## Compiling

We write programs in a high-level programming language like Kotlin, Scala, Java, C++, or C.

A compiler translates the source code to object code (machine code).

For C and C++, it is customary to compile to native machine code. It can be executed directly by the processor.

Native machine code is different for different processors, operating systems, and can depend on library versions.

Kotlin (like Java and Scala) are normally translated to object code for the JVM (Java virtual machine). A Java runtime environment is needed on the computer to execute the program. The exact same object code works on any system. JVM is heavily used on servers and on Android.



## Compiling a program

We cannot compile Kotlin scripts:

```
$ ktc hello.kt
error: expecting a top level declaration
println("Hello World")
^
```

Only declarations can be compiled:

- global variables with `val` and `var`,
- functions with `fun`,
- class definitions with `data class` or `class`.

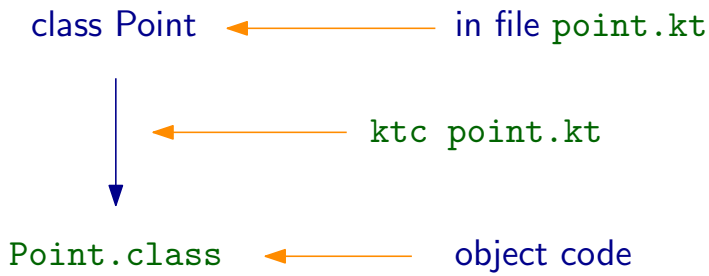
So how can we write a program? If all we have are declarations, how can we execute any code?

The magic `main` function.



## Compiling a class

The Kotlin compiler compiles each class to an object file.




The `Point` class can now be used in any code that can find `Point.class`.

```
$ ktc
Welcome to Kotlin version 1.0.5-2
>>> val p = Point(7, 5)
```

JVM finds `Point.class`



## Compiled Kotlin programs

1. In your source file `hello.kt`, define a function `main(args: Array<String>)` returning nothing:

```
fun main(args: Array<String>) {
  println("Hello World")
}
```

2. Compile the source file, resulting an a class file `HelloKt.class`:

```
$ ktc hello.kt
```

3. Run the program:

```
$ kt HelloKt arguments
```

This name is generated from the name of the source file.