자바 문자열

학습 목표

- 문자(char, Character)와 문자열(String)
- StringBuilder와 StringBuffer

문자와 문자열

미션 완료
나의 팀
슬라이드

- 문자열(String)은 자바 프로그램이 실행되는 동안 가장 많이 생성되는 객체이다.
- 문자열은 객체이지만 각각의 문자의 나열로 구성된다.
- char capitalA = ‘A’; // 문자
- String a = “abc”; // 문자열 == 문자의 배열
- String b = new String(“abc”); // 이와 같이 구현하지 마라.

new String() 구현시 차이

구현 코드

```
public void testString() throws Exception {
    String a = "abc";
}

public void testNewString() throws Exception {
    String b = new String("abc");
}
```

컴파일된 class 코드

```
 public void testString() throws java.lang.Exception;
    Code:
       0: ldc           #89                 // String abc
       2: astore_1
       3: return

  public void testNewString() throws java.lang.Exception;
    Code:
       0: new           #33                 // class java/lang/String
       3: dup
       4: ldc           #89                 // String abc
       6: invokespecial #93   // Method java/lang/String."<init>":(Ljava/lang/String;)V
       9: astore_1
      10: return
```

문자열(String)은 자바 프로그램이 실행되는 동안 **가장 많이 생성되는 객체**이다.
문자열은 객체이지만 각각의 **문자의 나열**로 구성된다.

String API에 대한 학습 테스트

```
public class StringTest {
    @Test
    public void 문자열_길이_구하기() throws Exception {
        String name = "박재성";
        assertThat([TODO 길이 구하기]).isEqualTo(3);
    }

    @Test
    public void 문자열_더하기() throws Exception {
        String name = "박재성";
        String welcome = "안녕!";
        assertThat([TODO 문자열 더하기]).isEqualTo("안녕!박재성");
    }

    @Test
    public void 문자열을_문자_단위로_출력() throws Exception {
        String name = "박재성";
        // String의 각 문자를 배열로 가져올 수 있는 API 활용해 구현 가능
        [TODO 문자열 출력]
    }

    @Test
    public void 문자열_뒤집기() throws Exception {
        String name = "박재성";

        // String의 각 문자를 배열로 가져올 수 있는 API 활용해 구현 가능
        [TODO 문자열 뒤집기]

        assertThat(reverseName).isEqualTo("성재박");
    }
}
```

미션 완료
나의 팀
슬라이드

StringBuilder API

String + String vs StringBuilder

- String은 불변하기(immutable)이기 때문에 String과 String을 더하면 새로운 String 객체를 생성한다. 따라서 String과 String을 더하는 시점에 메모리 할당과 메모리 해제가 계속 발생한다.
- StringBuilder는 String과 다르게 기존 데이터에 새로운 데이터를 더하는 방식을 취하기 때문에 속도가 더 빠르다.
- 따라서 긴 문자열을 더하는 상황이 발생하는 경우 StringBuilder를 활용해 구현한다.

immutable

- Immutable(불변)이란 객체를 생성한 후 상태를 변경할 수 없는 것을 의미한다.
- String 객체는 문자열의 상태 값을 변경할 수 없기 때문에 immutable 객체라 한다.

StringBuilder API 학습 테스트

> StringBuilder를 활용해 `코드스쿼드 화이트 레벨 수강생은? 18 명이다.` 문자 출력한다.
>
> 문자열을 더할 때 더하기(+) 연산자를 사용하지 않아야 한다.

```
import org.junit.Test;

import static org.assertj.core.api.Assertions.assertThat;

public class StringBuilderTest {
    @Test
    public void append () {
        assertThat(createMessage(14)).isEqualTo("코드스쿼드 백엔드 수강생은? 14 명이다.");
    }

    private String createMessage(int numberOfClass) {
        StringBuilder sb = new StringBuilder();
        // TODO append() 메소드 활용해 요구사항 구현
        return sb.toString();
    }
}
```

- StringBuilder를 사용하는 경우 + 연산을 통해 문자열을 생성하지 않고 append를 활용해야 한다.