JavaScript 와 Java 의 차이점

1. 각 변수의 앞에 자료형을 붙여줘야 한다.
(int a, string b)

2. 메소드에도 반환값을 지정한다.
(public void add(), public string)

3. main 메소드를 실행을 한다

4. 객체지향프로그래밍에 대한 이해가 더 깊고 풍부해야 코드를 잘 짤 수 있다.
(JS에서는 흉내만 낸 것 같다)

5. overriding - 상속받은 메소드를 자식 클래스에서 재정의 하는 것
(메소드 이름이 같아야 함, 메소드 리턴타입이 같아야 함, 파라미터도 같아야 함)
@Override 는 annotation 이지만 오버라이딩이 된지 안된지 검증하는 기능을 함, 즉 실제로 실행되지 않았다면 에러가 뜸

6. overloading - 같은 이름을 갖고 있지만, 서로 다른 매개변수 형식을 가지고 있는 메소드를 여러개 정의
(메소드 이름이 같아야 함, 파라미터의 개수 또는 자료형이 달라야 함)

오버로딩 - 기존에 없는 새로운 메소드를 추가하는 것

오버라이딩 - 상속받은 메소드를 재정의 하는 것

구분	Overriding	Overloading
접근 제어자	부모 클래스의 메소드의 접근 제어자보다 **더 넓은 범위의 접근 제어자**를 자식 클래스의 메소드에서 설정할 수 있다.	**모든 접근 제어자**를 사용할 수 있다.
리턴형	동일해야 한다.	달라도 된다.
메소드명	동일해야 한다.	동일해야 한다.
매개변수	동일해야 한다.	달라야만 한다.
적용 범위	**상속관계**에서 적용된다.	**같은 클래스** 내에서 적용된다.

7. 패키지 - 하나의 클래스 안에서 같은 이름의 클래스들을 사용하기 위한 방법(이름의 충돌을 막기 위한)
패키지 명은 일반적으로 클래스를 제작한 개인이나 단체가 소속된 웹사이트 도메인을 이용 (중복을 피하기 위해)

8. abstract - 상속을 강제하기 위해서 사용함
메소드의 가장 기본적인 ? 필요한 부분들을 정의를 해놓고 무조건 상속을 해서 사용하게 만듦

추상클래스의 조건

- 추상클래스(abstract)는일반클래스보다 조금더 추상적은 개념의 것을 정의한다.
- 추상클래스(abstract)의 추상 메서드와 그냥 메서드, 멤버필드, 생성자를 정의 할 수 있지만, 추상메서드는 내용을 정의 할 수 없다.
- 추상클래스(abstract)의 추상메서드를 정의하면 추상클래스를 상속받은 클래스에서는 반드시 추상메서드를 Override 하여 정의하여야 한다.
- 추상클래스(abstract)는 abstract class [클래스명] 으로 정의한다.
- 추상클래스(abstract)의 추상메서드는 [접근제한자] abstract [return 자료형] [메서드이름]() 으로 정의한다.
- 추상클래스(abstract)는 extends 로 상속받는다.
- 추상클래스(abstract)는 다중 상속이 안된다.

```
abstract class Animal {
	String animal_name;

	Animal(String name) {
		animal_name = name;
	}
	public abstract void cry();

	public abstract void behavior();

}

class Tiger extends Animal {

	public Tiger(String name) {
		super(name);
	}

	@Override
	public void cry() {
		// TODO Auto-generated method stub
		System.out.println("어흥");
	}

	@Override
	public void behavior() {
		// TODO Auto-generated method stub
		System.out.println("우측으로 빠르게 움직인다.");
	}

}
```

9. final - 상속을 금지하는 것
abstract 처럼 앞에 final 을 붙이면 실행
하지만 자주 사용되지는 않는다

10. interface - 반드시 인터페이스에서 강제하고 있는 메소드를 구현해야 한다.
공동 작업을 할 때, 메소드의 표준화를 시키기 위해서 많이 사용되며 서로 다른 클래스를 연결할 때도 사용

인터페이스(interface)의 조건

- 인터페이스(interface)는 추상클래스보다 훨씬 극단적이고 제한적인 성격을 가집니다.
- 인터페이스(interface)는 멤버필드와 추상메서드만 정의 할 수 있습니다.
- 인터페이스(interface)를 상속받을려면 extends를 사용하면 안되고 implements 를 사용하여야 합니다.
- 인터페이스(interface)는 다중상속이 가능합니다.
- 인터페이스(interface)의 추상메서드는 일반클래스의 메서드 형식과 같지만 몸통을 가질 수 없습니다.
- 인터페이스(interface)의 선언은 interface [이름]으로 합니다.