**Use:** Collection with Generics

**Generic type:** Khi muốn tổng quát lên (viết để dùng cho nhiều type) nhưng vẫn ràng buộc sau này khi instantiate (type safe, no casting). Say, generify class/method.

public class Box<T> {

// T stands for "Type"

private T t;

public void set(T t) { this.t = t; }

public T get() { return t; }

}

Box<Integer> intergerBox = new Box<>();

-> chỉ được set/get box for integer

Other example

public interface Pair<K, V> {

public K getKey();

public V getValue();

}

public class OrderedPair<K, V> implements Pair<K, V> {

private K key;

private V value;

public OrderedPair(K key, V value) {

this.key = key;

this.value = value;

}

public K getKey() { return key; }

public V getValue() { return value; }

}

The following statements create two instantiations of the OrderedPair class:

Pair<String, Integer> p1 = new OrderedPair<String, Integer>("Even", 8);

Pair<String, String> p2 = new OrderedPair<String, String>("hello", "world");

**Generic method**:  
ex:

public static < E > void printArray( E[] inputArray )

{

// Display array elements

for ( E element : inputArray ){

System.out.printf( "%s ", element );

}

System.out.println();

}

-> use:

Integer[] intArray = { 1, 2, 3, 4, 5 };

Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };

Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

System.out.println( "Array integerArray contains:" );

printArray( intArray ); // pass an Integer array

System.out.println( "\nArray doubleArray contains:" );

printArray( doubleArray ); // pass a Double array

System.out.println( "\nArray characterArray contains:" );

printArray( charArray ); // pass a Character array

-> method dùng để print nhiều loại type khác nhau nhưng vẫn đảm bảo được type safe.

More restrict: chỉ muốn method thao tác trên một số loại type cụ thể (extends from specific class/interface)

public class MaximumTest

{

// determines the largest of three Comparable objects

public static <T extends Comparable<T>> T maximum(T x, T y, T z)

{

T max = x; // assume x is initially the largest

if ( y.compareTo( max ) > 0 ){

max = y; // y is the largest so far

}

if ( z.compareTo( max ) > 0 ){

max = z; // z is the largest now

}

return max; // returns the largest object

}

public static void main( String args[] )

{

System.out.printf( "Max of %d, %d and %d is %d\n\n",

3, 4, 5, maximum( 3, 4, 5 ) );

System.out.printf( "Maxm of %.1f,%.1f and %.1f is %.1f\n\n",

6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ) );

System.out.printf( "Max of %s, %s and %s is %s\n","pear",

"apple", "orange", maximum( "pear", "apple", "orange" ) );

}

}

This method vẫn generic nhưng chỉ trên các child của Comparable interface