```
// Gives us read only map
fun getMap(): Map<String, Double> {
    return mapOf("EUR" to 123.20, "USD" to 110.34, "CHF" to 111.4)
}
```

And then, we access to members:

```
fun printMap(key: String) {
    val map = getMap()
    if (key in map) println(map[key]) else println("There is no such key $key")
}
```

Lazy initialization

Kotlin supports lazy initialization. Take a look at the following example *MyLazy.kt*:

```
class MyLazy {
    val lazyVar: String by lazy() {
        println("We called lazy initialization.")
        "Lazy value"
    }
}
```

We created one simple class with one field which will not be initialized untill it is used. Notice we put *println("We called lazy initialization.") in lazy initialization routine*, and then call it from our main function:

```
val lazy = MyLazy()
println(lazy.lazyVar)
println(lazy.lazyVar)
println(lazy.lazyVar)
println(lazy.lazyVar)
println(lazy.lazyVar)
```

We will have the following console output:

```
We called lazy initialization.
Lazy value
Lazy value
Lazy value
Lazy value
Lazy value
```

Print line with message " *We called lazy initialization* " is executed at the moment we initialized our field - only once. Every other time we have value associated to field so no message is printed again.