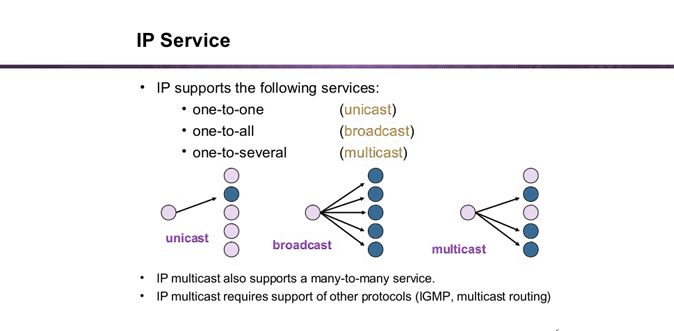
**What are unicast and broadcast?**



**Unicast**: traffic, many streams of IP packets that move across networks flow from a single point, such as a website server, to a single endpoint such as a client PC. This is the most common form of information transference on networks.

**Broadcast:**Here, traffic streams from a single point to all possible endpoints within reach on the network, which is generally a LAN. This is the easiest technique to ensure traffic reaches its destinations.

This mode is mainly utilized by television networks for video and audio distribution. Even if the television network is a cable television (CATV) system, the source signal reaches all possible destinations, which is the key reason that some channels’ content is scrambled. Broadcasting is not practicable on the public Internet due to the massive amount of unnecessary data that would continually reach each user’s device, the complications and impact of scrambling, and related privacy issues.

**What are the differences between unicast, multicast, and broadcast??**

**1. Unicast –**

This type of information transfer is useful when there is a participation of single sender and single recipient. So, in short, you can term it as a one-to-one transmission. For example, a device having IP address 10.1.2.0 in a network wants to send the traffic stream(data packets) to the device with IP address 20.12.4.2 in the other network, then unicast comes into the picture. This is the most common form of data transfer over the networks.

### 2. Broadcast –

Broadcasting transfer (one-to-all) techniques can be classified into two types :

* **Limited Broadcasting –**   
  Suppose you have to send stream of packets to all the devices over the network that you reside, this broadcasting comes handy. For this to achieve, it will append 255.255.255.255 (all the 32 bits of IP address set to 1) called as **Limited Broadcast Address** in the destination address of the datagram (packet) header which is reserved for information transfer to all the recipients from a single client (sender) over the network.

**.Direct Broadcasting –**   
This is useful when a device in one network wants to transfer packet stream to all the devices over the other network. This is achieved by translating all the Host ID part bits of the destination address to 1, referred as **Direct Broadcast Address** in the datagram header for information transfer.

### 3. Multicast –

In multicasting, one/more senders and one/more recipients participate in data transfer traffic. In this method traffic recline between the boundaries of unicast (one-to-one) and broadcast (one-to-all). Multicast lets server’s direct single copies of data streams that are then simulated and routed to hosts that request it. IP multicast requires support of some other protocols like **IGMP (Internet Group Management Protocol), Multicast routing** for its working. Also in Classful IP addressing **Class D** is reserved for multicast groups.

What are java Generics and wildcards?

# Wildcards in Java

The question mark (?) is known as the wildcard in generic programming . It represents an unknown type. The wildcard can be used in a variety of situations such as the type of a parameter, field, or local variable; sometimes as a return type. Unlike arrays, different instantiations of a generic type are not compatible with each other, not even explicitly. This incompatibility may be softened by the wildcard if ? is used as an actual type parameter.

**Types of wildcards in Java:**

Attention reader! Don’t stop learning now. Get hold of all the important [**Java Foundation**](https://practice.geeksforgeeks.org/courses/Java-Foundation?vC=1) and Collections concepts with the [**Fundamentals of Java and Java Collections Course**](https://practice.geeksforgeeks.org/courses/Java-Collections?vC=1) at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more,  please refer [**Complete Interview Preparation Course**](https://practice.geeksforgeeks.org/courses/complete-interview-preparation?utm_source=GeeksforGeeks&utm_medium=Text&utm_campaign=GFG_Article_Bottom_Text_CIP)**.**

1. **Upper Bounded Wildcards:** These wildcards can be used when you want to relax the restrictions on a variable. For example, say you want to write a method that works on List < integer >, List < double >, and List < number > , you can do this  using an upper bounded wildcard.  
   To declare an upper-bounded wildcard, use the wildcard character (‘?’), followed by the extends keyword, followed by its upper bound.

public static void add(List<? extends Number> list)

**Implementation:**

|  |
| --- |
| //Java program to demonstrate Upper Bounded Wildcards  import java.util.Arrays;  import java.util.List;    class WildcardDemo  {      public static void main(String[] args)      {            //Upper Bounded Integer List          List<Integer> list1= Arrays.asList(4,5,6,7);            //printing the sum of elements in list          System.out.println("Total sum is:"+sum(list1));            //Double list          List<Double> list2=Arrays.asList(4.1,5.1,6.1);            //printing the sum of elements in list          System.out.print("Total sum is:"+sum(list2));      }        private static double sum(List<? extends Number> list)      {          double sum=0.0;          for (Number i: list)          {              sum+=i.doubleValue();          }          return sum;      }  } |

**Output:**

Total sum is:22.0

Total sum is:15.299999999999999

**2-Lower Bounded Wildcards:**It is expressed using the wildcard character (‘?’), followed by the super keyword, followed by its lower bound: <? super A>.

**Syntax:** Collectiontype <? super A>

**Implementation:**

|  |
| --- |
| //Java program to demonstrate Lower Bounded Wildcards  import java.util.Arrays;  import java.util.List;    class WildcardDemo  {      public static void main(String[] args)      {          //Lower Bounded Integer List          List<Integer> list1= Arrays.asList(4,5,6,7);            //Integer list object is being passed          printOnlyIntegerClassorSuperClass(list1);            //Number list          List<Number> list2= Arrays.asList(4,5,6,7);            //Integer list object is being passed          printOnlyIntegerClassorSuperClass(list2);      }        public static void printOnlyIntegerClassorSuperClass(List<? super Integer> list)      {          System.out.println(list);      }  } |

**Output:**

[4, 5, 6, 7]

[4, 5, 6, 7]

**3-Unbounded Wildcard:** This wildcard type is specified using the wildcard character (?), for example, List. This is called a list of unknown type. These are useful in the following cases

* When writing a method which can be employed using functionality provided in Object class.
* When the code is using methods in the generic class that don’t depend on the type parameter

**Implementation:**

|  |
| --- |
| //Java program to demonstrate Unbounded wildcard  import java.util.Arrays;  import java.util.List;    class unboundedwildcardemo  {      public static void main(String[] args)      {            //Integer List          List<Integer> list1= Arrays.asList(1,2,3);            //Double list          List<Double> list2=Arrays.asList(1.1,2.2,3.3);            printlist(list1);            printlist(list2);      }        private static void printlist(List<?> list)      {            System.out.println(list);      }  } |

Output:

[1, 2, 3]

[1.1, 2.2, 3.3]

**Array** is a variable that can contain multiple elements with index starting from 0 whereas **enum** is an user defined datatype that contains a list of members for which an integer constant is assigned starting from 0. in case of enum the numbers starting from 0 are not indexes whereas in case of an array they are indexes. Also in case of enum you can assign your own constant values to the members that may or may not start from 0 and may or may not be in a sequence.

An Enum is basically a group of named constants. It is an alternative to numbered flag parameters. (It also doesn't have to be numbered from zero, you can specify the numbering.)

An Enum could be days of the week for example. A Enum could departments in a company: eg SALES, BILLING, HR ...

What is the difference between array list and enums?

A array is a sequence of memory locations. It is a collection. Each element in that collection is indexed by a number. So using that number you can retrieve the value stored at that location. Much like the page number in a book lets you look up the content of that page, the index on an array lets you look up the value stored at that location.

For example, if your company had numbered physical mail boxes for each department (starting from zero): and you were creating some very simple software to let users log in to check how many letters they had uncollected, you might choose to store them in an Array of Ints. Where the index is the mail box number, and the value is the number of letters in the box.

Then for ease of programming, you might choose define the departments as Enums (as described before) such that you could index of department name. But that would be getting more advanced.