**로고학습일지**

**kt ds University 자바 기반의 데이터 사이언티스트 양성과정**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학습일시 | 2018. 07. 18 (수) | 장소 | kt ds University B관 201호 | **시 간** | 09:00~18:00 |
| 학습범위 | 자바 | | | | |
| 작 성 자 | 유재헌 | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 | **01. 주석**  **02. String**  **03. Type Casting**  **04. Array & List & Map** |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| **01. 주석**  - 주석 : 코드에 흐름이나 어떤 기능으로 동작하는지 등 복잡한 코드에 대해 설명 기입할 때 사용한다.  - 주석은 3가치 형태로 작성할 수 있다.  C:\Users\Main\AppData\Local\Microsoft\Windows\INetCache\Content.Word\주석.png  **02. String**  🡪 JAVA가 제공하는 char 데이터 타입은 문자 하나를 저장하는데 쓰인다. 즉, “안녕하세요”라는 여러 개의 문자를 저장하려면 배열을이라는 특수한 키워드를 사용해야한다.  🡪 그러나, 모든 문자열들을 배열을 통해 저장할 수 없으므로 JAVA는 String이라는 문자열을 저장할 수 있는 특수한 타입을 제공한다.    🡪 char 타입으로 데이터를 저장하려면 ‘’(홀따옴표)를 사용하지만,  String 타입으로 데이터를 저장하려면 “”(쌍따옴표)를 사용한다.    🡪 어떤 프로그램이건 문자열은 데이터로서 아주 많이 사용된다. 그렇기 때문에 문자열을 생성하는 방법과 추출, 비교, 찾기, 분리, 변환 등을 제공하는 메소드를 잘 익혀두어야 한다.  🡪 자바의 문자열은 java.lang 패키지의 String 클래스의 인스턴스로 관리된다. 소스상에서 문자열 리터럴은 String 객체로 자동 생성되지만, String 클래스의 다양한 생성자를 이용해서 직접 String 객체를 생성할 수도 있다.  - 문자열에 관련한 메소드  1) 문자추출(charAt())  🡪 charAt() 메소드는 매개값으로 주어진 인덱스의 문자를 리턴한다. 여기서 인덱스란, 0에서부터 “문자열길이 -1”까지의 번호를 말한다.  🡪 “자바 프로그래밍” 문자열은 다음과 같이 인덱스를 매길 수 있다.  🡪 charAt(3)은 3인덱스 위치에 있는 문자를 말한다. 즉, ‘프’문자가 해당된다.  2) 문자열 비교(equals())  🡪 Primitive Type(byte, short, int, long, float, double, char) 변수의 값을 비교할 때에는 == 연산자를 사용한다. 하지만 문자열을 비교할 때에는 ==를 사용하면 원하지 않는 결과가 나올 수 있다.  🡪 자바는 문자열 리터럴이 동일하다면 동일한 String 객체를 참조하도록 되어 있다. 그래서 strVar2와 strVar3은 동일한 String 객체를 참조한다.  🡪 그러나 strVar1은 new 연산자를 생성된 다른 String 객체를 참조한다.  🡪 이 경우 변수 strVar1과 strVar2의 == 연산은 false를 산출하고 strVar2와 strVar3의 == 연산은 true를 산출한다. == 연산자는 각 변수에 저장된 번지를 비교하기 때문에 이러한 결과가 나온다.  🡪 만약 두 String 객체의 문자열만 비교하고 싶다면 == 연산자 대신에 equals() 메소드를 사용해야한다.  🡪 사용방법은 문자열.equals(비교대상문자열)이다.  3) 문자열 대/소문자 상관없이 비교(equalsIgnoresCase())  - 문자열 비교할 때 대소문자 상관없이 비교해주는 메소드이다.    🡪 name 변수의 값에는 “Yoo” 들어있고, 조건식에 name.equalsIgnoreCase(“yoo”) 문자열 비교할 때 true가 되어 {} 안에 있는 코드가 실행된다. 그 이유가 대/소문자를 무시하고 비교하기 때문에 “Yoo”와 “yoo”를 동일한 문자열로 인식한다.  4) 문자열 찾기(indexOf())  - indexOf() 메소드는 매개값으로 주어진 문자열이 시작되는 인덱스를 리턴한다. 만약 주어진 문자열이 포함되어 있지 않으면 -1을 리턴한다.  🡪 index 변수에는 3이 저장되는데, “자바 프로그래밍”에서 “프로그래밍” 문자열의 인덱스 위치가 3이기 때문이다.      🡪 indexOf() 메소드는 if의 조건식에서 특정 문자열이 포함되어 있는지 여부에 따라 실행 코드를 달리할 때 자주 사용한다. -1 값을 리턴하면 특정 문자열이 포함되어 있지 않다는 뜻이다.  5) 문자열 길이(length())  - length() 메소드는 문자열의 길이(문자의 수)를 리턴한다.  🡪 length 변수에 8이 저장된다. subject객체의 문자열 길이는 공백을 포함해서 8개이기 때문이다.  C:\Users\Main\AppData\Local\Microsoft\Windows\INetCache\Content.Word\String3.png  6) 문자열 대치(replace())  - replace() 메소드는 첫 번째 매개값인 문자열을 찾아 두 번째 매개값인 문자열을 대치한 새로운 문자열을 생성하고 리턴한다.  🡪 String 객체의 문자열은 변경이 불가한 특성을 갖기 때문에 replace() 메소드가 리턴하는 문자열은 원래 문자열의 수정본이 아니라 완전히 새로운 문자열이다.  🡪 따라서 newStr 변수는 다음 그림과 같이 새로 생성된 “JAVA 프로그래밍” 문자열을 참조한다.  7) 문자열 잘라내기(substring())  - substring() 메소드는 주어진 인덱스에서 문자열을 추출한다.  - substring() 메소드를 매개값의 수에 따라 2가지 형태로 사용한다.  1) substring(int beginIndex, int endIndex)  🡪 주어진 시작과 끝 인덱스 사이의 문자열을 추출한다.  2) substring(int beginIndex)  🡪 주어진 인덱스 이후부터 끝까지 문자열을 추출한다.  🡪 위 코드에서 firstNum 변수값은 “900325”이고, secondNum 변수값은 “1234567”이다.      🡪 number.substring(0,6)은 인덱스 0(포함) ~ 6(제외) 사이의 문자열을 추출하는 것이고,  number.substring(7)은 인덱스 7 이후의 문자열을 추출한다.  8) 알파벳 소/대문자 변경(toLowerCase(), toUpperCase())  - toLowerCase() 메소드는 문자열을 모두 소문자로 바꾼 새로운 문자열을 생성한 후 리턴한다.  - toUpperCase() 메소드는 문자열을 모두 대문자로 바꾼 새로운 문자열을 생성한 후 리턴한다.  🡪 lowercase 변수는 새로 생성된 “java programming” 문자열을 참조하고,  uppercase 변수는 새로 생성된 “JAVA PROGRAMMING” 문자열을 참조한다.  🡪 즉, 원래 “Java Programming” 문자열이 변경된 것이 아니다.  🡪 toLowerCase()와 toUpperCase() 메소드는 영어로 된 두 문자열을 대소문자와 관계없이 비교할 때 주로 이용된다.  9) 문자열 앞뒤 공백 잘라내기(trim())  - trim() 메소드는 문자열의 앞뒤 공백을 제거한 새로운 문자열을 생성하고 리턴한다.  - trim() 메소드는 앞뒤의 공백만 제거할 뿐 중간의 공백은 제거하지 않는다.  🡪 trim() 메소드를 사용한다고 해서 원래 문자열의 공백이 제거되는 것은 아니다.  String7  10) 문자열의 첫 부분만 검사(startsWith()),  문자열의 끝 부부만 검사(endsWith()),  문자열에서 원하는 문자열 포함 확인(contains())  - startsWith()는 문자열의 첫 부분만 검사할 때 사용하는 메소드이다.  - endsWith()는 문자열의 끝 부분만 검사할 때 사용하는 메소드이다.  - startsWith()와 endsWith()는 문자열이 ~으로 시작하는지 혹은 ~으로 끝나는지 검사할 때 유용하게 사용될 수 있다.  - contains()는 특정 문자열에서 원하는 문자가 포함되어 있는지 확인해야 될 때 사용하는 메소드이다.  🡪 웹 브라우저 URL창에 해당 주소 입력했을 때 접속하려고 하는 사이트의 문자열이 맞는지 검사할 때 유용하다.    11) 특정 문자열 파싱(split())  - split()은 특정문자를 기준으로 문자열을 잘라서 배열에 넣어주는 아주 편리한 메소드이다.  - split() 메소드는 String 값을 특수기호로 끊어서 배열에 저장시켜준다.  🡪 String 타입의 str객체에 “Java C C++ C# Python”이라는 문자열이 들어있다. 여기서 문자열.split()메소드를 사용하여 공백문자를 구분자로 두어 Java, C, C++, C#, Python이라는 단어들을 words 배열이 넣었다.  🡪 그리고 반복문을 실행했을 때 공백문자를 파싱하여 오른쪽에 있는 결과가 나오는 것을 알 수 있다.  **03. Type Casting**  - 형 변환은 현재 저장되어야할 값이 변수의 타입과 맞지 않을 경우, 값을 버리거나 더해서 해당 변수에 값이 들어갈 수 있게 바꿔주는 것을 말한다.  - Type Casting 의 2가지 종류  1) 묵시적 형 변환  🡪 크기가 작은 타입에서 큰 타입으로 저장될 때 발생한다.  🡪 자바는 묵시적 형 변환을 자동으로 변환해 저장할 수 있도록 지원해준다.      2) 명시적 형 변환  🡪 타입을 자동으로 변환해주는 묵시적 형 변환과 반대로 개발자가 형 변환을 직접 해야하는 경우도 있는데 이를 명시적 형 변환이라고 한다.  🡪 명시적 형 변환 사용할 때의 경우  (1) 나누기의 결과처럼 계산만을 위해 잠시 형을 바꿔주고 싶을 때  (2) 크기가 큰 타입의 변수에서 크기가 작은 타입의 변수로 값을 옮길 때    **04. Array & List & Map**  - 같은 타입의 변수를 나열하여 여러 개의 값을 저장할 수 있도록 하는 것이 배열이다.  - 배열의 선언 방법은 다음과 같다.    - JAVA는 JAVA Style과 C Style의 배열 선언 방법 모두를 지원한다. 그러나 C Style은 잘 사용하지 않는다.    - 아래 사진과 같이 scores라는 배열 변수는 정수 int 값을 4개를 저장할 수 있는 변수이다.  - 배열 변수는 각 배열 값을 0번부터 시작해 선언한 숫자만큼 할당 시킬 수 있습니다. 여기에 사용된 0, 1, 2, 3 값들은 ***인덱스*** 라고 부릅니다.  - 인덱스는 배열 선언 개수만큼 사용할 수 있습니다.  - 주의할 점은 scores 배열 변수는 int 타입으로 선언되었으므로 int에 맞는 값만 넣을 수 있다.  즉, 정수만 넣을 수 있다.  - 배열에 값을 할당하기 위해 인덱스를 사용한 것처럼 배열의 값을 알기 위해서 인덱스를 사용한다.  - 배열의 단점은 배열의 길이를 미리 명시하기 때문에 고정길이를 갖게 된다. 그래서 배열의 크기를 줄이거나 늘리거나 할 수 없다. 아래의 코드는 에러가 발생한다.  - 배열의 크기를 원하는 대로 추가/삭제 할 수 있도록 지원하는 것은 List이다.  - 배열의 잘못된 사용 방법 코드를 List로 바꾸면 에러가 발생하지 않는다.  - List는 배열의 크기 제한이 없다. add()만 사용하면 배열에 값을 넣을 수 있다.  - List에서 값을 추출할 때는 get(index)를 사용한다.  - List의 개수를 확인하고 싶을 때 size()를 사용한다.  - List의 값을 삭제하려면 remove(index)를 사용한다.    - List는 클래스가 아닌 인터페이스로 되어 있다. 인터페이스는 객체 생성이 불가능 하므로 List의 하위의 구현 클래스로 객체를 생성하는데 그 중 ArrayList 클래스가 있다.  - List를 사용할 때 주의할 점은 Java에서 Primitive Type이라 불리는  byte, short, int, long, float, double, boolean, char 타입을 사용할 수 없다.    - List에서 Primitive Type을 사용하려면 Java에서 미리 준비한 Wrapper Class를 사용해야한다.    - 배열의 인덱스를 숫자를 포함한 모든 타입으로 지정할 수 있는 것이 Map이다.  - Map은 키와 값을 쌍으로 가지는 특성이 있다.  - Map에 키와 값을 추가하려면 put(Key, Value)을 사용한다.  - 해당 키의 값을 가져오려면 get(Key)를 사용한다.    - 해당 Key값으로 데이터 삭제를 하려면 remove(Key)를 사용한다.  - Map을 사용할 때 주의할 점은 Key는 중복되지 않는다는 것이다.  - 같은 Key를 등록할 때 기존의 Key는 갱신된다.  - Map은 클래스가 아닌 인터페이스로 되어 있다. 인터페이스는 객체 생성이 불가능 하므로 Map의 하위의 구현 클래스로 객체를 생성하는데 그 중 HashMap 클래스가 있다.  - Map을 사용할 때 주의할 점은 Java에서 Primitive Type이라 불리는  byte, short, int, long, float, double, boolean, char 타입을 사용할 수 없다.    - List와 Map의 차이점  🡪 List는 배열처럼 인덱스마다 값을 넣는 형식이고 Map은 인덱스가 없고 해당 Key와 Value로 넣는 형식이다.  🡪 List는 순서가 있지만, Map는 순서가 없다.  🡪 List는 중복된 값을 넣을 수 있지만, Map은 중복이 없다. 즉, 중복되는 Key와 Value를 넣으면 기존의 Key의 값은 갱신된다. |
|  |

|  |  |
| --- | --- |
| 특이사항 |  |