Ruby Mapping for Enumerations

Previous Next

Ruby does not have an enumerated type, so a Slice enumeration is emulated using a Ruby class: the name of the Slice enumeration becomes the name of the Ruby class; for each enumerator, the class contains a constant with the same name as the enumerator. For example:

Slice

```
enum Fruit { Apple, Pear, Orange }
```

The generated Ruby class looks as follows:

Ruby

```
class Fruit
    include Comparable

    Apple = # ...
    Pear = # ...
    Orange = # ...

    def Fruit.from_int(val)

    def to_i

    def to_s

    def <=>(other)

    def hash

    # ...
end
```

The compiler generates a class constant for each enumerator that holds a corresponding instance of `Fruit`. The `from_int` class method returns an instance given its Slice value, while `to_i` returns the Slice value of an enumerator and `to_s` returns its Slice identifier.

Given the above definitions, we can use enumerated values as follows:

Ruby

```
f1 = Fruit::Apple
f2 = Fruit::Orange

if f1 == Fruit::Apple   # Compare for equality
    # ...

if f1 < f2              # Compare two enums
    # ...

case f2
when Fruit::Orange
    puts "found Orange"
else
    puts "found #{f2.to_s}"
end
```

Comparison operators are available as a result of including `Comparable`, which means a program can compare enumerators according to their Slice values. Note that, when using custom enumerator values, the order of enumerators by their Slice values may not match their order of declaration.

Suppose we modify the Slice definition to include a custom enumerator value:

Slice

```
enum Fruit { Apple, Pear = 3, Orange }
```

We can use `from_int` to examine the Slice values of the enumerators:

Ruby

```
Fruit::from_int(0)  # Apple
Fruit::from_int(1)  # nil
Fruit::from_int(3)  # Pear
Fruit::from_int(4)  # Orange
```

Back to Top ^

See Also

- Enumerations
- Ruby Mapping for Identifiers
- Ruby Mapping for Modules
- Ruby Mapping for Built-In Types
- Ruby Mapping for Structures
- Ruby Mapping for Sequences
- Ruby Mapping for Dictionaries
- Ruby Mapping for Constants
- Ruby Mapping for Exceptions
- Ruby Mapping for Interfaces
- Ruby Mapping for Operations

Previous Next