7장 연산자 오버로딩과 기타 관례

신림프로그래머 최범균

- 연산자 오버로딩
- 연산을 지원하기 위한 메서드 이름 관례
- 위임 프로퍼티

이항 산술 연산자 오버로딩

- operator 키워드와 지정한 함수 이름을 사용해서 연산자 오버로딩

식	함수 이름
a * b	times
a / b	div
a % b	rem (1.1 이전에는 mod)
a + b	plus
a - b	minus

```
data class Point(val x: Int, val y: Int) {
    operator fun plus(other: Point): Point {
        return Point(x + other.x, y + other.y)
    }
}

val p1 = Point(10, 20)
val p2 = Point(30, 40)
println(p1 + p2) // Point(x=40, y=60)
```

이항 산술 연산자 오버로딩

- 확장 함수로도 가능
- 여러 타입에 대한 연산자 오버로딩 가능

```
operator fun Point.times(scale: Double): Point { ... }
```

- 교환 법칙(a op b == b op a)을 지원하려면 각 타입에 연산자 오버로딩 필요

```
operator fun Double.times(p: Point): Point { ... }
```

복합 대입 연산자 오버로딩

- plus와 minus와 같은 연산자를 오버로딩하면, +=, -=의 복합 대입 연산자 자동 지원

```
var point = Point(1, 2)
point += Point(3, 4) // point = point + Point(3, 4)
```

- 리턴 타입이 Unit인 plusAssign 함수가 존재하면 += 연산자에 그 함수 사용
  - 다른 연산자도 비슷하게 minusAssign, timesAssign 등 이름 사용
  - 객체 자신을 변경하는 용도로 사용(리턴 타입이 Unit)

```
operator fun <T> MutableCollection<T>.plusAssign(element: T) {
    this.add(element)
}
val numbers = ArrayList<Int>()
numbers += 42
```

- +=에 대해 plus와 plusAssign을 둘 다 존재하면 컴파일 오류
  - 변수를 val로 하면 plus 대신 plusAssign 사용
  - 일관된 사용을 위해 두 연산을 동시에 정의하지 말 것

단항 연산자 오버로딩

식	함수 이름
+a	unaryPlus
-a	unaryMinus
!a	not
++a, a++	inc
--a, a--	dec

```
operator fun BigDecimal.inc() = this + BigDecimal.ONE

var bd = BigDecimal.ZERO
println(bd++)
```

비교 연산자 오버로딩: equals

- equals 연산자
  - a == b : a?.equals(b) ?: (b == null)
- equals는 Any에 정의되어 있으므로
  - override 필요
  - Any의 equals에 operator가 있으므로 붙일 필요 없음
  - 확장 함수로 정의할 수 없음

비교 연산자 오버로딩: compareTo

- Comparable 인터페이스의 compareTo 메서드 호출 관례 지원
  - <, <=, >, >= 연산자를 Comparable#compareTo 호출로 컴파일
  - a >= b : a.compareTo(b) >= 0

```
class Person(val firstName: String, val lastName: String) : Comparable<Person> {
    override fun compareTo(other: Person): Int {
        return compareValuesBy(this, other, Person::firstName, Person::lastName)
    }
}

val p1 = Person("Alice", "Smith")
val p2 = Person("Bob", "Johnson")
println(p1 < p2)
```

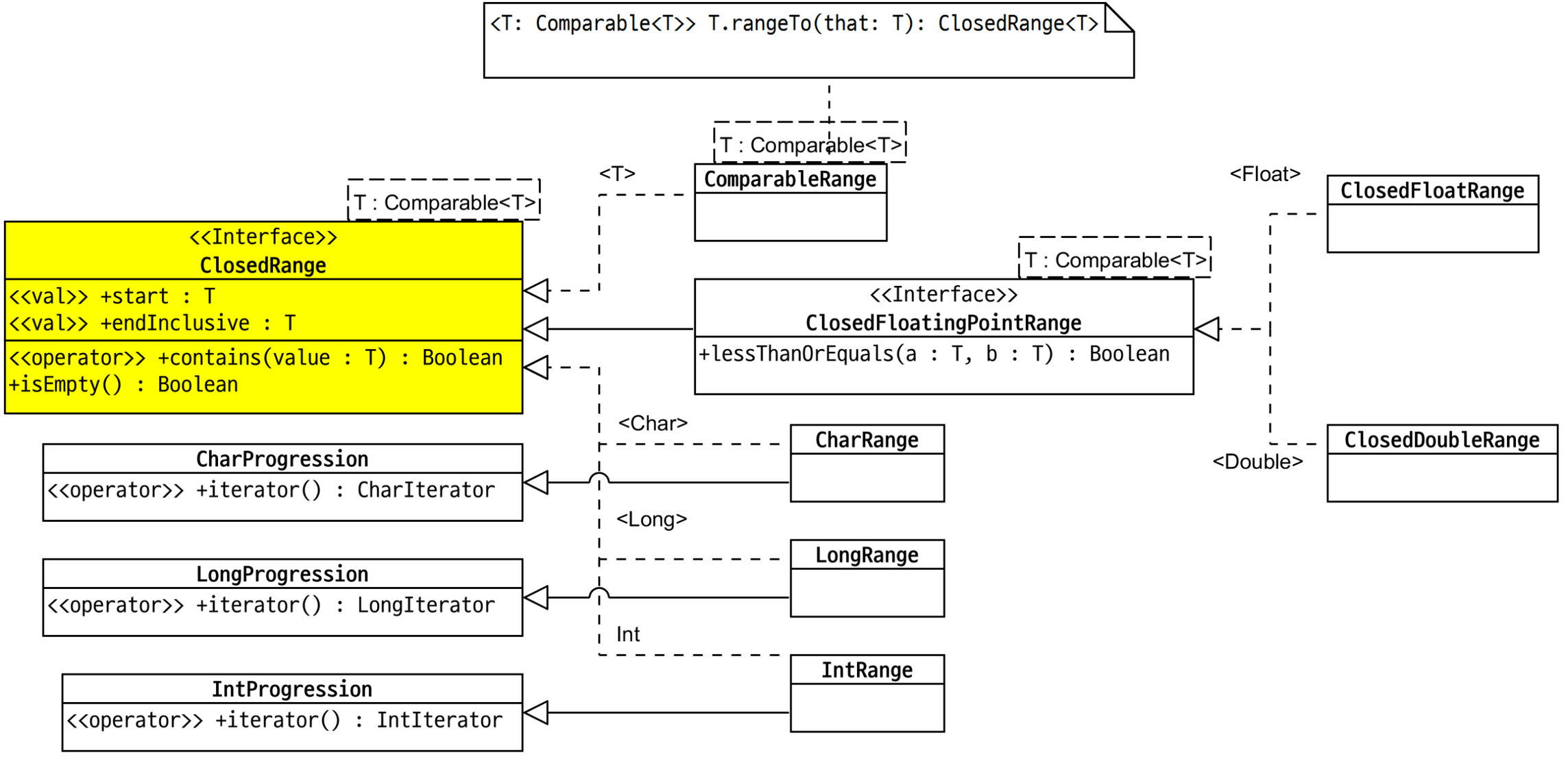
get, set, in

- x[a, b] : x.get(a, b)
- x[a, b] = c : x.set(a, b, c)
- a in c : c.contains(a)

rangeTo

- .. 구문으로 범위(Range) 생성
  - val range = 1..10
- rangeTo 함수 호출로 컴파일
  - start..end : start.rangeTo(end)
- rangeTo의 결과는 ClosedRange<T: Comparable>

ClosedRange



for 루프와 iterator

- kotlin.collections.Iterator를 리턴하는 iterator() 함수는 for 루프에서 사용 가능

```
operator fun ClosedRange<LocalDate>.iterator() : Iterator<LocalDate> =
        object : Iterator<LocalDate> {
            var current = start
            override fun hasNext(): Boolean = current <= endInclusive
            override fun next(): LocalDate = current.apply {
                current = plusDays(1)
            }
        }

fun main(args: Array<String>) {
    val newYear = LocalDate.ofYearDay(2018, 1)
    val daysOff = newYear.minusDays(1)..newYear
    for (dayOff in daysOff) { println(dayOff) }
}
```

구조 분해 선언과 component

```
val (a, b) = p
val a1 = p.component1()
val b1 = p.component2()

for ((key, value) in map) { // Map.Entry

}
```

- data 클래스는 컴파일러가 각 프로퍼티에 대해 component 함수 자동 생성
- 콜렉션의 경우 맨 앞 다섯 원소에 대한 component 확장 함수 제공

위임 프로퍼티

프로퍼티 접근을 다른 객체에 위임하는 기능

```
class Foo {
    var p: Type by Delegate()

    // p 프로퍼티에 대한 접근을 Delegate 객체에 위임
    // p: 위임할 프로퍼티
    // by 오른쪽 : 위임 객체
}
```

- 위임한 프로퍼티를 읽고 쓸 때마다 위임 객체의 getValue/setValue 호출

위임 객체의 함수 규칙

- operator getValue(obj: 프로퍼티포함타입, prop: KProperty<*>): 프로퍼타티입
- operator setValue(obj: 프로퍼티포함타입, prop: KProperty<*>, newValue:프로퍼티타입)

위임 프로퍼티 컴파일 규칙

코틀린코드	컴파일
val x = c.prop	val x = <delegate>.getValue(c, <property>)
c.prop = x	<delegate>.setValue(c, <property>, x)

- delegate: 위임 객체 의미
- property: 프로퍼티에 해당하는 KProperty

위임 프로퍼티 예: lazy

```
// 프로퍼티 초기화 지연
class Person(val name: String) {
    val emails by lazy { loadEmails(this) }
}
```

```
public inline operator fun <T> Lazy<T>.getValue(
        thisRef: Any?, property: KProperty<*>): T = value
```

위임 프로퍼티 예: 맵에 저장

```
class Person {
    private val _attributes = hashMapOf<String, String>()

    val name: String by _attributes
}
```

코틀린 맵이 위임 객체를 위한 getValue/setValue 확장 함수 제공