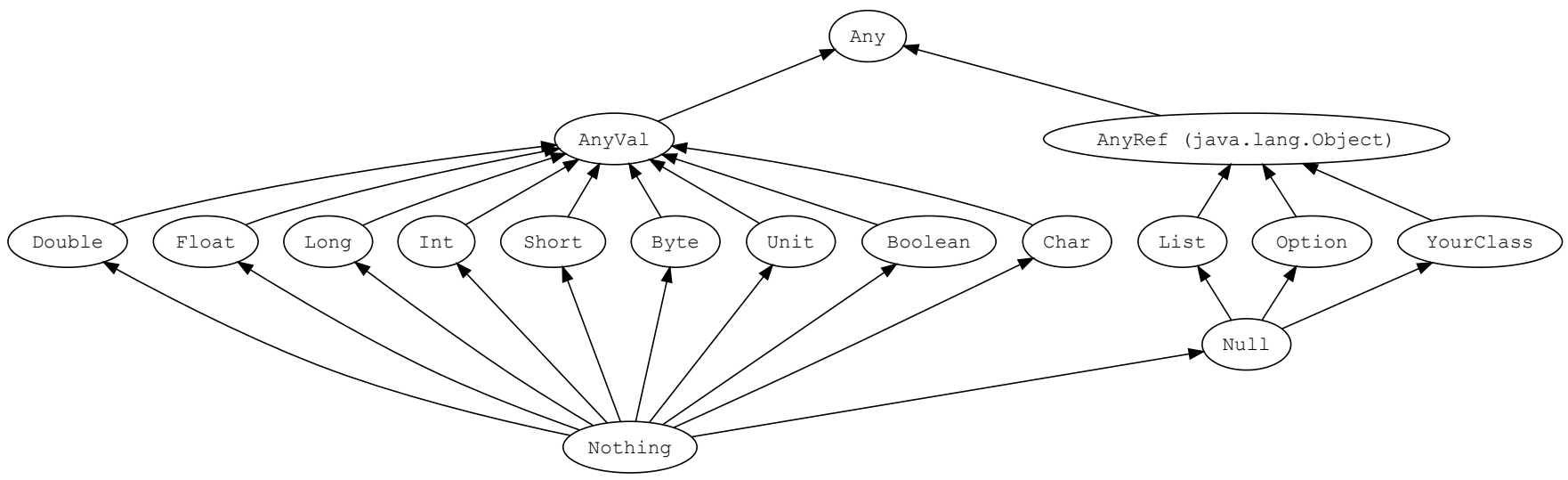
TOUR OF SCALA

UNIFIED TYPES

In Scala, all values have a type, including numerical values and functions. The diagram below illustrates a subset of the type hierarchy.



Scala Type Hierarchy

`Any` is the supertype of all types, also called the top type. It defines certain universal methods such as `equals`, `hashCode`, and `toString`. `Any` has two direct subclasses: `AnyVal` and `AnyRef`.

`AnyVal` represents value types. There are nine predefined value types and they are non-nullable: `Double`, `Float`, `Long`, `Int`, `Short`, `Byte`, `Char`, `Unit`, and `Boolean`. `Unit` is a value type which carries no meaningful information. There is exactly one instance of `Unit` which can be declared literally like so: `()`. All functions must return something so sometimes `Unit` is a useful return type.

`AnyRef` represents reference types. All non-value types are defined as reference types. Every user-defined type in Scala is a subtype of `AnyRef`. If Scala is used in the context of a Java runtime environment, `AnyRef` corresponds to `java.lang.Object`.

Here is an example that demonstrates that strings, integers, characters, boolean values, and functions are all of type `Any` just like every other object:

```
val list: List[Any] = List(
  "a string",
  732,  // an integer
  'c',  // a character
  true, // a boolean value
  () => "an anonymous function returning a string"
)

list.foreach(element => println(element))
```

It defines a value `list` of type `List[Any]`. The list is initialized with elements of various types, but each is an instance of `scala.Any`, so you can add them to the list.

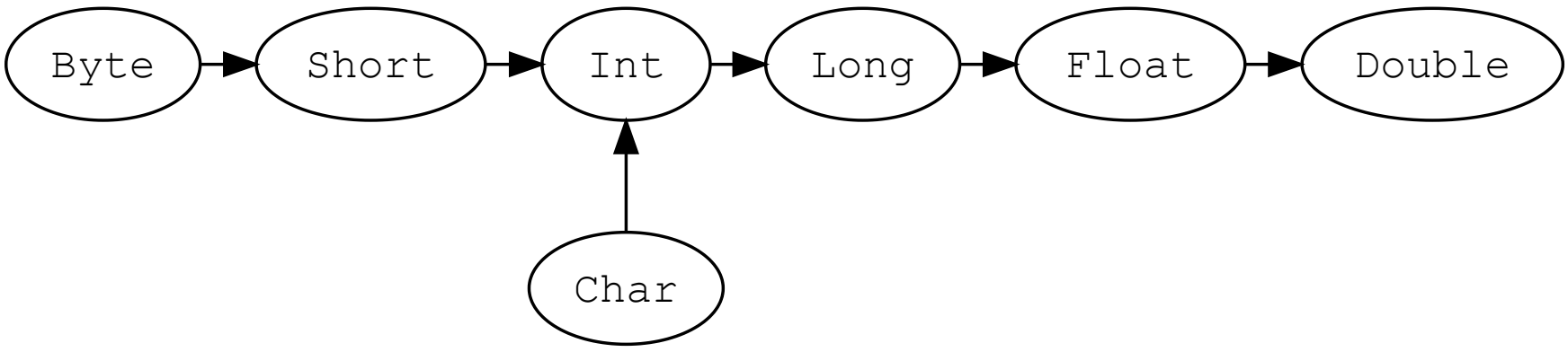
Here is the output of the program:

```
a string
732
```

```
732
c
true
<function>
```

Type Casting

Value types can be cast in the following way:



For example:

```
val x: Long = 987654321
val y: Float = x  // 9.8765434E8 (note that some precision is lost in this case)

val face: Char = '☺'
val number: Int = face  // 9786
```

Casting is unidirectional. This will not compile:

```
val x: Long = 987654321
val y: Float = x  // 9.8765434E8
val z: Long = y  // Does not conform
```

You can also cast a reference type to a subtype. This will be covered later in the tour.

Nothing and Null

`Nothing` is a subtype of all types, also called the bottom type. There is no value that has type `Nothing`. A common use is to signal non-termination such as a thrown exception, program exit, or an infinite loop (i.e., it is the type of an expression which does not evaluate to a value, or a method that does not return normally).

`Null` is a subtype of all reference types (i.e. any subtype of AnyRef). It has a single value identified by the keyword literal `null`. `Null` is provided mostly for interoperability with other JVM languages and should almost never be used in Scala code. We'll cover alternatives to `null` later in the tour.

← previous next →

Contributors to this page: