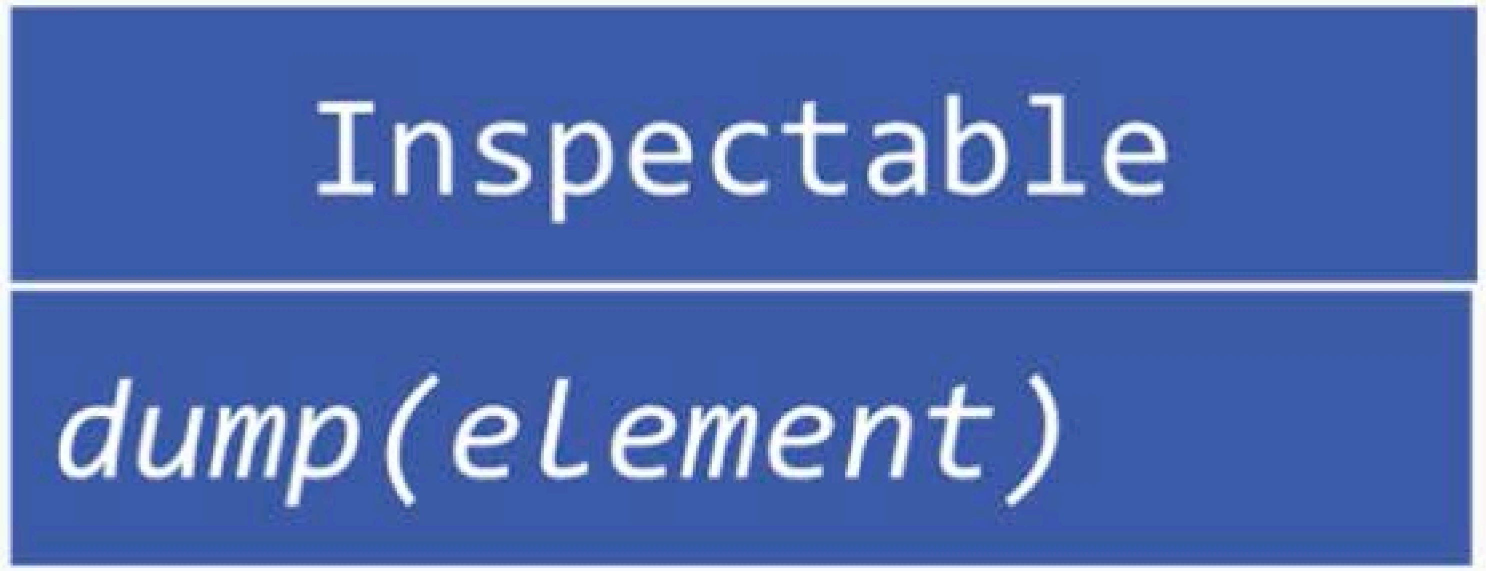
Implementing Protocols

In this Video, we are going to take a look at…

- Implementing Protocols for existing types
- Defining a custom type and implementing a Protocol for it

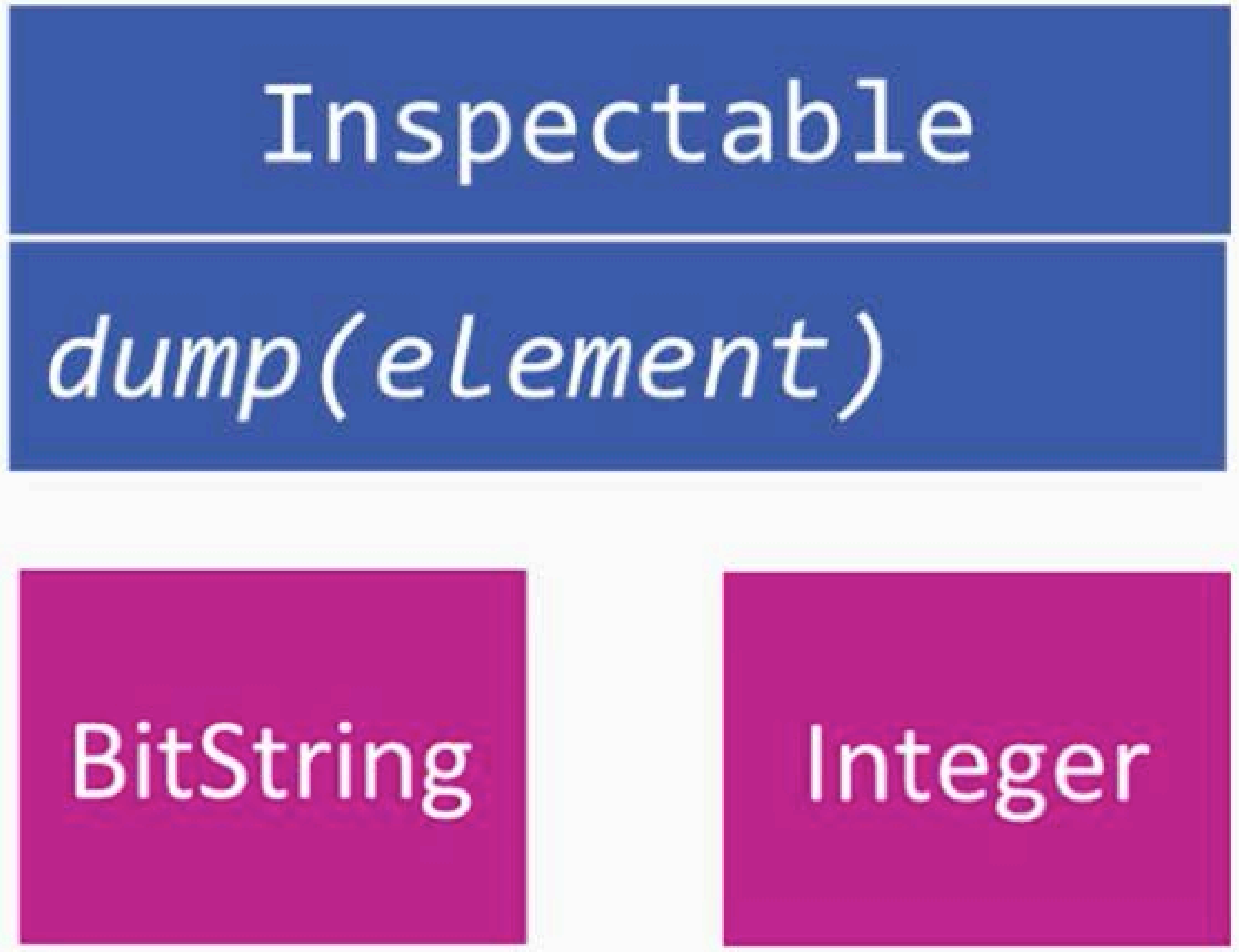
Inspectable

Exposes a dump function for logging type information.



Inspectable

Exposes a dump function for logging type information.



Inspectable

Exposes a dump function for logging type information.

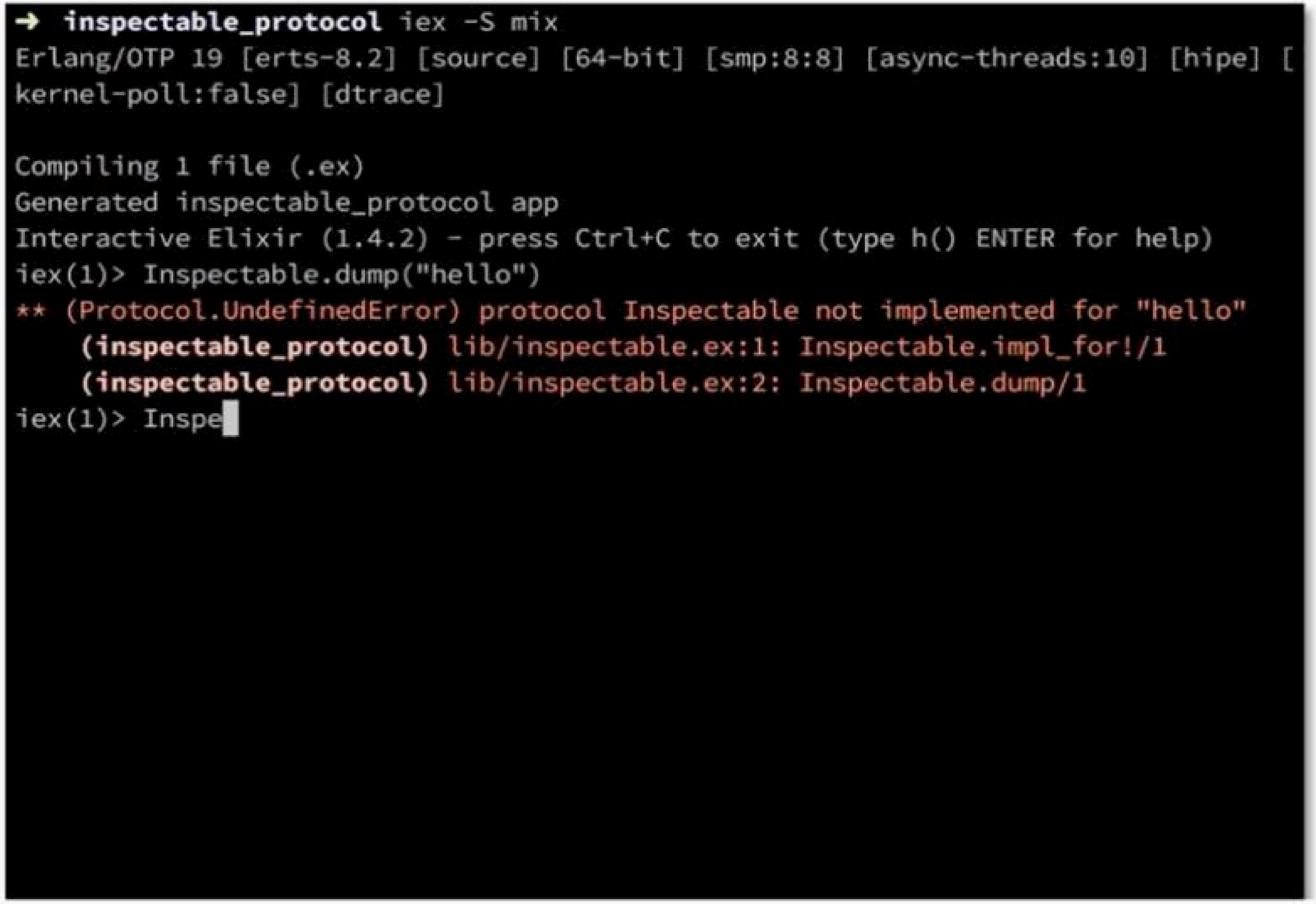
Implementing the Protocol for Integer and String.

```
➜  inspectable_protocol
```

Inspectable

Exposes a dump function for logging type information.

Implementing the Protocol for Integer and String.

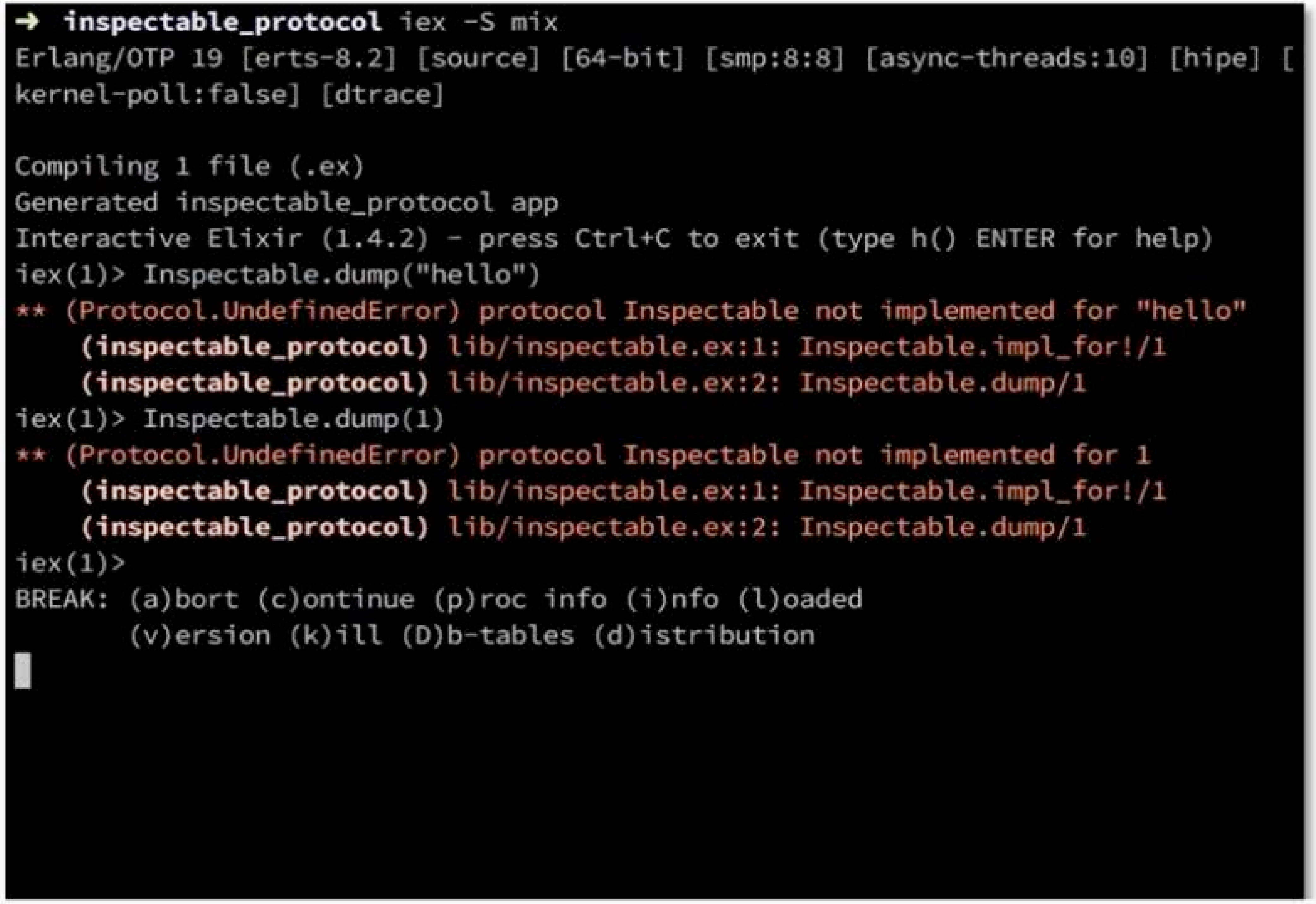


```
➜ inspectable_protocol iex -S mix
Erlang/OTP 19 [erts-8.2] [source] [64-bit] [smp:8:8] [async-threads:10] [hipe] [
kernel-poll:false] [dtrace]

Compiling 1 file (.ex)
Generated inspectable_protocol app
Interactive Elixir (1.4.2) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Inspectable.dump("hello")
** (Protocol.UndefinedError) protocol Inspectable not implemented for "hello"
    (inspectable_protocol) lib/inspectable.ex:1: Inspectable.impl_for!/1
    (inspectable_protocol) lib/inspectable.ex:2: Inspectable.dump/1
iex(1)> Inspe
```

Inspectable

Exposes a dump function for logging type information.

Implementing the Protocol for Integer and String.

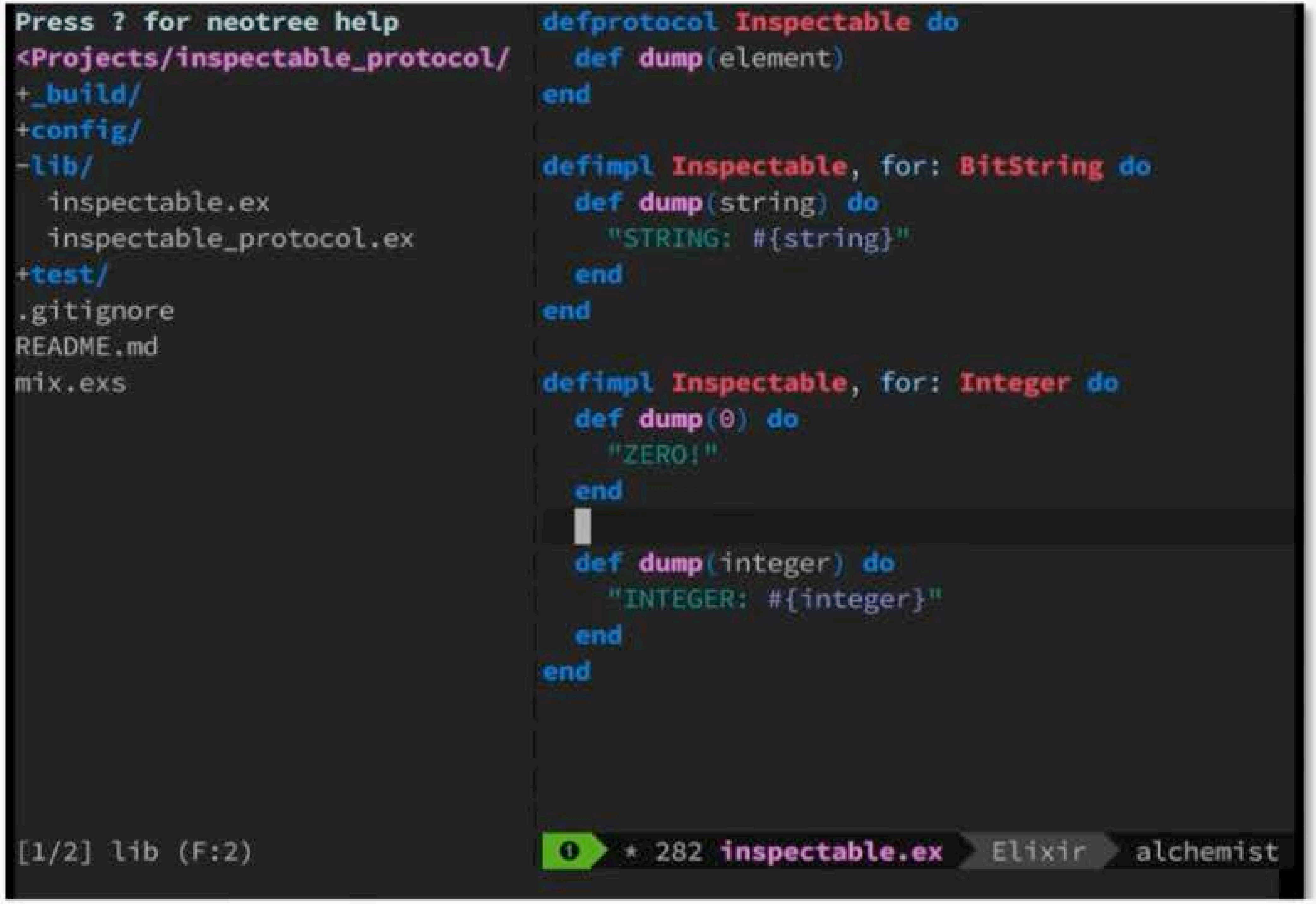


```
➜ inspectable_protocol iex -S mix
Erlang/OTP 19 [erts-8.2] [source] [64-bit] [smp:8:8] [async-threads:10] [hipe] [
kernel-poll:false] [dtrace]

Compiling 1 file (.ex)
Generated inspectable_protocol app
Interactive Elixir (1.4.2) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Inspectable.dump("hello")
** (Protocol.UndefinedError) protocol Inspectable not implemented for "hello"
    (inspectable_protocol) lib/inspectable.ex:1: Inspectable.impl_for!/1
    (inspectable_protocol) lib/inspectable.ex:2: Inspectable.dump/1
iex(1)> Inspectable.dump(1)
** (Protocol.UndefinedError) protocol Inspectable not implemented for 1
    (inspectable_protocol) lib/inspectable.ex:1: Inspectable.impl_for!/1
    (inspectable_protocol) lib/inspectable.ex:2: Inspectable.dump/1
iex(1)>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
```

Inspectable

Exposes a dump function for logging type information.

Implementing the Protocol for Integer and String.



```
Press ? for neotree help
<Projects/inspectable_protocol/
+_build/
+config/
-lib/
  inspectable.ex
  inspectable_protocol.ex
+test/
.gitignore
README.md
mix.exs

[1/2] lib (F:2)
```

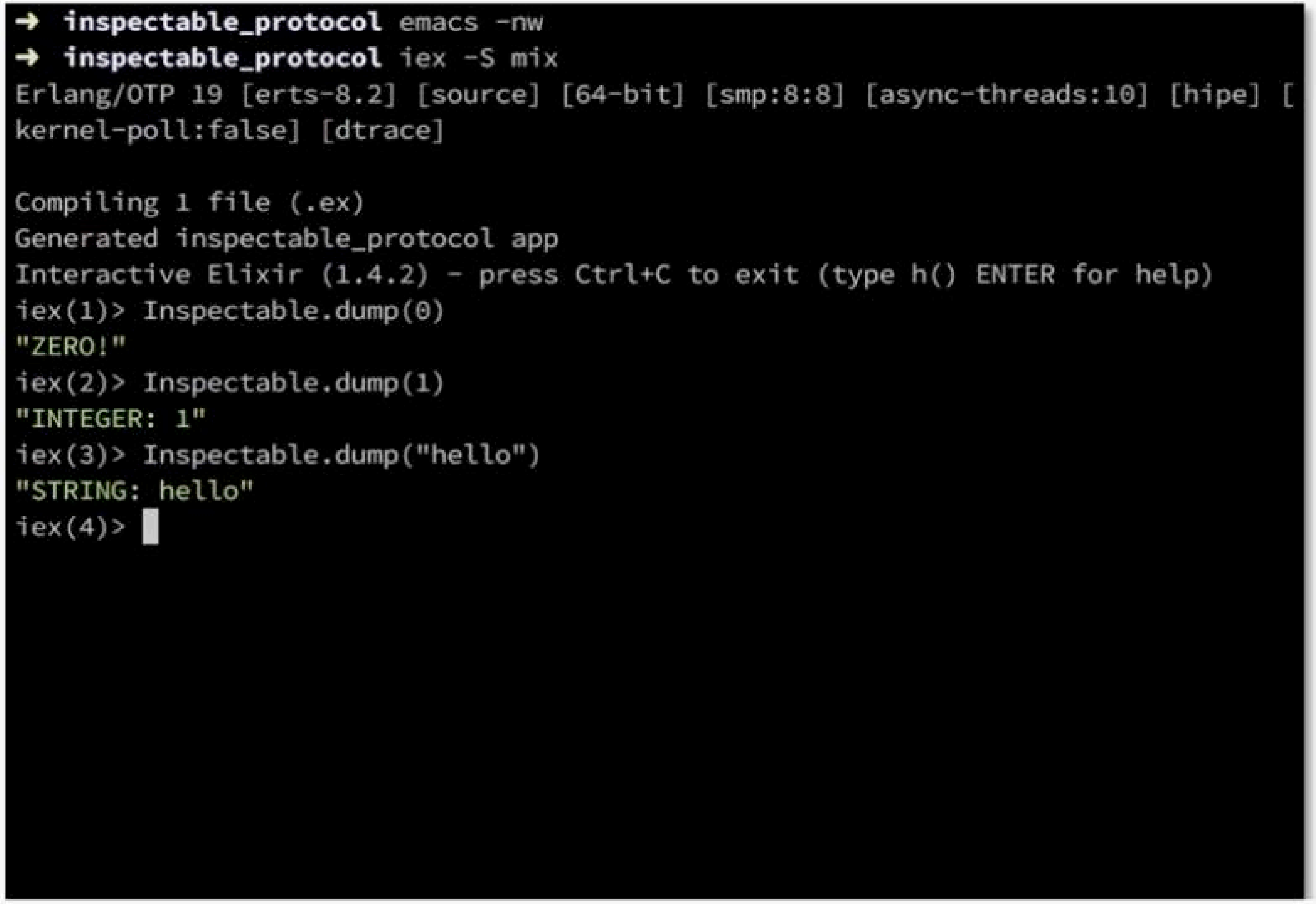
```elixir
defprotocol Inspectable do
  def dump(element)
end

defimpl Inspectable, for: BitString do
  def dump(string) do
    "STRING: #{string}"
  end
end

defimpl Inspectable, for: Integer do
  def dump(0) do
    "ZERO!"
  end

  def dump(integer) do
    "INTEGER: #{integer}"
  end
end
```

```
0  * 282 inspectable.ex  Elixir  alchemist
```

Inspectable

Exposes a dump function for logging type information.

Implementing the Protocol for Integer and String.

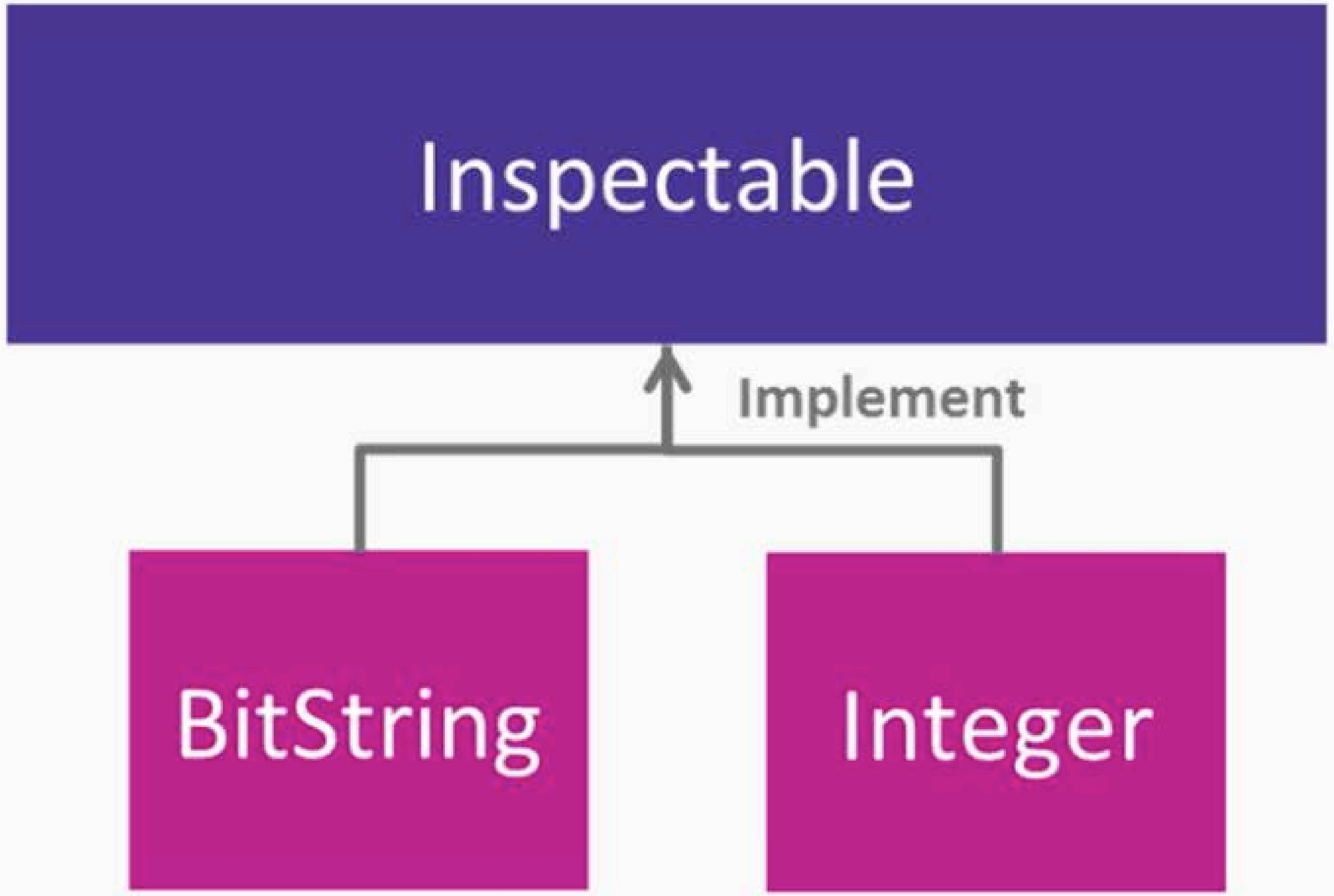


```
➜  inspectable_protocol emacs -nw
➜  inspectable_protocol iex -S mix
Erlang/OTP 19 [erts-8.2] [source] [64-bit] [smp:8:8] [async-threads:10] [hipe] [
kernel-poll:false] [dtrace]

Compiling 1 file (.ex)
Generated inspectable_protocol app
Interactive Elixir (1.4.2) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Inspectable.dump(0)
"ZERO!"
iex(2)> Inspectable.dump(1)
"INTEGER: 1"
iex(3)> Inspectable.dump("hello")
"STRING: hello"
iex(4)> 
```

Inspectable

Exposes a dump function for logging type information.

Implementing the Protocol for Integer and String.



Any

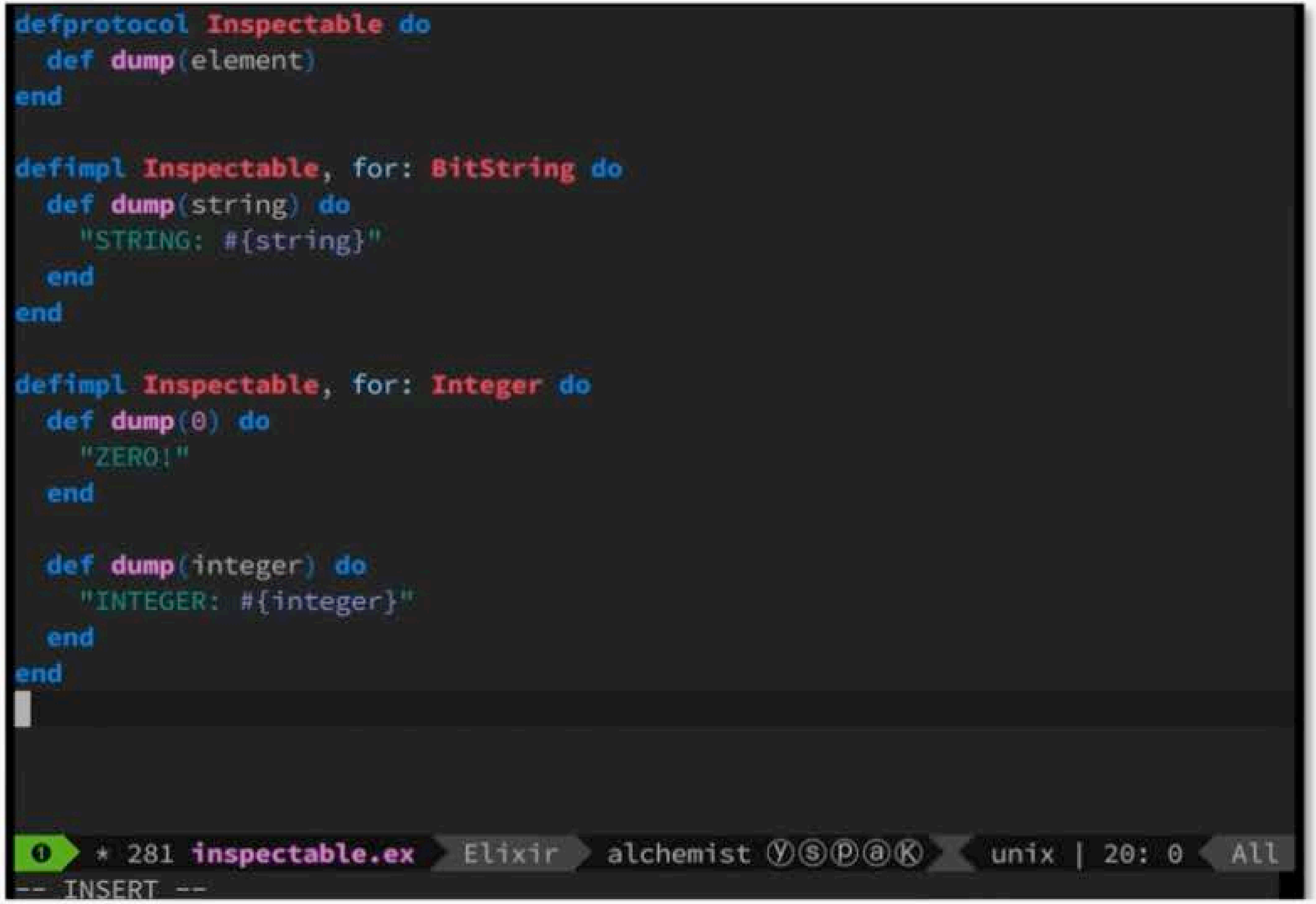
Default implementation that matches any type

Default implementation of a Protocol for all data types.

Has to be specifically enabled for the protocol.

Any

Implementing the Inspectable Protocol for every type.

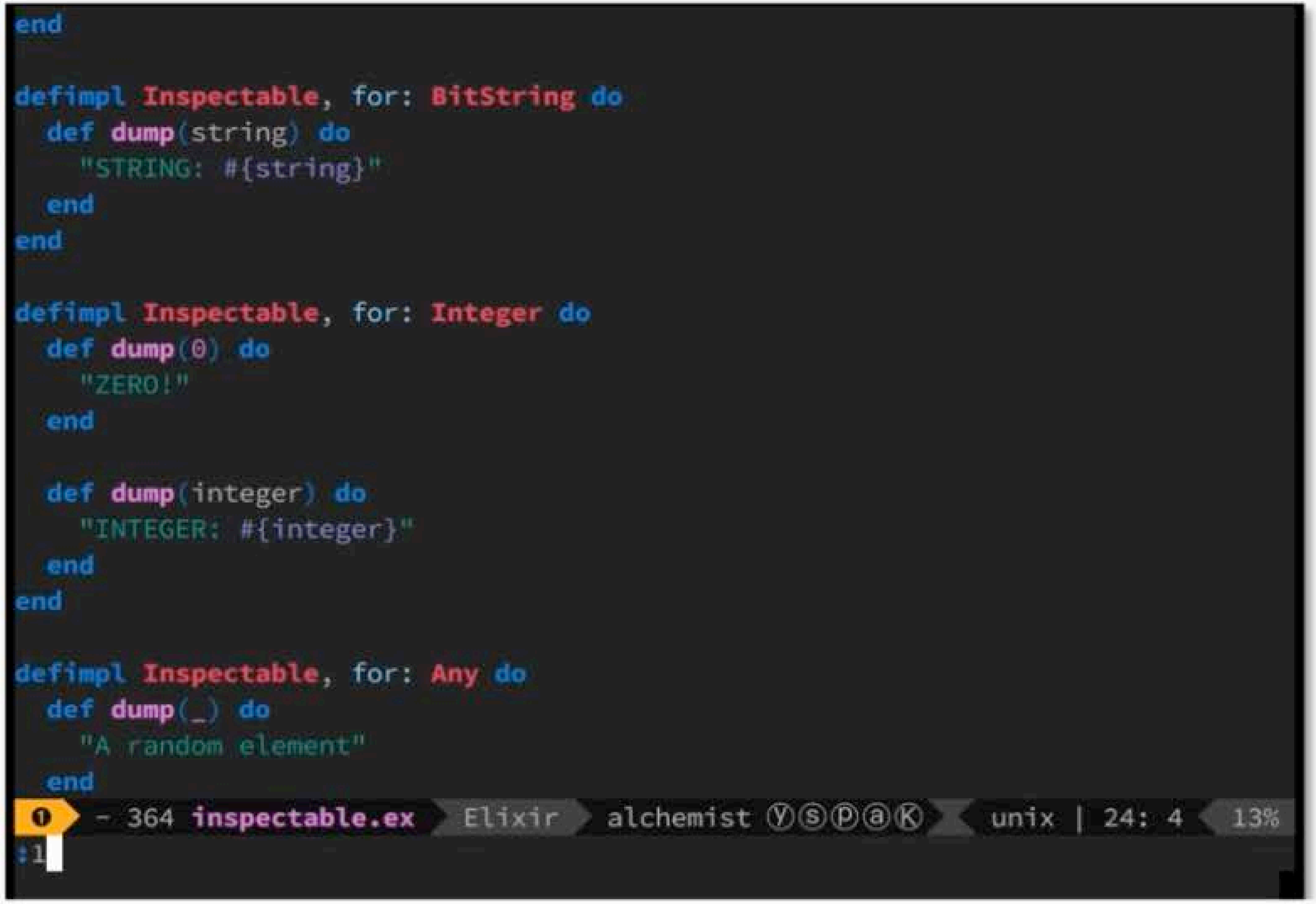


```elixir
defprotocol Inspectable do
  def dump(element)
end

defimpl Inspectable, for: BitString do
  def dump(string) do
    "STRING: #{string}"
  end
end

defimpl Inspectable, for: Integer do
  def dump(0) do
    "ZERO!"
  end

  def dump(integer) do
    "INTEGER: #{integer}"
  end
end
```

```
❶ * 281 inspectable.ex  Elixir  alchemist ⓨⓢⓟⓐⓚ  unix | 20: 0  All
-- INSERT --
```

Any

Implementing the Inspectable Protocol for every type.

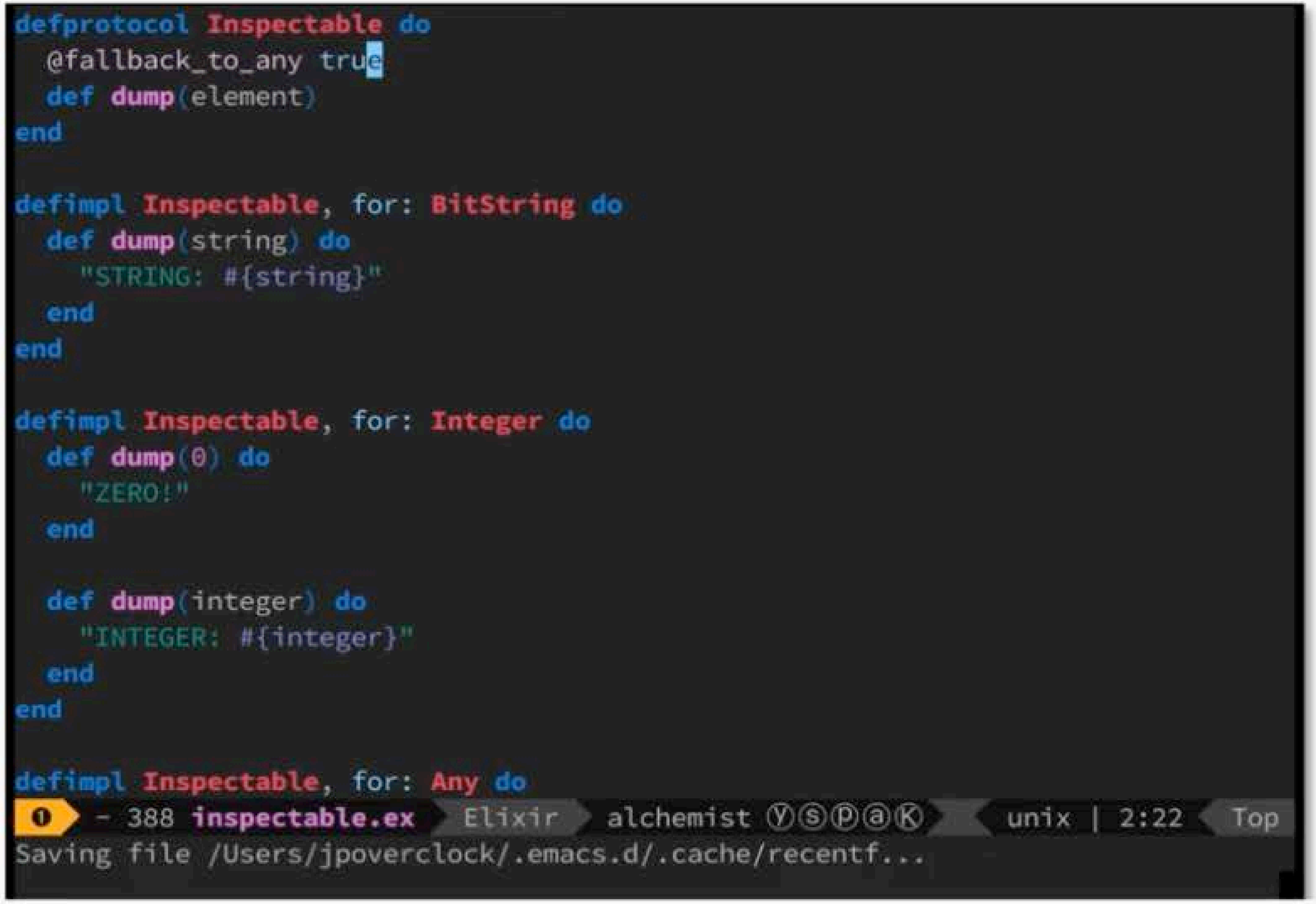


```
end

defimpl Inspectable, for: BitString do
  def dump(string) do
    "STRING: #{string}"
  end
end

defimpl Inspectable, for: Integer do
  def dump(0) do
    "ZERO!"
  end

  def dump(integer) do
    "INTEGER: #{integer}"
  end
end

defimpl Inspectable, for: Any do
  def dump(_) do
    "A random element"
  end
 ❶ - 364 inspectable.ex  Elixir  alchemist ⓨⓢⓟⓐⓚ  unix | 24: 4  13%
:1
```

Packt>

Any

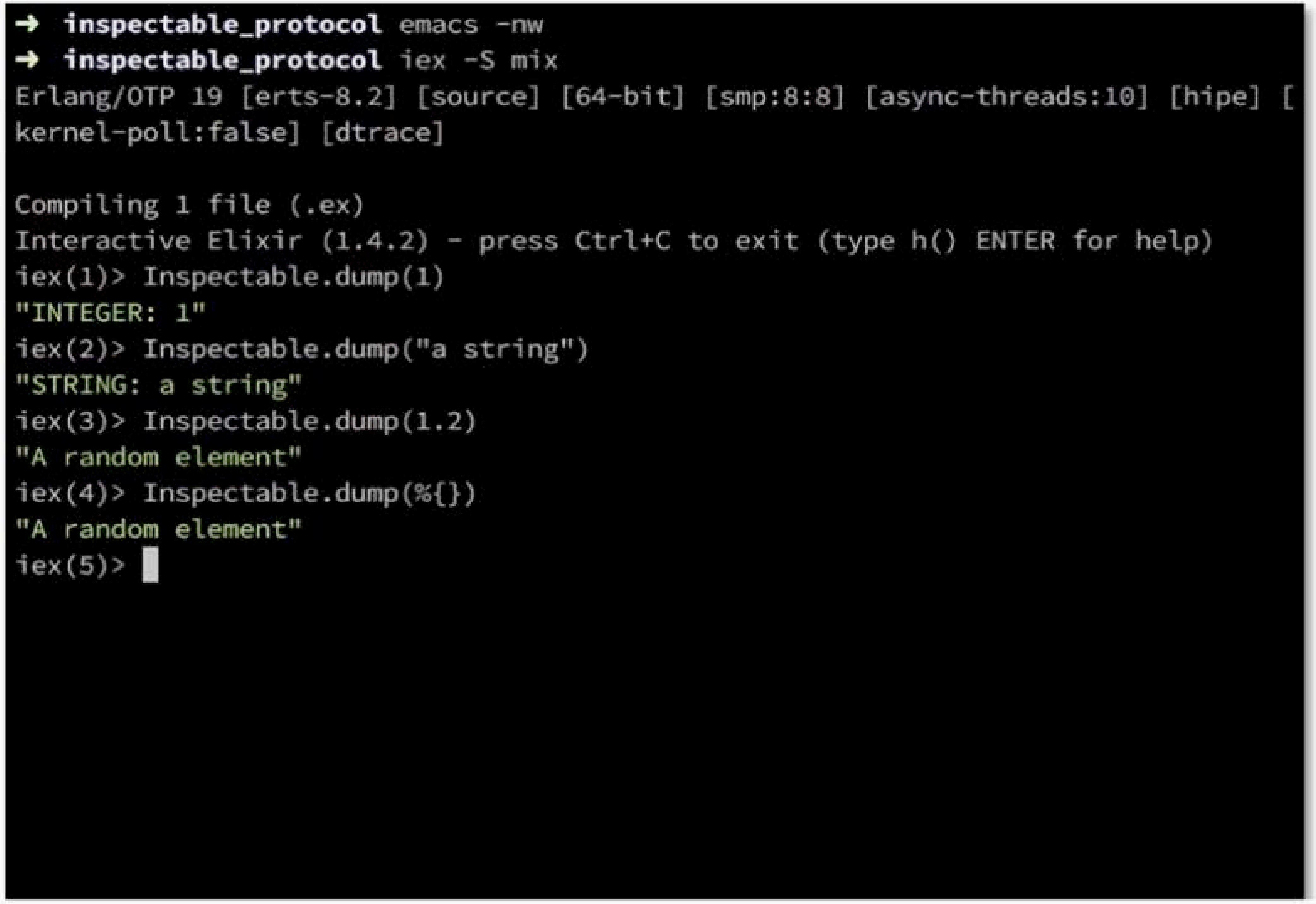
Implementing the Inspectable Protocol for every type.



```
defprotocol Inspectable do
  @fallback_to_any true
  def dump(element)
end

defimpl Inspectable, for: BitString do
  def dump(string) do
    "STRING: #{string}"
  end
end

defimpl Inspectable, for: Integer do
  def dump(0) do
    "ZERO!"
  end

  def dump(integer) do
    "INTEGER: #{integer}"
  end
end

defimpl Inspectable, for: Any do
 ❶  - 388 inspectable.ex  Elixir  alchemist ⓨⓢⓟⓐⓚ   unix | 2:22  Top
Saving file /Users/jpoverclock/.emacs.d/.cache/recentf...
```

Any

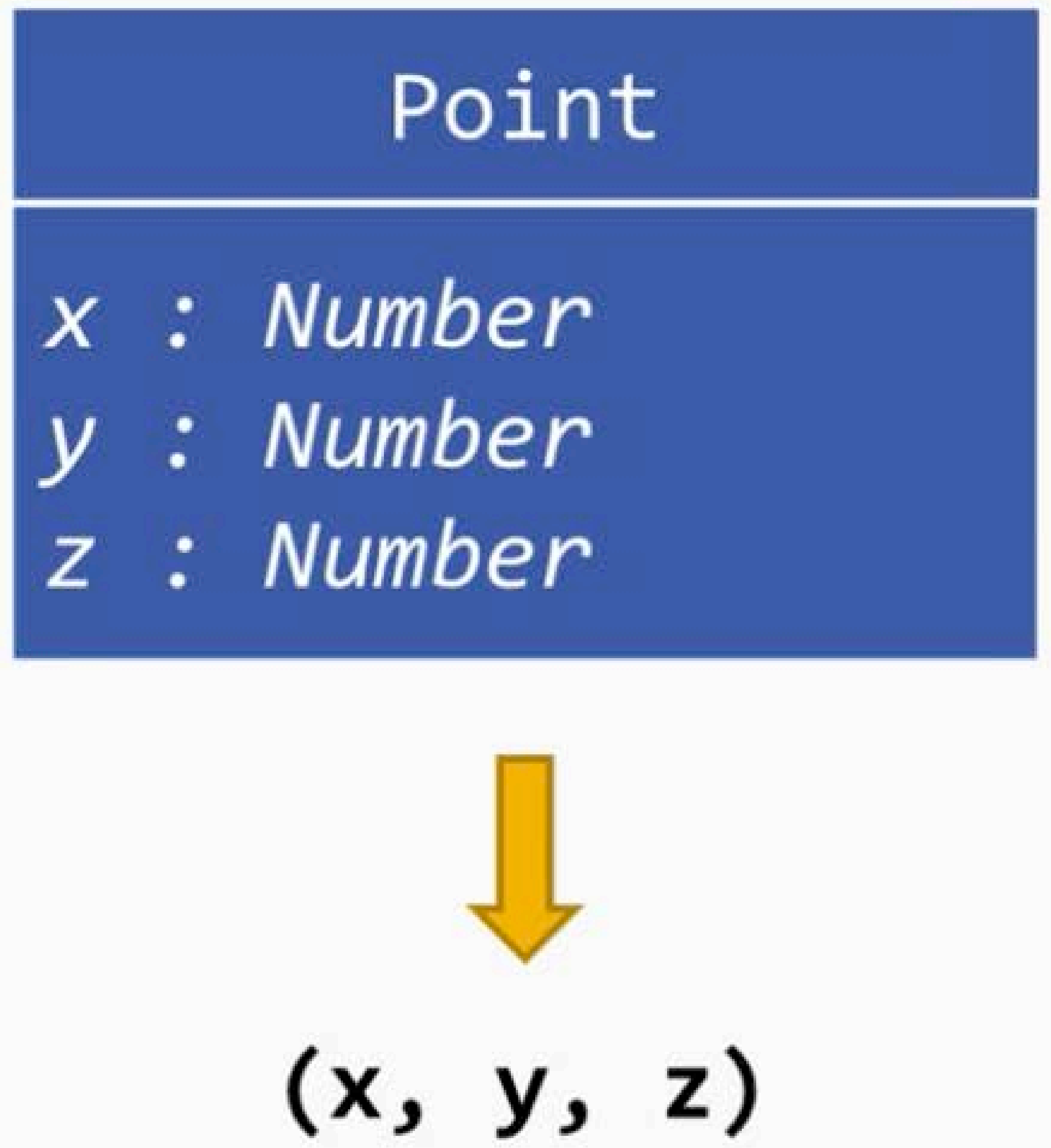
Implementing the Inspectable Protocol for every type.



```
➜  inspectable_protocol emacs -nw
➜  inspectable_protocol iex -S mix
Erlang/OTP 19 [erts-8.2] [source] [64-bit] [smp:8:8] [async-threads:10] [hipe] [
kernel-poll:false] [dtrace]

Compiling 1 file (.ex)
Interactive Elixir (1.4.2) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Inspectable.dump(1)
"INTEGER: 1"
iex(2)> Inspectable.dump("a string")
"STRING: a string"
iex(3)> Inspectable.dump(1.2)
"A random element"
iex(4)> Inspectable.dump(%{})
"A random element"
iex(5)> 
```

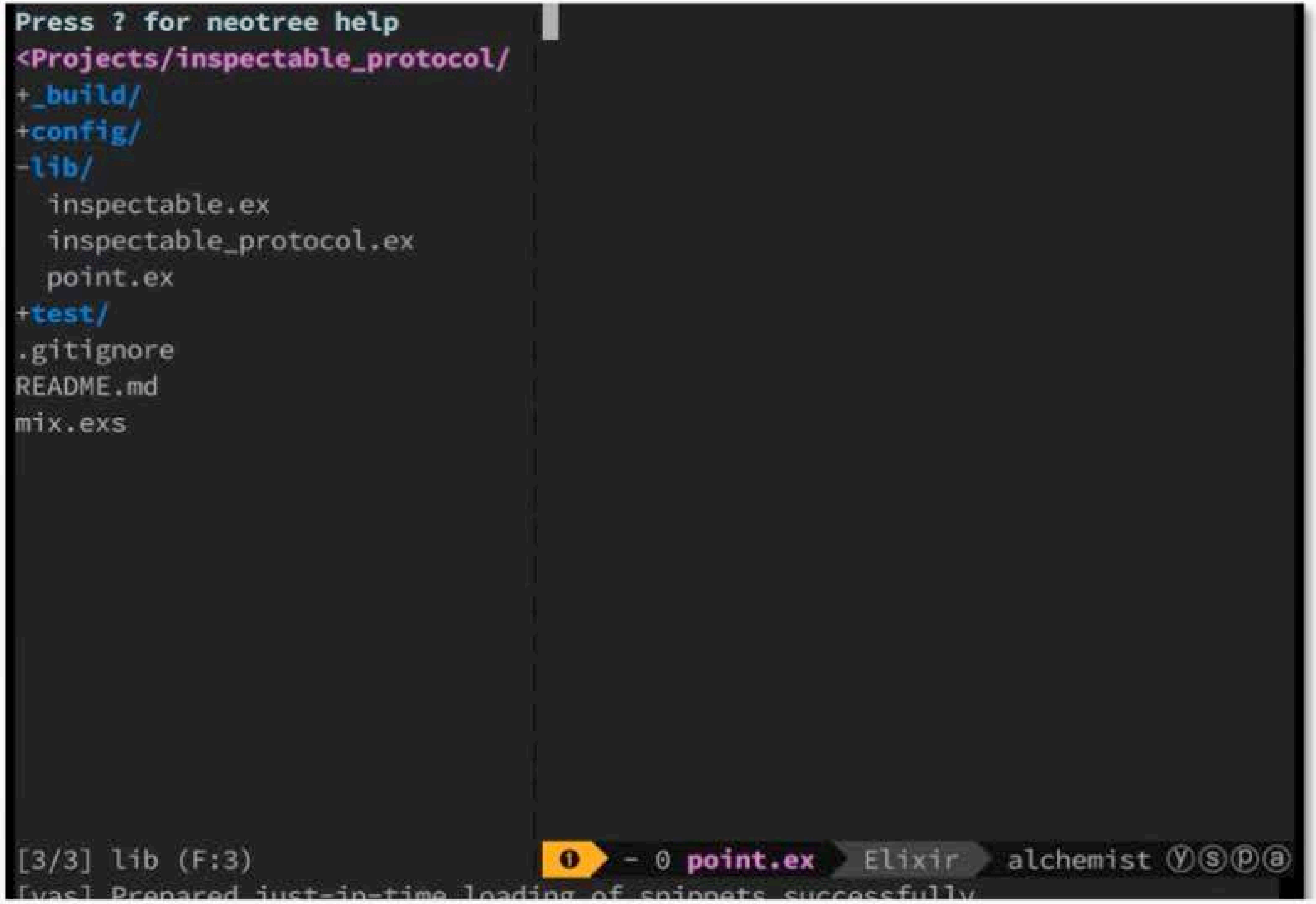
Custom Type

Implementing the Inspectable Protocol for a custom structure.



Custom Type

Implementing the Inspectable Protocol for a custom structure.

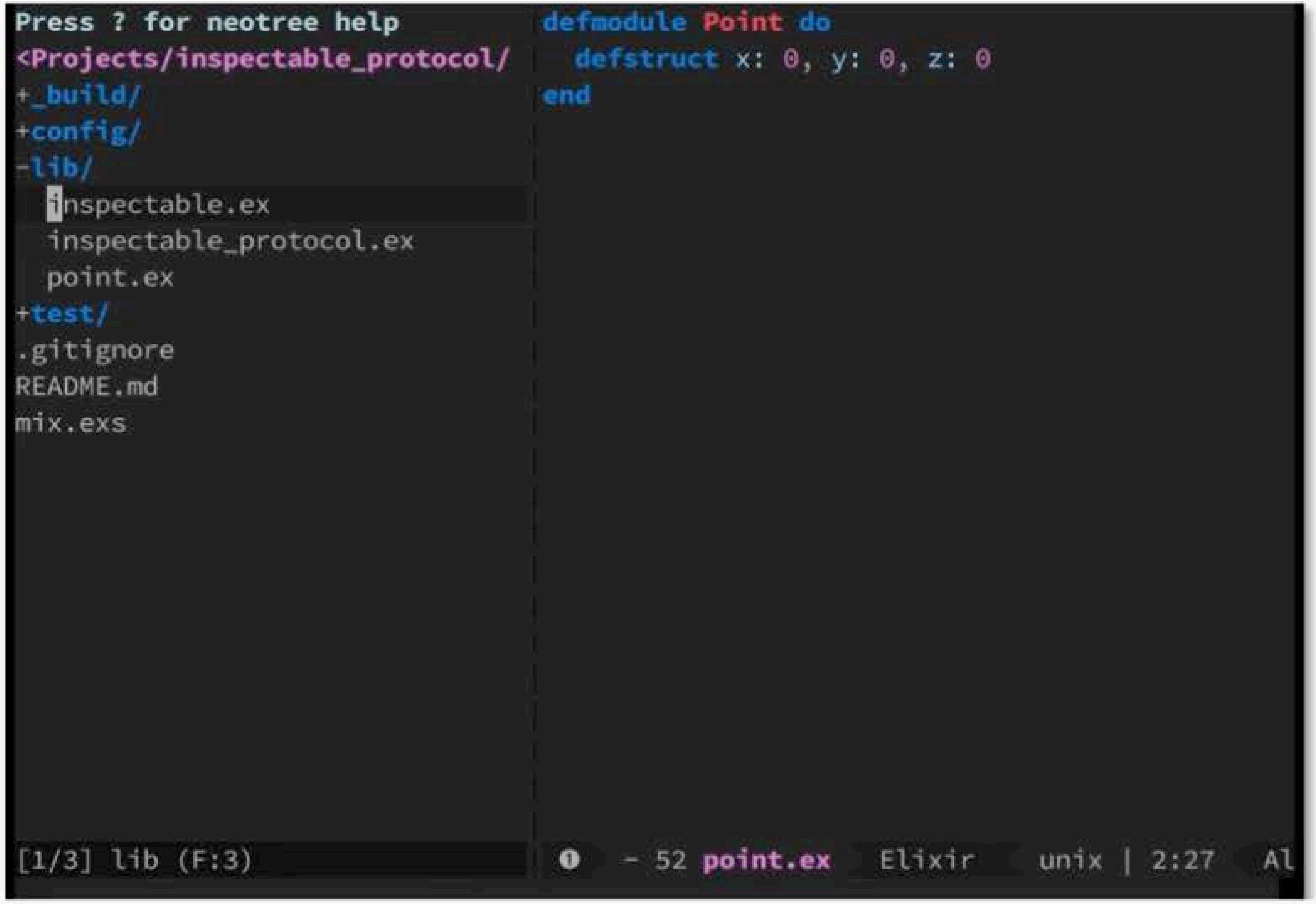


```
Press ? for neotree help
<Projects/inspectable_protocol/
+_build/
+config/
-lib/
  inspectable.ex
  inspectable_protocol.ex
  point.ex
+test/
.gitignore
README.md
mix.exs




[3/3] lib (F:3)                ❶ - 0 point.ex  Elixir  alchemist ⓨⓢⓟⓐ
[yas] Prepared just-in-time loading of snippets successfully
```

Packt>

Custom Type

Implementing the Inspectable Protocol for a custom structure.



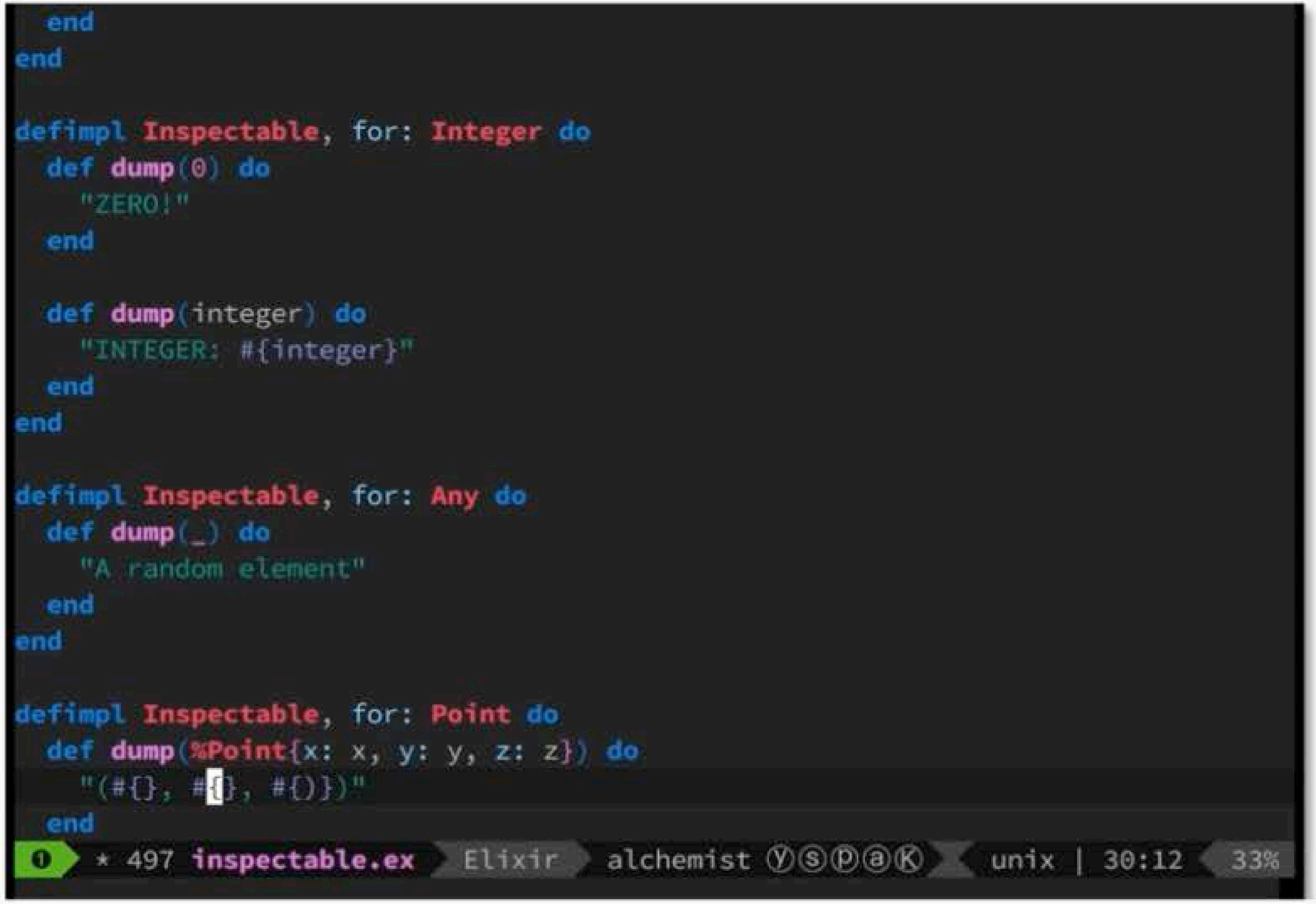
```
Press ? for neotree help
<Projects/inspectable_protocol/
+_build/
+config/
-lib/
  inspectable.ex
  inspectable_protocol.ex
  point.ex
+test/
.gitignore
README.md
mix.exs

[1/3] lib (F:3)
```

```elixir
defmodule Point do
  defstruct x: 0, y: 0, z: 0
end
```

```
- 52 point.ex  Elixir  unix | 2:27  Al
```

Custom Type

Implementing the Inspectable Protocol for a custom structure.

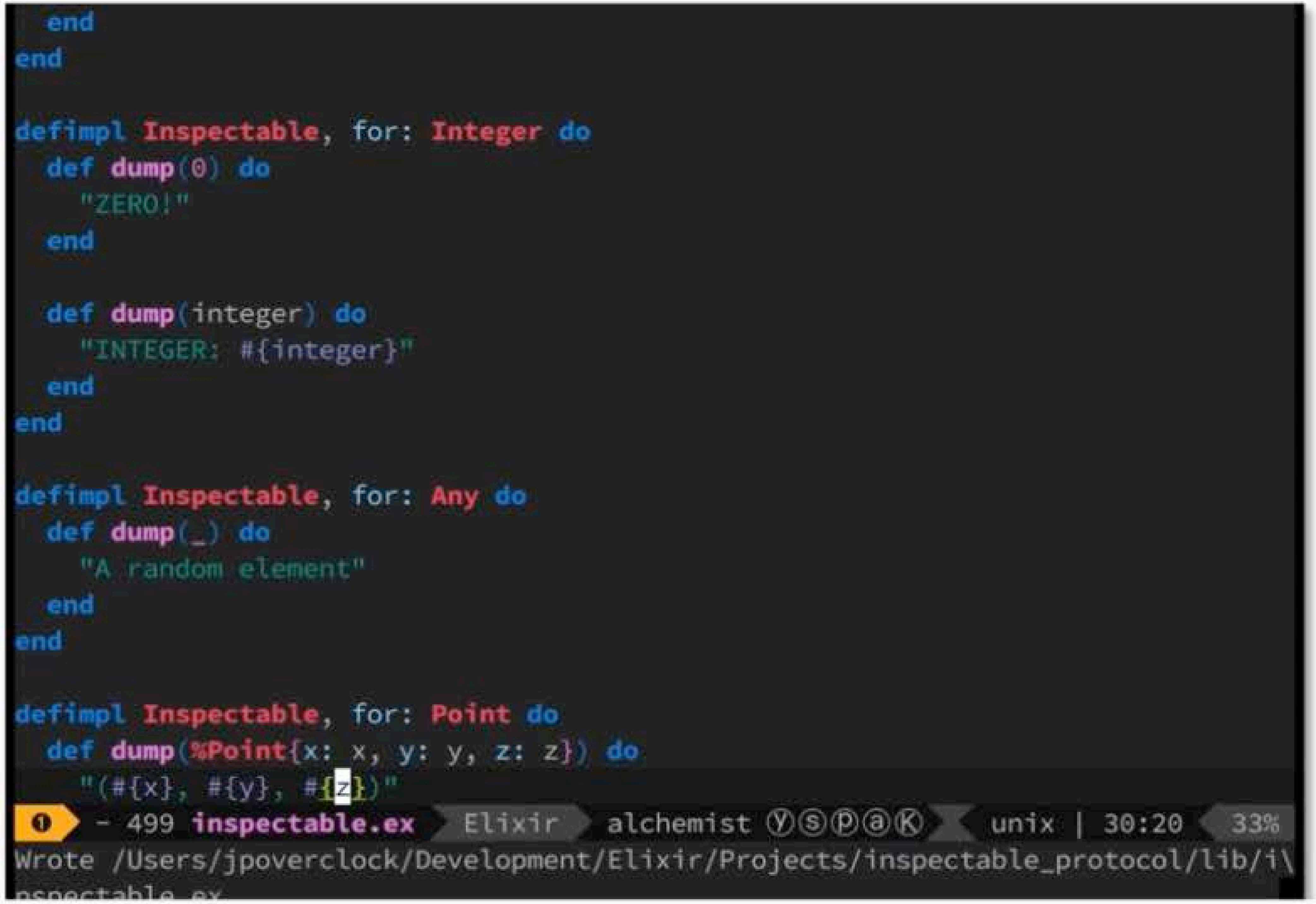


```
  end
end

defimpl Inspectable, for: Integer do
  def dump(0) do
    "ZERO!"
  end

  def dump(integer) do
    "INTEGER: #{integer}"
  end
end

defimpl Inspectable, for: Any do
  def dump(_) do
    "A random element"
  end
end

defimpl Inspectable, for: Point do
  def dump(%Point{x: x, y: y, z: z}) do
    "(#{}, #{}, #{})"
  end
```

```
 1  * 497 inspectable.ex  Elixir  alchemist ⓨⓢⓟⓐⓚ  unix | 30:12  33%
```

Custom Type

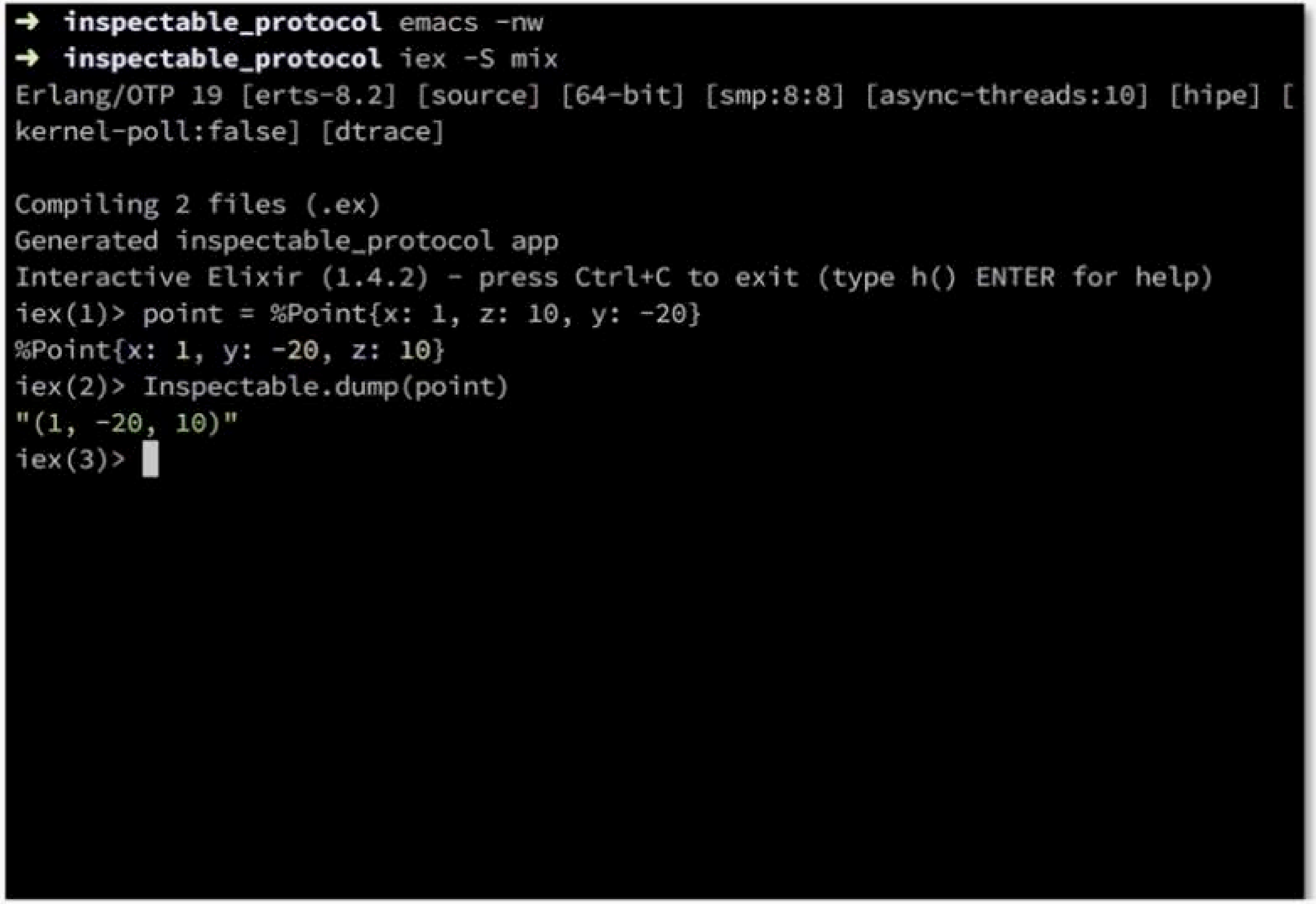
Implementing the Inspectable Protocol for a custom structure.



```
  end
end

defimpl Inspectable, for: Integer do
  def dump(0) do
    "ZERO!"
  end

  def dump(integer) do
    "INTEGER: #{integer}"
  end
end

defimpl Inspectable, for: Any do
  def dump(_) do
    "A random element"
  end
end

defimpl Inspectable, for: Point do
  def dump(%Point{x: x, y: y, z: z}) do
    "(#{x}, #{y}, #{z})"
 ❶  - 499 inspectable.ex  Elixir  alchemist ⓨⓢⓟⓐⓚ  unix | 30:20  33%
Wrote /Users/jpoverclock/Development/Elixir/Projects/inspectable_protocol/lib/i\
nspectable.ex
```

Custom Type

Implementing the Inspectable Protocol for a custom structure.



```
➜ inspectable_protocol emacs -nw
➜ inspectable_protocol iex -S mix
Erlang/OTP 19 [erts-8.2] [source] [64-bit] [smp:8:8] [async-threads:10] [hipe] [
kernel-poll:false] [dtrace]

Compiling 2 files (.ex)
Generated inspectable_protocol app
Interactive Elixir (1.4.2) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> point = %Point{x: 1, z: 10, y: -20}
%Point{x: 1, y: -20, z: 10}
iex(2)> Inspectable.dump(point)
"(1, -20, 10)"
iex(3)>
```

Summary

- Defined Protocols
- Implemented Protocols