Datatypes

January 2, 2025

[1]:
```
// Byte - 8-bit signed value
val byteValue1: Byte = 10
val byteValue2: Byte = 127
println(s"Byte values: $byteValue1, $byteValue2")
```

```
Byte values: 10, 127
```

```
byteValue1 = 10
byteValue2 = 127
```

[1]: 127

[2]:
```
// Short - 16-bit signed value
val shortValue1: Short = -32768
val shortValue2: Short = 32767
println(s"Short values: $shortValue1, $shortValue2")
```

```
Short values: -32768, 32767
```

```
shortValue1 = -32768
shortValue2 = 32767
```

[2]: 32767

[3]:
```
// Int - 32-bit signed value
val intValue1: Int = 2147483647
val intValue2: Int = -2147483648
println(s"Int values: $intValue1, $intValue2")
```

```
Int values: 2147483647, -2147483648
```

```
intValue1 = 2147483647
intValue2 = -2147483648
```

[3]: -2147483648

[4]:
```
// Long - 64-bit signed value
val longValue1: Long = 9223372036854775807L
```

```
val longValue2: Long = -9223372036854775808L
println(s"Long values: $longValue1, $longValue2")
```

```
Long values: 9223372036854775807, -9223372036854775808
```

```
longValue1 = 9223372036854775807
longValue2 = -9223372036854775808
```

[4]: -9223372036854775808

[5]:
```
// Float - 32-bit single-precision floating point
val floatValue1: Float = 3.14f
val floatValue2: Float = -3.14f
println(s"Float values: $floatValue1, $floatValue2")
```

```
Float values: 3.14, -3.14
```

```
floatValue1 = 3.14
floatValue2 = -3.14
```

[5]: -3.14

[6]:
```
// Double - 64-bit double-precision floating point
val doubleValue1: Double = 2.71828
val doubleValue2: Double = -2.71828
println(s"Double values: $doubleValue1, $doubleValue2")
```

```
Double values: 2.71828, -2.71828
```

```
doubleValue1 = 2.71828
doubleValue2 = -2.71828
```

[6]: -2.71828

[7]:
```
// Char - 16-bit unsigned Unicode character
val charValue1: Char = 'A'
val charValue2: Char = '\u263A' // Unicode smiley face
println(s"Char values: $charValue1, $charValue2")
```

```
Char values: A,
```

```
charValue1 = A
charValue2 =
```

[7]:

[8]:
```
// String - A sequence of characters
val stringValue1: String = "Hello, Scala!"
val stringValue2: String = "Multiline\nString"
println(s"String values:\n$stringValue1\n$stringValue2")
```

```
String values:
Hello, Scala!
Multiline
String
```

```
stringValue1 = Hello, Scala!
stringValue2 =


Multiline
String
```

[9]:
```
// Boolean - Either true or false
val booleanValue1: Boolean = true
val booleanValue2: Boolean = false
println(s"Boolean values: $booleanValue1, $booleanValue2")
```

```
Boolean values: true, false
```

```
booleanValue1 = true
booleanValue2 = false
```

[9]:
```
false
```

[10]:
```
// Unit - No value
val unitValue: Unit = {
  println("Unit type example")
}
println(s"Unit value: $unitValue")
```

```
Unit type example
Unit value: ()
```

```
unitValue = ()
```

[10]:
```
()
```

[11]:
```
// Null - Null or empty reference
val nullString: String = null
val nullObject: Null = null
println(s"Null values: $nullString, $nullObject")
```

```
Null values: null, null
```

```
nullString = null
nullObject = null
```

[11]:
```
null
```

[12]:
```
// Nothing - Subtype of all types, no values
def exampleFailure(message: String): Nothing = {
  throw new RuntimeException(message)
}
// Uncomment to test: exampleFailure("This will throw an exception")
```

```
exampleFailure: (message: String)Nothing
```

[13]:
```
// Any - Supertype of all types
val anyValue1: Any = 42
val anyValue2: Any = "Scala Rocks!"
println(s"Any values: $anyValue1, $anyValue2")
```

```
Any values: 42, Scala Rocks!
```

```
anyValue1 = 42
anyValue2 = Scala Rocks!
```

[13]:
```
Scala Rocks!
```

[14]:
```
// AnyRef - Supertype of all reference types
val anyRefValue: AnyRef = "This is AnyRef"
println(s"AnyRef value: $anyRefValue")
```

```
AnyRef value: This is AnyRef
```

```
anyRefValue = This is AnyRef
```

[14]:
```
This is AnyRef
```

[ ]:
```
```