**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 | Java Programming | | | | |
| 작 성 자 | 권석현 | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 | 오버로딩, 오버라이딩, 인터페이스, abstract |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| 생성자 만들 때 파라미터 추가해서 만듬 -> 생성자 오버로딩  메소드도 아귀먼트의 종류나 갯수에 따라서 오버로딩 가능.  -> 알고있으면 좋으나 생각보다 쓰는경우 많지않다.  오버로딩과 오버라이딩의 차이? -> 면접 시 많이 묻는다.(기술면접)  utility(도구, 틀)짤때 많이 사용.  메소드 오버로딩은 아래 조건 충족  1. 같은 리턴타입  2. 같은 메소드 이름  3. 다른 파라미터 타입/개수  - static (utility를 만들기 위해 필요)  JVM 역할  1. java app 실행  2. Memory 관리 (static과 관련있음)  .java -> Compile -> .class(변수들 존재) 실행 중에 변수, 결과를 메모리에 담음(rumtime)  static -> 실행 중 언제든지 가져다 쓰고 싶은 변수, 메소드, 클래스 앞엔 static 붙인다.  그러면 실행 중에 메모리를 만드는게 아니라 자바 애플리케이션이 부팅 될 때 메모리에 올린다.  실행 중 필요하면 메모리에 올린다 <--> 자바 애플리케이션 처음 실행될때 올리느냐의 차이.  메모리 세개로 나뉨  heap, stack, static  heap 영역 -> runtime때 만들어지는 변수나, 메소드 결과, 아큐먼츠값들 저장,관리  static -> 공유 메모리 static붙은거 올라옴(독립 보장 힙 스택과 완전 분리)  -> heap에서 static 접근 가능하지만 그 반대는 안된다.  static끼리는 통신 가능.  멤버변수 -> Instance Variables(fields)  멤버 메소드 -> Instance Method  class -> 추상적 정의 ex) 설계도  class가 구체적이 될려면 Instance(객체)(구체적 실체)로  static이 붙으면 Instance의 것이 아닌 class의 것이 된다.  즉 객체 생성을 안하고도 가져올 수 있다.(객체 불필요) -> static은 독립된 메모리는 사용하고 있으니까.  static class variables  static class Methods  instance변수 메소드는 실행중에 할당, static은 부팅될때 메모리에 올린다.  클래스는 실행될 때 메모리에 올라감. 아닐땐 안올라감.  즉 객체를 생성해야 올라감.  면접시 static none static의 차이점을 물어봄. 자주나옴  static은 자원 공유 가능. none static은 실행중일 때 잠깐 만들어지고 사라진다,  static에서 none static을 호출할수 없지만 none static에서 static을 호출할 수 있다.  static은 정말 필요한 경우 아니면 사용 x 메모리낭비심함.  기울어진 글씨체 -> 스테틱  클래스명.~~ -> static  static 부팅될 때 메모리에 올라가므로 초기값 반드시 필요  <Package>  // 여러개의 클래스를 효율적으로 관리하기위해 용도 종류별로 구분하여 정리함.  패키지명은 홈페이지 주소 역순으로 한다.  ex ) cafe.daum.net -> net.daum.cafe  다른 패키지의 class를 사용하기 위해선 반드시 import가 필요.  자주 사용되는 클래스에 대해서는 import 생략 가능  java에서 미리 만들어 놓은 클래스 -> java.lang 패키지에 넣으면 됨.  패키지가 다르면 import필요, 같은 패키지면 필요없음.  사용하려는 클래스명 외부 클래스명과 다르게 해야됨.  //Ctrl + Shift + O --> Auto Import  패키지명에는 대문자 사용x, 예약어 x, 소문자로만 .으로 구분  <Interface>  기능 정의 ,기능에대한 약속  ex) public interface MemberService {  public class MemberServideImpl implement Member{  1. 접점  2. 다형성 -> 기능은 같지만 동작들이 다른 tv들을 만들 수 있다.  MemberService memberService = new MemberServiceImpl ( );  MemberService memberService2 = new MemberServiceImpl2 ( );  MemberService memberService3 = new MemberServiceImpl3 ( );  //인터페이스는 바디 정의 x 형태만 정의  // Sub class is a Super Class  // Sub class ==> LGTV  // Super Class ==> TV  단점 : 인터페이스에 없는 기능을 추가 할 수 없다.  추가해도 그 인터페이스를 사용하는 것들도 다 추가해야함.  \*\* 상속을 해야함.  <상속>  상속 ==> 확장  1. Interface를 상속(확장) 가능.  2. AbstractClass, Class 상속(확장) 가능  3. interface --> implement  class --> extends  abstract class  똑같은 부분이 있고 중간에 다른 부분들이 생기면  그부분을 추상화 한다. 각자 정의해서 사용할 수 있게.  어떤 절차가 있고 달라지는 부분 코드를 분리 해 냄.  interface implements, extends, abstract class  가 무엇인지 정도 파악하기.  interface -> 기능을 정의해놓고 그 기능을 정의하는 여러 클래스들을 만들어 접점을 만든다.(다형성)  extends -> 상속, 확장 의미, 클래스, 인터페이스 둘다 extends 가능.  abstact class -> 절차가 정해져 있고 부분부분만 다르다면 바뀌는부분만 추상화 시켜서 분리해서 그 클래스를 상속받아서 구현시킨다.  다른사람이 짠 코드를 많이 봐야함. |

|  |  |
| --- | --- |
| 특이사항 |  |