Upper Bounded Wildcards

The Upper Bounded wildcards are the wildcard that relaxes the restriction of the variable type. That is, if we want to relax the restriction on the type of the variable in the method, we can use this type of wildcards.

==========Example Code==============

public class UpperBoundWildcard {

public static void main(String[] args) {

//Upper Bounded Integer List

List < Integer > intList = Arrays.asList(10, 20, 30, 50);

//printing the sum of integer elements in list

System.out.println("Total sum is:" + sum(intList));

//Upper Bounded Double list

List < Double > doubleList = Arrays.asList(13.2, 15.6, 9.7, 22.5,20);

//printing the sum of double elements in list

System.out.print("Total sum is: " + sum(doubleList));

}

private static double sum(List < ?extends Number > myList) {

double sum = 0.0;

for (Number iterator: myList) {

sum = sum + iterator.doubleValue();

}

return sum;

}

}

Output : Total sum is: 110.0  
Total sum is: 81.0

#### Lower Bounded Wildcards

We use the Lower Bounded wildcards to widen the use of the type of variable. For example, if we want to add the list of integers in our method we can use the List<Integer>, but using this we will be bound to use only the list of integers.

======== Example Code ==========

import java.util. \* ;

public class LowerBoundWildcard {

public static void main(String[] args) {

//Lower Bounded Integer List

List < Integer > intList = Arrays.asList(10, 20, 40, 80);

//Passing Integer list object

printOnlyIntegerClassorSuperClass(intList);

//Number list

List < Number > numberList = Arrays.asList(10, 20, 40, 80);

//Passing Integer list object

printOnlyIntegerClassorSuperClass(numberList);

}

public static void printOnlyIntegerClassorSuperClass(List < ?super Integer > list) {

System.out.println(list);

}

}

Output : [10, 20, 40, 80]  
[10, 20, 40, 80]