Generics

프로그램 작성법

런타임 에러를 줄이는 방식으로 작성해야한다.

문자열을 초기화할 때 null로 초기화하지말고 빈 문자열로 초기화한다.
예를 들어, length()를 호출할 때 null이라면 에러가 발생한다.

배열을 초기화할 때도 위와 같이 null로 초기화하지말고 길이가 0인 배열을 선언하거나 {}로 초기화한다.

Generics?

다양한 타입의 객체들을 다루는 메서드나 컬렉션 클래스에 컴파일 시의 타입 체크를 해주는 기능이다.
런타임에러의 발생을 줄이기위해서 사용한다.

지네릭스의 장점이다.

1. 타입 안정성을 제공한다.
2. 타입 체크와 형변환을 생략할 수 있으므로 코드가 간결해진다.

지네릭스 선언

지네릭스는 클래스와 메서드에 선언할 수 있다.

지네릭스 제한

1. 타입 변수는 인스턴스 변수로 간주되기 때문에 static 멤버에는 타입 변수를 사용할 수 없다.
2. new 연산자를 사용할 수 없기 때문에 지네릭 타입의 배열을 생성할 수 없다.

지네릭스 클래스

클래스를 작성할 때, Object 타입 대신 타입 변수를 사용한다.
클래스 옆에 <타입 변수>를 붙이면 된다.

지네릭스 클래스의 객체를 생성할 때는 참조 변수와 생성자에 타입 변수 대신에 사용될 실제 타입을 지정해주어야 한다.

제한된 지네릭스 클래스

타입 변수에 extends를 사용하면 특정 타입의 자손들만 대입할 수 있게 제한할 수 있다.
이때 인터페이스도 키워드 extends를 사용하고, 클래스의 상속과 인터페이스의 구현을 같이 해야한다면 & 기호로 연결한다.

와일드 카드 ?

하나의 참조 변수로 서로 다른 타입에 대입된 여러 지네릭스 객체를 다루기 위해 사용한다.
단, 와일드 카드에는 & 기호를 사용할 수 없다.

종류	기능
<? extends T>	와일드 카드의 상한 제한이며 T와 그 자손들만 가능하다.
<? super T>	와일드 카드의 하한 제한이며 T와 그 조상들만 가능하다.
<?>	제한이 없으며 모든 타입이 가능하다.(<? extends Object>와 동일하다.)

지네릭스 메서드

메서드의 선언부에 지네릭스 타입이 선언된 메서드이다.
메서드를 호출할 때마다 다른 지네릭스 타입을 대입할 수 있게 한 것이다.
static 메서드는 타입 매개변수를 사용할 수 없지만 지네릭스 메서드로 선언하고 사용하는 것은 가능하다.
왜냐하면, 메서드에 선언된 타입 변수는 지역 변수를 선언한 것과 같기 때문이다.
지네릭스 메서드를 호출할 때 대부분의 경우 타입 변수를 생략할 수 있지만, 에러가 나면 타입 변수에 타입을 대입해야한다.

지네릭스 타입의 형변환

지네릭스 타입과 넌지네릭스 타입간의 형변환은 항상 가능하지만 경고가 발생한다.
대입된 타입이 다른 지네릭스 타입으로 형변환하는 것은 불가능하다.
하지만 와일드 카드가 사용된 지네릭스 타입으로는 형변환이 가능하다.
이 말은 다형성이 적용이 된다는 뜻이다.

지네릭스 타입의 제거

컴파일러는 지네릭스 타입을 이용해서 소스 파일을 체크하고, 필요한 곳에 형변환을 넣어주고 지네릭스 타입을 제거한다.
왜냐하면, 지네릭스가 도입되기 이전의 소스 코드와의 호환성을 유지하기 위해서이다.

지네릭스 타입의 제거되는 순서는 다음과 같다.

1. 지네릭스 타입의 경계를 제거한다.
2. 지네릭스 타입을 제거한 후에 타입이 일치하지 않으면, 형변환을 추가한다.
3. 와일드 카드가 포함된 경우에는, 적절한 타입으로 형변환을 추가한다.