Tech on the Toilet

Google

Arrange Your Code to Communicate Data Flow

by Sebastian Dörner

We often read code linearly, from one line to the next. To make code easier to understand and to reduce cognitive load for your readers, make sure that adjacent lines of code are coherent. One way to achieve this is to **order your lines of code to match the data flow inside your method**:

```
fun getSandwich(
    bread: Bread, pasture: Pasture
): Sandwich {
  // This alternates between milk- and
  // bread-related code.
  val cow = pasture.getCow()
  val slicedBread = bread.slice()
  val milk = cow.getMilk()

  val toast = toastBread(slicedBread)
  val cheese = makeCheese(milk)

  return Sandwich(cheese, toast)
}
```

```
fun getSandwich(
    bread: Bread, pasture: Pasture
): Sandwich {
  // Linear flow from cow to milk to cheese.
  val cow = pasture.getCow()
  val milk = cow.getMilk()
  val cheese = makeCheese(milk)

  // Linear flow from bread to slicedBread to
  // toast.
  val slicedBread = bread.slice()
  val toast = toastBread(slicedBread)

  return Sandwich(cheese, toast)
}
```

To visually emphasize the grouping of related lines, you can add a blank line between each code block.

Often you can further improve readability by extracting a method, e.g., by extracting the first 3 lines of the function on the above right into a `getCheese` method. However, **in some scenarios, extracting a method isn't possible or helpful**, e.g., if data is used a second time for logging. **If you order the lines to match the data flow, you can still increase code clarity:**

```
fun getSandwich(bread: Bread, pasture: Pasture): Sandwich {
  // Both milk and cheese are used below, so this can't easily be extracted into a method.
  val cow = pasture.getCow()
  val milk = cow.getMilk()
  reportFatContentToStreamz(cow.age, milk)
  val cheese = makeCheese(milk)

  val slicedBread = bread.slice()
  val toast = toastBread(slicedBread)

  logWarningIfAnyExpired(bread, toast, milk, cheese)
  return Sandwich(cheese, toast)
}
```

It isn't always possible to group variables *perfectly* if you have more complicated data flows, but even *incremental changes* in this direction improve the readability of your code. A good starting point is to declare your variables as close to the first use as possible.

More information and archives: `testing.googleblog.com`