

# **Mastering Interruptions**



# Mastering Interruptions

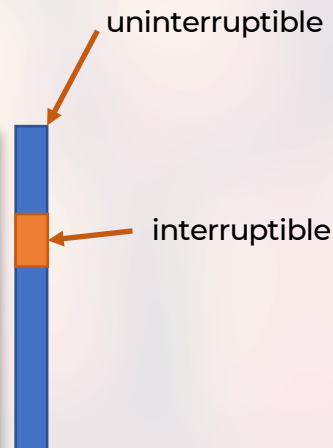
ZIOs can be marked uninterruptible

```
val atomicPayment = ZIO.uninterruptible(paymentSystem)
```

Define pinpoint interruptible regions by using the restorer

- everything wrapped in the restore call has *the same flag as before*
- everything else is not interruptible

```
val authFlow = ZIO.uninterruptibleMask { restore =>
  for {
    pw <- restore(inputPassword)
    verification <- verifyPassword(pw)
    _ <- if (verification) ZIO.succeed("Auth success!").debugThread
      else ZIO.succeed("Auth failed.").debugThread
  } yield ()
}
```



The restorer is the local opposite of uninterruptible

- can be called as many times as we like

**ZIO rocks**

