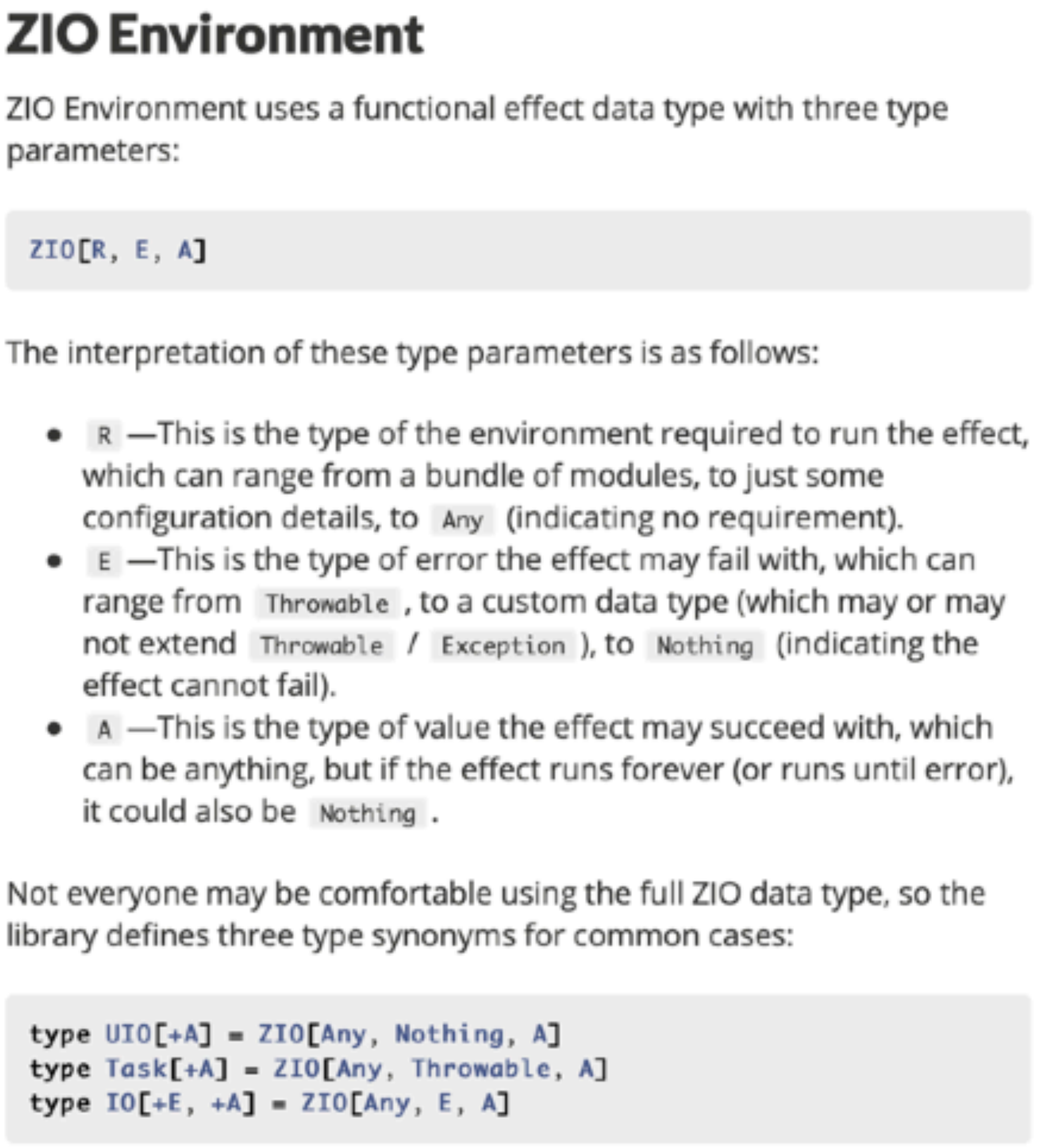
IO Monad - ZIO



ZIO Environment

ZIO Environment uses a functional effect data type with three type parameters:

```
ZIO[R, E, A]
```

The interpretation of these type parameters is as follows:

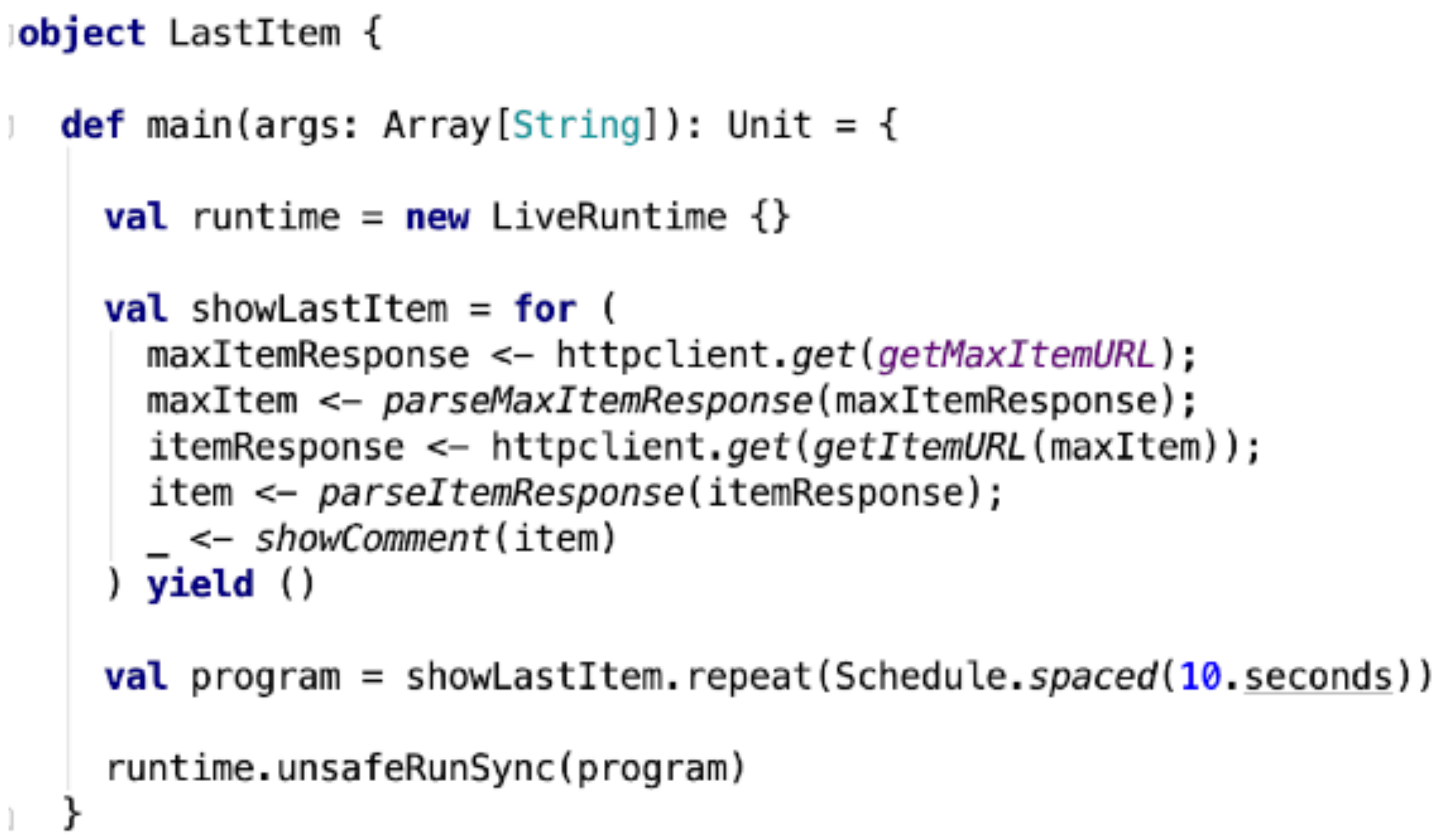
- `R` —This is the type of the environment required to run the effect, which can range from a bundle of modules, to just some configuration details, to `Any` (indicating no requirement).
- `E` —This is the type of error the effect may fail with, which can range from `Throwable` , to a custom data type (which may or may not extend `Throwable` / `Exception`), to `Nothing` (indicating the effect cannot fail).
- `A` —This is the type of value the effect may succeed with, which can be anything, but if the effect runs forever (or runs until error), it could also be `Nothing` .

Not everyone may be comfortable using the full ZIO data type, so the library defines three type synonyms for common cases:

```
type UIO[+A] = ZIO[Any, Nothing, A]
type Task[+A] = ZIO[Any, Throwable, A]
type IO[+E, +A] = ZIO[Any, E, A]
```

IO Monad – ZIO

Example ZIO program



```
object LastItem {

  def main(args: Array[String]): Unit = {

    val runtime = new LiveRuntime {}

    val showLastItem = for (
      maxItemResponse <- httpclient.get(getMaxItemURL);
      maxItem <- parseMaxItemResponse(maxItemResponse);
      itemResponse <- httpclient.get(getItemURL(maxItem));
      item <- parseItemResponse(itemResponse);
      _ <- showComment(item)
    ) yield ()

    val program = showLastItem.repeat(Schedule.spaced(10.seconds))

    runtime.unsafeRunSync(program)
  }
```