Kotlin's Type Inference

Understand Kotlin's powerful type inference capabilities and how to use them to write more succinct code.

WE'LL COVER THE FOLLOWING

Type inference is a compiler feature that allows you to omit types in your code when the compiler can infer it for you.

Type Inference in Kotlin

Kotlin's compiler can infer the types of most variables, so adding the type is optional:

```
// Run the code to see the variable's types
val string = "Educative"
val int = 27
val long = 42L
val double = 2.71828
val float = 1.23f
val bool = true
```

Note: The terms on the left-hand side of the equals sign are just the variable names, not the data types.

Type Inference for Literals

The compiler automatically infers that the `string` variable must have type `String` because it's assigned to the value `"Educative"`.

For integer values, the compiler infers `Int` by default. You can use the `L` suffix as in `42L` to transform the value into a `Long`. Similarly, the compiler infers `Double` for floating point numbers unless you add an `f` as suffix as in `1.23f`, in which case it is considered a `Float`. There's no such shortcut suffix for `Byte` or `Short` because these aren't used as often.

> If you run the code above, it will print the mapped Java type for each variable at runtime.
>
> (Kotlin reflection is currently not available on Educative so it prints the Java types. These are evaluated at runtime using `string.javaClass`, `int.javaClass`, and so forth.)

Type Inference for Objects

Naturally, type inference doesn't only work with literal values on the right-hand side. The compiler can infer types of object just as easily:

```
import java.util.*
import java.time.*
import java.io.*

val stringBuffer = StringBuffer("PREFIX")
val localDate = LocalDate.now()
val file = File("foo.txt")
```

Essentially, as long as it's possible to infer the variable type, Kotlin's compiler can do so.

> **Tip:** You may still choose to *add explicit types for clarity* or to use a *more abstract type* than what would be inferred. It is good practice to program against abstract interfaces where it makes sense.

More Type Inference

Apart from simple variable types, Kotlin's compiler can infer types in lambda expressions, function return types, generic type parameters, and so forth – given sufficient context.

Using type inference is fundamental in idiomatic Kotlin code so you won't see many explicit types in this course. If you're working in a larger Kotlin codebase with IntelliJ, you can switch on type hints to let the IDE show you the inferred types. This may be helpful while getting used to inferred types.

Quiz

To understand the compiler... you must become the compiler.

1 Which type is inferred for "17"?

COMPLETED 0%

1 of 3

Summary

Here are the key takeaways from this lesson:

- The Kotlin compiler can infer most types.
  - Writing down the type explicitly is optional in these cases.
- You may prefer to use explicit types for clarity.
  - Especially when the right-hand side is a complex expression or

function call.

- Inferred types are ubiquitous in idiomatic Kotlin code.

In the next lesson, you will discover Kotlin’s fundamental feature of nullable types which make every use of `null` in your code explicit and well-defined.