Extensibility: traits
data & behavior

Algorithms + Data Structures = Programs

Niklaus Wirth
abstract
t:
no data or behavior is available

abstract class
all methods are abstract

trait
usually defines behavior

case class
usually defines data

class
defines data & behavior

object
all data & behavior available

abstract method def
no implementation

method def
implementation

method
callable

disclaimer: This diagram doesn’t capture all the nuances of the abstract / concrete spectrum in Scala. For example, it’s possible to have a trait that’s more abstract than an abstract class.
trait Logger {
  def log(message: String): Unit

  def infoTag = "[info]"
  def warningTag = "[warning]"
  def errorTag = "[error]"

  def info(message: String) = log(s"$infoTag $message")
  def warning(message: String) = log(s"$warningTag $message")
  def error(message: String) = log(s"$errorTag $message")
}

class ConsoleLogger extends Logger {
  override def log(message: String) {
    println(message)
  }
}

val logger = new ConsoleLogger()
trait Logger {
  def log(message: String): Unit

  def infoTag = "[info]"
  def warningTag = "[warning]"
  def errorTag = "[error]"

  def info(message: String) = log(s"$infoTag $message")
  def warning(message: String) = log(s"$warningTag $message")
  def error(message: String) = log(s"$errorTag $message")
}

class ConsoleLogger extends Logger {
  override def log(message: String) {
    println(message)
  }
}

val logger = new ConsoleLogger()
Pattern: Thin Interface
trait Logger {
  info
  warning
  error
}

class ConsoleLogger extends Logger {
  log
}

val logger = new ConsoleLogger()
Pattern: Mixin
trait Logger {
  info
  warning
  error
  log
}

class ConsoleLogger extends Logger {
  log
}

val logger = new ConsoleLogger()
trait Timestamping extends Logger {
  def timestamp = new java.util.Date()
  override def log(message: String) = s"[\$timestamp] \$message"
}

object logger extends ConsoleLogger with Timestamping
trait Timestamping extends Logger {
  def timestamp = new java.util.Date()
  override def log(message: String) =
    super.log(s"[$timestamp] $message")
}

object logger extends ConsoleLogger with Timestamping
trait Timestamping extends Logger {
  def timestamp = new java.util.Date()
  abstract override def log(message: String) = 
    super.log(s"[$timestamp] $message")
}

class ConsoleLogger extends Logger {
  // all data & behavior is available
}

object logger extends ConsoleLogger with Timestamping

trait Logger {
  def info = log
  def warning = log
  def error = log
}

abstract no data or behavior is available
Pattern: Stacked Traits
Pattern: Stacked Traits

```
object logger extends ConsoleLogger with Timestamping with Lowercasing
```

```
trait Timestamping extends Logger {
    log
}
```

```
trait Lowercasing extends Logger {
    log
}
```

```
trait Logger {
    info
    warning
    error
}
```

```
class ConsoleLogger extends Logger {
    log
}
```
Is this a DSL?

- Everyone speaks with an English accent.
- Verity multiplies every number by 10.
- Lambert divides every number by 3.
- Never say “mattress” to Lambert; say “dog kennel” instead.

So, if Verity wants to say, “This mattress costs £90” to Lambert, it will come out as “This dog kennel costs £900” (in an English accent).