Java Generics

Wildcards

- Wildcards help in allowing more than one type of class in the Collections
- We come across setting an upperbound and lowerbound for the Types which can be allowed in the collection
- The bounds are identified using a ? Operator which means ‘an unknown type’
- `?` indicates a *wild-card* type parameter, one that can be any type.
  - `List<?> list = new List<?>();  // anything`

- Difference between `List<?>` and `List<Object>`:
  - `?` can become any particular type; `Object` is just one such type.
  - `List<Object>` is restrictive; wouldn't take a `List<String>`

Wildcards

- Method 1 (ArrayList<String> AL):
- Method 2 (ArrayList <?> AL):
- Method 3 (ArrayList <? extends x> AL):
- Method 4 (ArrayList <? super X> AL):

X can be either class or interface

"super" keyword is only allowed to be used on The method level.

Wildcards (First Method)

```
ArrayList<String>  Arr1 = new ArrayList <String>();
Method-1 (Arr1);

public static void Method-1 (ArrayList<String> AL) {
        AL.add("A");     √
        AL.add(null);    √
        AL.add(10)       X

  }
```

Wildcards (Second Method)

```
ArrayList<String>  Arr1 = new ArrayList <String>();
Method-2 (Arr1);
ArrayList<Integer> Arr2 = new ArrayList <Integer>();
Method-2(Arr2);
ArrayList<Student> Arr3 = new ArrayList <Student>();
Method-2(Arr3);

public static void Method-2 (ArrayList<?> AL) {
        AL.add("A");
        AL.add(null);
        AL.add(10)
  }
```

Which one is valid?

Wildcards (Second Method)

```
ArrayList<String>  Arr1 = new ArrayList <String>();
Method-2 (Arr1);
ArrayList<Integer> Arr2 = new ArrayList <Integer>();
Method-2(Arr2);
ArrayList<Student> Arr3 = new ArrayList <Student>();
Method-2(Arr3);

public static void Method-2 (ArrayList<?> AL) {
        AL.add("A");     X
        AL.add(null);    √
        AL.add(10)       X
  }
```

You can’t add anything to this type
Of method except null.

The main advantage of this method is
For read only operations.

Wildcards (Third Method)

```
ArrayList<String>  Arr1 = new ArrayList <String>();
Method-3 (Arr1);
ArrayList<Integer> Arr2 = new ArrayList <Integer>();
Method-3(Arr2);
ArrayList<Student> Arr3 = new ArrayList <Student>();
Method-3(Arr3);

public static void Method-3 (ArrayList<? Extends x> AL) {
        AL.add("A");     X
        AL.add(null);    √
        AL.add(10)       X
  }
```

You can’t add anything to this type
Of method except null.

The main advantage of this method is
For read only operations.

Wildcards (Fourth Method)

```
ArrayList<String>  Arr1 = new ArrayList <String>();
Method-4 (Arr1);
ArrayList<Integer> Arr2 = new ArrayList <Integer>();
Method-4(Arr2);
ArrayList<Student> Arr3 = new ArrayList <Student>();
Method-4(Arr3);

public static void Method-4 (ArrayList<? Super x> AL) {

        AL.add("A");      X
        AL.add(null);     √
        AL.add(10)        X
        AL.add(object);   √

  }
```

You can’t add anything to this type
Of method except null and object.

The main advantage of this method is
For read only operations.

Wildcards Questions

- Which declaration is valid and which one is not?
  - `ArrayList <String>  Arr1  = new ArrayList<String>();`
  - `ArrayList <?>  Arr2  = new ArrayList<String>();`
  - `ArrayList <?>  Arr3  = new ArrayList<Integer>();`
  - `ArrayList <? extends Number>  Arr4 = new ArrayList<Integer>();`
  - `ArrayList <? extends Number>  Arr5 = new ArrayList<String>();`
  - `ArrayList <?>  Arr6 = new ArrayList<?>();`
  - `ArrayList <?>  Arr7 = new ArrayList<? extends Number>();`
  - `ArrayList <? super String>  Arr5 = new ArrayList<Object>();`