# Type Classes

# Type Classes

A pattern of organizing code that

- decouples definition from implementations
- allows multiple implementations for the same type
- brings back (or improves) expressiveness

Part 1: type class definition

```
trait MyTypeClass[T]:
  def action(value: T): String
```

Part 2: type class instances

```
given intInstance: MyTypeClass[Int] with
  override def action(value: Int) = value.toString
// same for other types you want to support
```

# Type Classes

Part 1: type class definition

```
trait MyTypeClass[T]:
  def action(value: T): String
```

Part 2: type class instances

```
given intInstance: MyTypeClass[Int] with
  override def action(value: Int) = value.toString
// same for other types you want to support
```

Part 3: front-facing API

```
object MyTypeClass:
  def action[T](value: T)(using instance: MyTypeClass[T]) = instance.action(value)
  def apply[T](using instance: MyTypeClass[T]): MyTypeClass[T] = instance
```

Part 4: expressiveness with extension methods

```
object MyTypeClassSyntax:
  extension [T](value: T)
    def action(using instance: MyTypeClass[T]): String = instance.action(value)
```

# Scala rocks