **in the console?**

A. javac HiThere
   java HiThere !
B. javac HiThere.java
   java HiThere !
C. javac HiThere.java !
   java HiThere
D. javac HiThere.java
   java HiThere.class !

I didn't see anything out of the ordinary, either. But spent on this one a few minutes instead of few seconds stubbornly resisting to trust my own eyes and instincts. All because I was wound up for devilishly clever trickery – where there was none…

We have already covered the make up of the exam questions according to their basic functions (the 'Core', 'Advanced' and 'Guinea Pig' types). Another possible classification is based on the inclination of the question to deceive you, to run the proverbial red herring across your path. Viewed from this angle, a question can be:

- a straight arrow,
- a curved ball to throw you off balance, and
- a caltrop that is meant to cripple you.

The class **HiThere** is, obviously, of the first type. Let's have a curved ball:

[16] B&S mentions some other techniques that you can use during Review like applying context clues from other questions, etc.

```
List<Integer> l = new ArrayList<>();
l.add(42);
l.add(null);
for (int i = 0; i<l.size();i++) System.out.print(l.get(i));    // line 1
for (Object i : l) System.out.print(i);                         // line 2
for (int i : l) System.out.print(i);                            // line 3
```

Does the code compile? If it does, what is the result? If not, what is the reason for the comperr? And if the code throws an RTE, what is the reason for that?

The answer is this: the code does compile, both line 1 and line 2 print 42null each, and line3 prints 42 then throws an NPE because we attempt to unbox an **Integer** by implicitly calling **Integer.intValue()** on a `null` object.

Please don't be alarmed; in all probability, you will not get such a question on the exam: it is more of the OCP realm. But there are at least three questions in our OCA 1Z0-808 that relate to the behavior of `null`, which includes several scenarios – and this is your curved ball: you'll be needing to keep track of a number of things at the same time. We'll cover these scenarios when discussing NPE.

How about meeting a simple yet nasty caltrop? Here you go:

Given:

```
int a = 0;
String str = new String("Yell");
str = str.replace("Yell", "Hell");
str = str.insert(4,"o");
if ("Hello" == str && a++ == 0) str = null;
System.out.println(a);
```

What is the result?

A. The code prints 1
B. The code prints 0
C. Compilation fails
D. The code throws an exception at run time

The correct answer is C: compilation fails.

The exam knows that we got used to see **insert()** rather often – and catches us off guard[17].

And now I am going to confide to you THE mistake I made. The real one. All-embracing. The mother of all mistakes. Ready?

I didn't read the questions. Hah! how about that? Bet you didn't see it coming. Me neither :(

But fact is a fact is a fact. I did not read the textual part properly, that is, carefully. Was too anxious to get to the ʻmeat', the code itself. On my way home from the exam I was, naturally, thinking about the questions while writing them down from memory – and suddenly saw that a lot of times I tried to mark as correct the lines that would compile while the question must have been asking for the lines that wouldn't, and vice versa. "Ah, so that's why I couldn't get the right number of options so many times! The question asked for two but I found three… And since the GUI wouldn't allow me to checkmark so many, I relied on my blind luck and did it for only two. Both incorrect ones…"

Line from "The Rainmaker". "*You must be stupid, stupid, stupid!*"…

[17] **insert(**int offset, String str**)** belongs to the class **StringBuilder** rather than **String**.

After failing the first sitting, after spending weeks recalling and reconstructing the exam questions, analyzing my own behavior and realizing what I have done, a set of most simple rules formed in my mind:

- ❑ Don’t jump to the code. Read – the question – first. *Force yourself to read it*.
- ❑ Give the problem your best shot at the very first pass.
- ❑ Do it unhurriedly and never let yourself re-read what you have already read carefully.
- ❑ Take the code at its face value. Not every snippet is meant to deceive you.
- ❑ No cherry picking; do the questions as they come in.
- ❑ Treat your Review marks as gold.
- ❑ Trust your gut feelings.
- ❑ Resist the temptation to redo calculations right away, leave it for the Review.

In hindsight, it all became so painfully apparent: I did not clearly think my approach through *beforehand*. Despite the fact that it is so uncomplicated… Take, for example, the ‘Review marks’ rule. If certain questions are too tough for you, if you are barely sure of what the code is doing, marking the problems for Review won’t help much – especially when you mark too many of them: you simply won’t have time to do a decent job. On the other hand, when you go through the questions like an ice-breaker, without dwelling on them too much, when you completely unleash your instincts as a coder and let them work for you… then you will have tons of time left and, therefore, stand a good chance of catching mistakes and disarming previously unnoticed traps on the Review. Not to mention gathering context clues during the initial pass…

But enough of this fountain of words, let’s have hard facts. Here’s the timeline for my second sitting:

Initial pass – 77 questions – 40 minutes. Yes, I did say forty. Not 2h15m like before. ("Wow… I’m sorta flying through all this like a Greek god or something…")

1st Review – a dozen of marked up questions (half of them loops, nested or otherwise) – 18 minutes. ("Damn, they were right, after all. It *is* kiddies stuff…") Swiftly found and corrected two silly mistakes.

1st *complete* Review – all 77 questions – 35 minutes. ("Day-dreaming on a golden cloud…") After recalling a puzzle in the Enthuware, changed one of the answers.

2nd complete Review – ditto – 22 minutes. ("Bo-oring…") No mistakes spotted this time.

Seeing that I still have a good half an hour left and no more interesting things to do, I decided to call it a day.

My final conclusions after two sittings:

All things considered, I must say that the most important, overarching rule that leads to success is also the simplest and most logical one:

> Read the question first, then options and then code *carefully*. Subvocalize, even.

For example, take a look at this:

```
int[][] arr2D = new int[][] { { 0, 1, 2, 3 } { 4, 5 } };
```

Monospaced fonts are notoriously hard on eyes, particularly when the font size is unusually large – as it is on the actual exam. It is surprisingly easy to overlook a missing comma in the above LOC *especially* when you're tired after, say, fifty or so questions and time is already pressing because:

a) you have, most probably, marked up a good dozen of questions for Review (which means you'll need more time for them), and

b) there may be some frustratingly time-consuming loop(s) ahead…

Another important guideline: if you see more correct answers than the question asks for, and you still can't find what is wrong with the options you're ready to tick off, review <u>the most ʿobvious' choice</u> extra carefully: it just may contain a trap.

Never assume that the exam creators made a mistake.[18]

I also noticed this: **weird looking, overly complex code**[19] **is most likely to be booby-trapped** on a fairly *simple concept* such as (listed in the order of prevalence):

– out-of-scope vars;
– invalid syntax;
– unreachable stats.

[18] Unbelievable as it may sound, they do have a typo in one of their questions (which has to do with the class **LocalDate**): it lists a misspelled `DateParseExcpetion` as one of the options. I met this typo both times…

[19] Particularly involving loops or post-decrement/post-increment notation.