**로고학습일지**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학습일시 | 2018. 07. 25 (수) | 장소 | kt ds University B관 201호 | **시 간** | 09:00~18:00 |
| 학습범위 | UI/UX | | | | |
| 작 성 자 | 류복기 | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 | -메소드 |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| 오늘은 저번 시간에 이은 클래스와 메소드에 대해 더 깊이 알아보고자 합니다.  먼저 접근제한자입니다.  접근 제한자에는 public, protected, default, private이 있습니다.   * 접근 가능한 범위를 나타낸 표  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 접근 제한자 | 클래스 내부 | 동일 패키지 | 하위 클래스 | 그 외의 항목 | | Public | O | O | O | O | | Protected | O | O | O |  | | Default | O | O |  |  | | Private | O |  |  |  |  * 앞에 접근제한자를 붙이지 않을 경우에는 default로 선언됨.   보통 멤버변수는 private을 많이 사용하고 메소드는 public을 많이 사용합니다.  그렇다면 멤버변수에 왜 private을 많이 쓸까??  외부에서 접근하여 마음대로 수정되는 것을 막기 위해 입니다.  예를 들어,  String name; 이라는 멤버변수가 있습니다.  이것을 private으로 쓰지 않으면,  외부에서 class를 호출하여 name의 값을 마음대로 바꿀 수 있습니다.  따라서 멤버변수에는 해당 클래스 안에서만 접근이 가능한 private을 사용합니다.  하지만 이렇게 private으로 쓰게 되면, 다른 클래스에서 멤버변수는 사용하지 못합니다.  그러면, 이러한 문제를 어떻게 해결할까요?  이 때 쓰는 방법이 Setter와 Getter 메소드를 사용하는 것입니다.  멤버변수 아래에 커서를 대고 alt + shift + s 버튼을 누릅니다.  창 하나가 뜰겁니다.    여기서 Generate Getters and Setters를 클릭합니다.  만약에 getter와 setter 메소드를 생성했다면, 이미 생성되었다는 창이 뜰겁니다.    이렇게 쓰는 이유는 세가지로 볼 수 있습니다.  첫째, 멤버변수에게 값을 할당하기 위함.  둘째, 값을 필터링 하기 위함. -> 예) setter 안에 filltering을 적용하여 값을 골라서 받음  셋째, 외부로 나가지 못하게 하기 위함입니다. 이를 encapsulation, 캡슐안에 넣는다고 합니다.  특히 이는 Reference type에 매우 중요합니다!  원시타입과 다르게 reference Type은 값이 저장되는 것이 아니라 메모리의 주소값이 저장되기 때문에,  Private을 쓰더라도 class안의 값이 수정될 위험이 있습니다.  예를들어, Array를 예로 들어봅시다.  Array를 멤버변수로 가진 Objectclass를 호출했습니다.  Main Class의 array에 새로운 값을 추가할 때, Main Class의 array와  Object Class의 List가 같은 메모리 값을 가지기 때문에,  private으로 선언한 List의 값도 추가되게 됩니다.  이를 막기 위해서, setter와 getter를 통해 encapsulation을 합니다.  코드 예시    빨간색 테두리의 getHealthPointList부분은 Getter 메소드입니다.  순서대로 보면,  healthPointListCopy라는 새로운 이름의 리스트 객체를 만듭니다.  그리고 addAll이라는 리스트안의 메소드를 통해,  멤버변수인 this.healthPointList의 각 인덱스에 해당하는 값들을 healthPointListCopy의 List에 추가합니다.  마지막으로, healthPointListCopy라는 리스트를 리턴합니다.  멤버변수를 리턴할 경우 멤버변수의 주소값이 같기 때문에 멤버변수 값의 변경위험이 있습니다.  따라서, get메소드의 경우 새로운 리스트를 만들어 리턴해 줍니다.  파란색 테두리의 setHealthPointList부분은 setter 메소드입니다.  순서대로 보면,  멤버변수에 새로운 주소값의 리스트를 만듭니다.  그리고 이 멤버변수의 값에 argument에 입력한 리스트의 값들을 멤버변수 List에 추가합니다.  이제, 멤버변수에 입력한 리스트의 값들이 들어가게 됩니다.  하지만, Main의 리스트와 멤버변수 리스트의 주소값은 다르게 설정됩니다.    다음으로 위 그림의 this. 에 대해 알아보겠습니다.  만약, this.을 사용하지 않고 healthPoint = healthPoint; 라고 쓰게 되면,  빨간색 테두리에 있는 int healthPoint라는 argument를 같은 int healthPoint에 할당하게 됩니다.  하지만 우리는 멤버변수에 이 argument를 할당하고 싶습니다.  따라서, healthPoint 앞에 this.을 붙임으로써 클래스의 멤버변수로 표현할 수 있습니다.   * 생성자   우리는 앞서 멤버변수를 초기화 하지 않았습니다.  이는 생성자를 통해 할 수 있습니다.  예를 들어, Scanner를 사용한다고 해봅시다.  Scanner input = new Scanner(System.in)의 형태로 input이라는 객체를 만들 것입니다.  이 때, new 뒷부분이 생성자에 해당합니다.  여기서 new는 메모리 확보에 해당합니다.  New의 뒷부분은 객체를 만드는 것에 해당합니다.  종합하여 말하면, Scanner클래스를 객체화시켜 type이 Scanner인 input이라는 객체에 할당하는 것입니다.  Public class Calculator(){  Public Calculator(){  }  }  이 예제에서 생성자는 public Calculator() {}에 해당하는 부분입니다.  생성자는 멤버변수의 아래에다 만듭니다.  그리고 생성자는 객체 생성시 단 한번만 호출됩니다.  생성자 메소드를 만드는 방법은 다른 메소드와 조금 다릅니다.  메소드와 다르게 Return type이 생략되어 있는 것처럼 보입니다.  하지만 반환타입이 없는 것이 아니라 이름이 없는 것입니다.  반환타입은 클래스의 이름에 해당합니다. |