Objects are the basis of object-oriented programming. In Kotlin, every piece of data is an object.

An object

- stores data (has state), and
- provides methods to access or manipulate its state.

Consider an object as an atomic units. Clients (users of the object) do not care about the implementation of the object, they only use the exposed methods and fields.

A class defines a new type of object. Think about a class as a blueprint for objects. You can create objects from the blueprint.

KAIST CS109

Date class

This type guarantees that date values are always consistent. The init block is executed when the object is created.

We can write functions to work with these objects, for instance to compute the number of days between two Dates.

But:

- We cannot guarantee that objects are consistent (that is, that they represent a legal date)
- There is no obvious connection between the date type and the functions that work on dates.

KAIST CS109

Methods

Let's add a method to convert the date to a day index, starting at day 0 on 1901-01-01.

Every object has a method toString. We can override it to

KAIST CS109

In Kotlin, operators are just normal methods. For instance, in the expression a - b, we are calling the minus method of the a object. The same expression can be written a.minus(b).

```
data class Date(...) {
   // ...
   operator fun minus(rhs: Date): Int =
        dayIndex() - rhs.dayIndex()
}
>>> val d1 = Date(2017, 5, 14)
>>> val d2 = Date(2017, 6, 16)
>>> d2 - d1
33
```

```
Let's compute the number of days between two dates:
data class Date(...) {
   // ...
   fun diff(rhs: Date): Int =
      dayIndex() - rhs.dayIndex()
}

>>> val d1 = Date(2017, 5, 14)
>>> val d2 = Date(2018, 1, 13)
>>> d2.diff(d1)
244
>>> d1.diff(d2)
```

KAIST CS109

-244

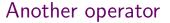
Operators

Overloading

It is allowed to have different functions or methods that have the same name, and only differ in the type of arguments they accept.

```
fun f(n: Int) {
   println("Int " + n)
}
fun f(s: String) {
   println("String " + s)
}
f(17)
f("CS109")
```

Overloading means that we have different methods with the same name (here f), distinguished by their argument type.





We can also define + and - operators that add and subtract a number of days from the given date.

```
data class Date(...) {
    // ...
    operator fun minus(rhs: Date): Int =
        dayIndex() - rhs.dayIndex()

    operator fun plus(n: Int): Date =
        num2date(dayIndex() + n)
    operator fun minus(n: Int): Date =
        num2date(dayIndex() - n)
}
```