Promise Note #1, Paid in Full

public static void main(String[] args) -- the line you typed on faith in Chapter 1 -- decoded one word at a time, now that every piece of it is something you already know.

Automation Foundations · ashishautomationlabs.store

On page one of this book you typed a line you did not understand, and were asked to trust it and move on:

```
public static void main(String[] args)
```

That was Promise Note #1, the first debt this book took on. You have quietly earned it back since -- `String[]` is an array of text (Chapter 5), a class holds behavior (Chapter 7), and you have written methods already without stopping to study what one really is. Read it now, one word at a time.

- `public` -- the access level, the opposite of the `private` you met in

Chapter 7. It means anything may call this. The JVM lives outside your class and has to call it, so it must be public.

- `static` -- it belongs to the class, not to an object. This one matters: when

your program starts, no objects exist yet, so the JVM has nothing to call a method on. `static` is what makes `main` callable out of thin air.

- `void` -- it returns nothing to the JVM. The program's success or failure is

not reported this way.

- `main` -- the name the JVM looks for. A fixed convention, not a keyword -- this

exact spelling is where the JVM begins.

- `String[] args` -- an array of text, holding whatever arguments were typed on

the command line when the program was launched.

That last part is easy to prove. This program does nothing but report its own arguments:

```
public class MainArgs {
    public static void main(String[] args) {
        System.out.println("Arguments received: " + args.length);
        for (int i = 0; i < args.length; i++) {
            System.out.println("  args[" + i + "] = " + args[i]);
        }
```

```
    }
}
```

Run with `java MainArgs.java uat chrome` and the array is filled:

```
Arguments received: 2
  args[0] = uat
  args[1] = chrome
```

Run it with no arguments and the array is empty, with a length of zero:

```
Arguments received: 0
```

Note that it is empty, not `null` -- so a loop over it simply runs zero times rather than crashing. Every word in that line is a concept you have used and understood. If an interviewer asks you to explain `public static void main`, you can walk through all five parts and say why each one has to be what it is -- along with the rest of this chapter's common ground: the difference between a parameter and an argument (slot versus value passed in), whether Java is pass-by-value or pass-by-reference (always pass-by-value; for objects the value copied is the reference, which is why a method can change the object but cannot repoint your variable), what overloading is (same name, different parameter lists, chosen at the call), and why a static method cannot use instance fields (there is no object, so those fields do not exist for it).

Promise Note #1 -- opened in Chapter 1, paid here, in full. The remaining open note is #3: the Map and Set containers, which need the equality and `hashCode` work you completed in Chapter 7.

Every idea in this chapter looks simple in isolation. The next question is what happens when a method is used carelessly -- and there are two ways to do that, both silent.