업캐스팅과 다운캐스팅

1. 업캐스팅은 **상속관계에서 자손타입의 참조변수가 조상타입의 참조변수로 변환하는 것을 말한다.**

2. 다운캐스팅은 **상속관계에서 조상타입의 참조변수가 자손타입의 참조변수로 변환하는 것을 말한다.**

3. 다운캐스팅 전제 조건

첫째, **상속관계를 만들어야 한다.**

둘째, **다운캐스팅을 하기전 먼저 업캐스팅을 해야 한다.** 이유는 **업캐스팅을 한것에 한해서만 다운캐스팅을 허용한다.**

셋째, **캐스팅 연산자를 사용하여 명시적인 다운캐스팅을 해야 한다.**

instanceof 연산자

1.**참조변수가 참조하고 있는 인스턴스의 실제 타입을 알아보기 위해 instanceof 연산자를 사용**한다.

2.주로 **if 조건문을 사용**하며, instanceof 연산자의 **왼쪽에는 참조변수**를 **오른쪽에는 타입(클래스명)이 피연산자로 위치**한다.

3.연산의 결과가 **true로 나오면 참조변수가 검사한 타입으로 형변환이 가능**하다는 뜻이다.

자바의 추상클래스 특징

1. **추상클래스**는 **abstract 키워드로 정의**한다.

2. **추상클래스**는 **객체 생성을 할 수 없다.**

3. **추상클래스**에는 **추상메서드가 올수 있다. 추상메서드는 {}가 없고 ,실행문장이 없다.** 그러므로 **호출이 불가능**하다.

4. **추상클래스를 상속받은 자식클래스에서 부모 추상클래스의 추상메서드를 오버라이딩 해야 한다.**

5. 추상클래스는 멤버변수와 메서드를 가질 수 있다.

final 키워드 특징

1. **변수**를 final로 정의하면 **수정할 수 없는 변수 즉 상수**가 된다.

2. **클래스**를 final로 선언하면 **더 이상 상속을 허락하지 않는다.**

3. **메서드**를 final로 선언하면 **더 이상 오버라이딩을 허락하지 않는다.**

자바의 상속방법

1. 클래스에서 인터페이스를 구현: **implements**

2. 클래스에서 부모클래스 상속: extends

3. **인터페이스에서 인터페이스 상속: extends**

자바의 인터페이스 특징

1. 인터페이스는 **interface 예약어로 정의**한다.
2. 인터페이스는 **하나 이상의 부모로부터 다중 상속이 가능**하다.
3. 인터페이스에는 **public static final로 인식되는 클래스(정적)상수만 올수 있다.**
4. 인터페이스는 **public abstract 로 인식되는 추상메서드가 올수 있다.**
5. 인터페이스는 **자식클래스에서 implements 키워드로 구현한다. 상속받은 자식클래스에서 부모 인터페이스의 추상메서드를 반드시 오버라이딩을 해야 한다. 그래야 자식클래스 객체 생성이 가능하다.**
6. 인터페이스는 **객체 생성을 못한다.**

자바의 예외 처리

1. 자바에서 실행 **예외란 프로그램이 실행되는 동안 예상치 못했던 에러를 의미** 한다. 자바는 컴파일 할때는 예외 처리 코드를 검사하지 않기 때문에 아무 에러가 없다가 실행될때 예기치 않는 에러가 발생되면 예외 처리 클래스를 통해서 예외를 처리한다.

2. 자바에서 대표적인 **예외 에러는 정수 숫자를 0으로 나눌때 예외를 발생**한다.

3. **예외 처리 방법**

첫째, **try ~ catch문**

```
try{
  정상구문이 실행되다 예외가 발생하면 제어가 catch문으로 옮겨진다.
}catch(예외 처리 클래스 e){
 예외 처리할 문장;
}//catch문에서 예외를 예외 처리클래스로 처리한다.
```

둘째, **throw 키워드**

throw는 예외를 일부러 발생시킬 때 사용하는 키워드이다.

형식) throw new 예외 클래스 생성자;

셋째, **throws 키워드**

throws 키워드는 발생된 예외를 자신이 직접 처리하는 것이 아니라 자신을 호출한 곳으로 떠넘기는 역할을 한다.

형식) public void 메서드명(매개변수) throws Exception{.....}

넷째, **finally 키워드**

finally 키워드는 예외와 상관없이 무조건 마지막에 실행한다.

다섯째. **J D K 1.7 이후에 추가된 try-with-resources 문장**

가. 자원은 프로그램이 종료되면 반드시 close() 메서드로 닫혀져야 한다. 하지만 try-with-resources 문을 사용하면 문장의 끝에서 자원들이 자동으로 닫혀지게 한다.

나. 사용형식

```
try(리소스 자료형1 변수명1=초기값; 리소스자료형2 변수명2=초기값2;...){
 문장...
}
```

다. 소스예

```
try(PrintWriter output=new PrintWriter("test.txt")){
   for(String s:list){//향상된 for
   output.println(s.toLowerCase());
 }//for
}
```

이처럼 try다음에 소괄호가 나오면 리소스로 취급한다. 즉 PrintWriter 객체가 try-with-resources 문으로 선언되었기 때문에 try문장이 정상적으로 종료되건 예외가 발생하건 간에 무조건 닫혀진다.즉 finally문에서 close();메서드로 명시적으로 닫지 않아도 자동으로 해당 객체는 닫혀지게 된다. 이 기능을 사용하려면 자원 객체가 java.lang.AutoCloseable 인터페이스를 구현하고 있어야 한다.