문자열 배열

Mobile Software
2022 Fall

What to do next?

- 배열 생성
- 문자열 배열 리소스를 Array 객체에 저장
- 배열 객체의 모든 원소 출력
  - for, forEach, forEachIndexed
  - Iterator
  - 객체.contentToString()
- 배열 정렬
- 정수 배열

- 강의 노트에 포함된 코드 제공(7-1.소스코드. hwp)

실습 프로젝트 생성

- 새 프로젝트 생성
  - **Project name** : **My Array**
  - **Package name : com.example.myarray**
  - **Activity** : **Empty Activity**
  - **Activity name** : **MainActivity.kt**
  - **Layout name** : **activity_main.xml**

- **자동 생성된 XML 파일**의 root view는 **ConstraintLayout**

1차원 배열 객체 생성 (1/2)

- 배열 객체를 생성하는 3가지 방법
  - arrayOf (*개별 초기값*)
  - Array (원소개수, *일괄 초기값 설정*)

람다 식

```
val strArr:Array<String> = arrayOf("red", "green", "blue")
// val strArr = arrayOf("red", "green", "blue")

val strArr2:Array<String> = Array( size: 3) {""}

val strArr22:Array<String?> = arrayOfNulls<String>( size: 3)
// val strArr22 = arrayOfNulls<String>(3)

val strArr3 = Array(strArr.size) { i -> strArr[i] }
```

배열 객체를 생성하는 방식에 상관없이 type은 같음

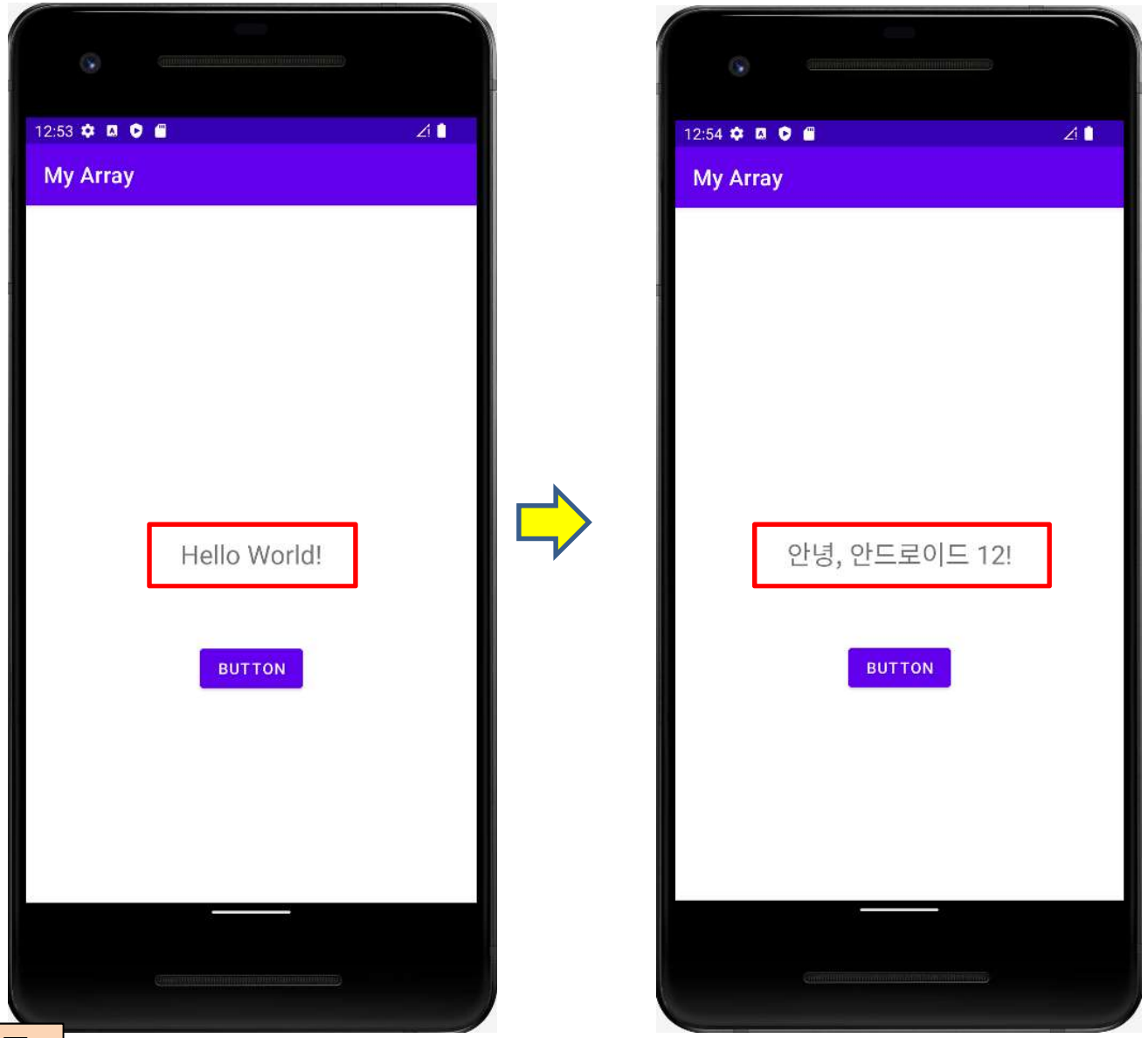
1차원 배열 객체 생성 (2/2)

- 배열 객체를 생성하는 3가지 방법
  - 타입+ArrayOf( )
    - 단, 타입에 String을 사용할 수 없음

```
val checked:Array<Boolean> = arrayOf(false, false, false)
// var checked = arrayOf(false, false, false)

val checked2:Array<Boolean> = Array( size: 3) {false}
// var checked2 = Array(3) {false}

val checked3:BooleanArray = booleanArrayOf(false, false, false)
// var checked3 = booleanArrayOf(false, false, false)

val intArr:IntArray = intArrayOf(1, 2, 3)
// var intArr = intArrayOf(1, 2, 3)
```

2개 타입은
호환되지 않음!

레이아웃 + 코드(event handler)



소스코드 – 1~2쪽

문자열 배열 리소스를 Array 객체에 저장

```xml
<resources>
    <string name="app_name">My Array</string>
    <string-array name="colors">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
    </string-array>
</resources>
```

```
val myButton: Button = findViewById(R.id.button)
myButton.setOnClickListener { it: View!
    val strArr:Array<String> = resources.getStringArray(R.array.colors)

    val s = "원소 개수=${strArr.size}, ${strArr[0]}, ${strArr[1]}, ${strArr[2]}"
    textView.text = s
}
```

원소 개수=3, Red, Green, Blue

```
myButton.setOnClickListener { it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val s = "첫 번째 원소 : ${strArr.get(0)}"
    textView.text = s
}
```

첫 번째 원소 : Red

[ ]을 사용해 원소를 참조하는 것을 선호.

소스코드 – 3쪽

배열 객체의 모든 원소 출력: for 문

```
myButton.setOnClickListener { it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val sb = StringBuilder( )
    for (element in strArr) {
        sb.append("$element ")
    }
    textView.text = sb.toString()
}
```

배열 원소

Red Green Blue

```
myButton.setOnClickListener { it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val sb = StringBuilder( )
    for (i in strArr.indices) {
        sb.append("${strArr[i]} ")
    }
    textView.text = sb.toString()
}
```

배열 인덱스

```
myButton.setOnClickListener { it: View!
    val strArr = resources.getStringArray(R.array.colors)
    textView.text = strArr.contentToString()
    // textView.text = Arrays.toString(strArr)
}
```

배열 원소를 출력하는
간단한 방법

소스코드 – 4쪽

forEach, forEachIndexed

```
myButton.setOnClickListener { it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val sb = StringBuilder( )
    strArr.forEach {
        element -> sb.append("$element ")
    }
    textView.text = sb.toString()
}
```

```
myButton.setOnClickListener { it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val sb = StringBuilder( )
    strArr.forEachIndexed {
        i, element -> sb.append("($i, $element) ")
    }
    textView.text = sb.toString()
}
```

(0, Red) (1, Green) (2, Blue)

소스코드 – 5쪽

배열의 원소 순환: iterator

```
myButton.setOnClickListener {  it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val sb = StringBuilder( )
    val iterator:Iterator<String> = strArr.iterator()
    // val iterator = strArr.iterator()    // 타입 생략
    while (iterator.hasNext()) {
        var element = iterator.next()
        sb.append("$element, ")
    }
    textView.text = sb.toString()
}
```

배열 객체를 iterator 객체로 변환

hasNext( ): iterator 객체에서 다음 원소가 있는지 확인
next( ): 다음 원소를 반환.

소스코드 – 5쪽

배열 객체 생성 및 정렬 (1/2)

```xml
<resources>
    <string name="app_name">My Array</string>
    <string-array name="colors">
        <item>,red</item>
        <item>!green</item>
        <item>Blue</item>
    </string-array>
</resources>
```

Array() 생성자를 사용한 배열 객체 생성

```
myButton.setOnClickListener {  it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val strArr2:Array<String> = Array( strArr.size){ i -> strArr[i] }
    var sb = StringBuilder()

    var a0:Char = (strArr2[0])[0]
    var a1:Char = (strArr2[1])[0]
    var a2:Char = (strArr2[2])[0]
    sb.append("$a0=${a0.code}, $a1=${a1.code}, $a2=${a2.code}")

    sb.append('\n')
    strArr2.sort()
    sb.append(strArr2.contentToString())
    textView.text = sb
}
```

```
,=44, !=33, B=66
[!green, ,red, Blue]
```

소스코드 – 6쪽

배열 객체 생성 및 정렬 (2/2)

```
myButton.setOnClickListener {  it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val strArr2:Array<String> = Array( strArr.size){ i -> strArr[i] }
    var sb = StringBuilder()

    val chArr:Array<Char> = Array(strArr2.size) { i -> (strArr2[i])[0] }
    for (ch in chArr)
        sb.append("$ch=${ch.code}, ")

    sb.append('\n')
    strArr2.sort()
    sb.append(strArr2.contentToString())
    textView.text = sb
}
```

문자 배열 객체로 변환

```
myButton.setOnClickListener {  it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val strArr2:Array<String> = Array( strArr.size){ i -> strArr[i] }
    var sb = StringBuilder()

    var a0:Char = (strArr2[0])[0]
    var a1:Char = (strArr2[1])[0]
    var a2:Char = (strArr2[2])[0]
    sb.append("$a0=${a0.code}, $a1=${a1.code}, $a2=${a2.code}")

    sb.append('\n')
    strArr2.sort()
    sb.append(strArr2.contentToString())
    textView.text = sb
}
```

소스코드 – 6쪽

배열 정렬

방식	배열 원본을 정렬	정렬 결과를 새 배열로 반환
오름차순	sort( )	sortedArray( )
내림차순	sortDescending( )	sortedArrayDescending( )

```
myButton.setOnClickListener {  it: View!
    val strArr = resources.getStringArray(R.array.colors)
    strArr.sort()
    textView.text = strArr.contentToString()
}
```

[Blue, Green, Red]

오름차순

내림차순

[Red, Green, Blue]

```
myButton.setOnClickListener {  it: View!
    val strArr = resources.getStringArray(R.array.colors)
    val strArr2 = strArr.sortedArrayDescending()
    textView.text = strArr2.contentToString()
}
```

소스코드 – 7쪽

정수 배열 리소스

```xml
<resources>
    <string name="app_name">My Array</string>
    <string-array name="colors">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
    </string-array>
    <integer-array name="intValues">
        <item>0</item>
        <item>1</item>
        <item>2</item>
    </integer-array>
</resources>
```

```
myButton.setOnClickListener {  it: View!
    val intArr:IntArray = resources.getIntArray(R.array.intValues)
    val sb = StringBuilder()
    intArr.forEach {  element -> sb.append("$element, ") }
    textView.text = sb.toString()
}
```

소스코드 – 8쪽

실수 배열 리소스 ➔ 문자열 배열 리소스

```xml
<resources>
    <string name="app_name">My Array</string>
    <string-array name="colors">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
    </string-array>
    <integer-array name="intValues">
        <item>0</item>
        <item>1</item>
        <item>2</item>
    </integer-array>
    <string-array name="numericalValue">
        <item>3.14</item>
        <item>290.0</item>
        <item>-0.24</item>
    </string-array>
</resources>
```

strings.xml: 실수 배열을 정의하는 태그가 없음.
➔ 문자열 배열 리소스를 정의하는 방식으로 실수 배열 원소를 정의.
➔ 문자열 배열 리소스를 가져와 String 타입을 Float 타입으로 변환

```
myButton.setOnClickListener {  it: View!
    val strArr:Array<String> = resources.getStringArray(R.array.numericalValue)
    val floatArr:Array<Float> = Array(strArr.size){ i -> strArr[i].toFloat()}
    val sb = StringBuilder()

    floatArr.forEach {  element -> sb.append("$element, ") }
    textView.text = sb.toString()
}
```

소스코드 – 9쪽