Scala Puzzlers
Excerpt
Scala offers several convenient ways to initialize multiple variables. Sometimes, this can lead to unexpected surprises.

What is the result of executing the following code in the REPL?

```scala
var MONTH = 12; var DAY = 24
var (HOUR, MINUTE, SECOND) = (12, 0, 0)
```

**Possibilities**

1. Prints:
   
   ```
   MONTH: Int = 12
   DAY: Int = 24
   HOUR: Int = 12
   MINUTE: Int = 0
   SECOND: Int = 0
   ```

2. Both statements fail to compile.

3. The first statement prints:

   ```
   MONTH: Int = 12
   DAY: Int = 24
   ```

   and the second throws a runtime exception.
4. The first statement prints:

```scala
MONTH: Int = 12
DAY: Int = 24
```

and the second fails to compile.

**Explanation**

You might recall something about uppercase variables and constant values and wonder whether either line compiles. As it happens, the first line compiles fine; it’s the second statement that fails to compile. The correct answer is number 4:

```scala
scala> var MONTH = 12; var DAY = 24
MONTH: Int = 12
DAY: Int = 24

scala> var (HOUR, MINUTE, SECOND) = (12, 0, 0)
<console>:11: error: not found: value HOUR
     var (HOUR, MINUTE, SECOND) = (12, 0, 0)
         ^

<console>:11: error: not found: value MINUTE
     var (HOUR, MINUTE, SECOND) = (12, 0, 0)
         ^

<console>:11: error: not found: value SECOND
     var (HOUR, MINUTE, SECOND) = (12, 0, 0)
         ^
```

Scala will happily allow you to use an uppercase variable name for plain, single-value assignments of `val`s and `var`s, as in the case of `MONTH` and `DAY`. However, as the second statement demonstrates, uppercase variable names are tricky in multiple-variable assignments.

This trickiness arises because multiple-variable assignments are based on pattern matching, and within a pattern match, variables starting with an uppercase letter take on a special meaning: they are *stable identifiers*.

Stable identifiers are intended for matching against constants:
```scala
scala> final val TheAnswer = 42
scala> def checkGuess(guess: Int) = guess match {
    case TheAnswer => "Your guess is correct"
    case _ => "Try again"
}
scala> checkGuess(21)
res8: String = Try again
scala> checkGuess(42)
res9: String = Your guess is correct
```

Lowercase variables, by contrast, define *variable patterns*, which cause values to be assigned:

```scala
scala> var (hour, minute, second) = (12, 0, 0)
hour: Int = 12
minute: Int = 0
second: Int = 0
```

In the case of our code example, you are not carrying out a variable assignment as intended, therefore, but a match against constant values.

**Discussion**

If you are trying to use uppercase variable names that, by extreme coincidence, happen to match values that are in scope (which could happen with common names in a large program), the pattern match will compile successfully, and either succeed or fail depending on whether the values match:

```scala
val HOUR = 12; val MINUTE, SECOND = 0;
scala> var (HOUR, MINUTE, SECOND) = (12, 0, 0)
scala.MatchError: (12,0,0) (of class scala.Tuple3)
...
```

Note that, even in the first case where the match is successful, no variables are actually assigned: stable identifiers are never assigned a value during a
pattern match, by definition. In short, at best nothing happens, otherwise you get an exception at runtime—neither of which was intended.

Lowercase variables can also be treated as stable identifiers by enclosing them in backticks. In that case, they must be `vals`, since we are treating them as constants.

```scala
final val theAnswer = 42
def checkGuess(guess: Int) = guess match {
  case `theAnswer` => "Your guess is correct"
  case _ => "Try again"
}
```

```scala
scala> checkGuess(42)
res0: String = Your guess is correct
```

```scala
var theAnswer: Int = 42 // not a val, and not final either
```

```scala
scala> def checkGuess(guess: Int) = guess match {
  case `theAnswer` => "Your guess is correct"
  case _ => "Try again"
}
<console>:9: error: stable identifier required, but theAnswer found.
```

It’s unlikely to come as a surprise that uppercase names for `vars` are not considered Scala best practice: use lowercase names for `vars` (better still, avoid them completely!), and uppercase names for constants. As described in *The Scala Language Specification*, constants should also be declared `final`.¹ This prevents subclasses from overriding them, and has an additional performance benefit in that the compiler can inline them.

¹Odersky, *The Scala Language Specification*, Section 4.1. [Ode08]