192010
喜田圭伍

**1-1**

オブジェクト指向はある一つの目的のためだけに存在している。オブジェクト指向とは他人が読み書きしやすいコードを書くためにカプセル化などのルールを守った上でデータや機能のまとまりを作るためにある。オブジェクト指向ではカプセル化というものがあり、ユーザーが使うのに必要な部分だけ可視化し、それ以外は隠すという特徴がある。内部を知らなくてもプログラムを実行するということが可能になります。もう一つの特徴にポリモーフィズムがあり、外面は同じにして、内部の処理を変更できるものです。内部の構造は違っても同じ機能を持つということが可能になるのがポリモーフィズムです。

**1−2**

```
fun main(args: Array<String>) {
    var fizzbuzz = fizzbuzz()


}


fun fizzbuzz() : Unit{
     for (i in 1..100){
        if(i%15== 0){
            println("FizzBuzz")
        }
        else if(i%3 == 0){
            println("Fizz")
        }
        else if(i%5 == 0){
            println("Buzz")
        }
                    else {
            println(i)
        }
    }
}
```

**out**

```
1
2
Fizz
4
```

Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
17
Fizz
19
Buzz
Fizz
22
23
Fizz
Buzz
26
Fizz
28
29
FizzBuzz
31
32
Fizz
34
Buzz
Fizz
37
38
Fizz
Buzz
41
Fizz
43
44
FizzBuzz
46
47
Fizz
49
Buzz
Fizz
52
53
Fizz
Buzz
56
Fizz
58

```
59
FizzBuzz
61
62
Fizz
64
Buzz
Fizz
67
68
Fizz
Buzz
71
Fizz
73
74
FizzBuzz
76
77
Fizz
79
Buzz
Fizz
82
83
Fizz
Buzz
86
Fizz
88
89
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

**1-3**

```
fun main(args: Array<String>) {
    var count = CountChar("zbabczca")
    println(count)

}

fun CountChar(letter: String) :Map<Char,Int>{
```

```
    var count = letter.groupingBy { it }.eachCount()

    return count

}
```

**Out**

```
{z=2, b=2, a=2, c=2}
```

```
interface Figure{
    var center : Point
    fun collideWith(ccl: Circle) :Boolean
}

class Point(var x:Double, var y :Double)

class Circle(override var center : Point, var rad: Double): Figure{
    override fun collideWith(ccl: Circle):Boolean {
        var square_x = (ccl.center.x - center.x)*(ccl.center.x - center.x)
        var square_y = (ccl.center.y - center.y)*(ccl.center.y - center.y)
        if (Math.sqrt((square_x)+(square_y)) <= rad + ccl.rad ) return true else return false
    }
}
```

```
fun main(args: Array<String>) {
    val ccl1 = Circle(Point(5.0,4.0),3.0)
    val ccl2 = Circle(Point(1.0,1.0),2.0)
    val ccl3 = Circle(Point(1.0,1.0),1.9)
    println(ccl1.collideWith(ccl2))
    println(ccl1.collideWith(ccl3))

}
```

**Out**
true
false

1-5
```
class Basket {
    private var buyMap = mutableListOf<String>()

    fun add(item: String) {
        this.buyMap.add(item)
    }

    fun calcPrice(priceList: MutableMap<String, Int>) : Int {
        var totalPrice = 0
        this.buyMap.forEach { item ->
            if (priceList.containsKey(item)) {
                totalPrice += priceList[item] as Int
            }
        }
        return totalPrice
    }
}

fun main(args: Array<String>) {
    val bask = Basket()
    bask.add("juice")
    bask.add("juice")
    bask.add("bagle")
    bask.add("umai")
    bask.add("thunder")
    print(bask.calcPrice(mutableMapOf("juice" to 100, "bagle" to 300, "umai" to 0, "thunder" to
30)))

}
```

**Out**
530