DAY01

Summarize the methods of *DatagramSocket* and *DatagramPacket* classes.

Java DatagramSocket Class

Method	Description
void bind(SocketAddress addr)	It binds the DatagramSocket to a specific address and port.
void close()	It closes the datagram socket.
void connect(InetAddress address, int port)	It connects the socket to a remote address for the socket.
void disconnect()	It disconnects the socket.
boolean getBroadcast()	It tests if SO_BROADCAST is enabled.
DatagramChannel getChannel()	It returns the unique DatagramChannel object associated with the datagram socket.
InetAddress getInetAddress()	It returns the address to where the socket is connected.
InetAddress getLocalAddress()	It gets the local address to which the socket is connected.
int getLocalPort()	It returns the port number on the local host to which the socket is bound.
SocketAddress getLocalSocketAddress()	It returns the address of the endpoint the socket is bound to.
int getPort()	It returns the port number to which the socket is connected.
int getReceiverBufferSize()	It gets the value of the SO_RCVBUF option for this DatagramSocket that is the buffer size used by the platform for input on the DatagramSocket.
boolean isClosed()	It returns the status of socket i.e. closed or not.
boolean isConnected()	It returns the connection state of the socket.

void send(DatagramPacket p)	It sends the datagram packet from the socket.
void receive(DatagramPacket p)	It receives the datagram packet from the socket.

Java DatagramPacket Class Methods

Method	Description
1) InetAddress getAddress()	It returns the IP address of the machine to which the datagram is being sent or from which the datagram was received.
2) byte[] getData()	It returns the data buffer.
3) int getLength()	It returns the length of the data to be sent or the length of the data received.
4) int getOffset()	It returns the offset of the data to be sent or the offset of the data received.
5) int getPort()	It returns the port number on the remote host to which the datagram is being sent or from which the datagram was received.
6) SocketAddress getSocketAddress()	It gets the SocketAddress (IP address + port number) of the remote host that the packet is being sent to or is coming from.
7) void setAddress(InetAddress iaddr)	It sets the IP address of the machine to which the datagram is being sent.
8) void setData(byte[] buff)	It sets the data buffer for the packet.
9) void setLength(int length)	It sets the length of the packet.
10) void setPort(int iport)	It sets the port number on the remote host to which the datagram is being sent.

11) VoidsetSocketAddress(SocketAddressaddr)	It sets the SocketAddress (IP address + port number) of the remote host to which the datagram is being sent.

What is *Socket* class

The Socket class represents client sockets, and is a communication channel between two TCP communications ports belonging to one or two machines. A socket may connect to a port on the local system, avoiding the need for a second machine, but most network software will usually involve two machines. TCP sockets can't communicate with more than two machines, however. If this functionality is required, a client application should establish multiple socket connections, one for each machine.

What is *InetAddress* class is used for?

The java.net.InetAddress class provides methods to get the IP address of any hostname. An IP address is represented by 32-bit or 128-bit unsigned number. InetAddress can handle both IPv4 and IPv6 addresses.

DAY02

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

Data is transported over a network by three simple methods i.e. Unicast, Broadcast, and Multicast. So let's begin to summarize the difference between **these three:**

- **Unicast**: from one source to one destination i.e. One-to-One
- **Broadcast**: from one source to all possible destinations i.e. One-to-All
- **Multicast**: from one source to multiple destinations stating an interest in receiving the traffic i.e. One-to-Many

Note: There is no separate classification for Many-to-Many applications, for example, video conferencing or online gaming, where multiple sources for the same receiver and where receivers often are double as sources. This service model works on the basis of one-to-many multicast and for that reason requires no unique protocol. The original multicast design i.e. RFC 1112, supports both the ASM (any-source-multicast) based on a many-to-many service model and the SSM (source-specific multicast) based on a one-to-many model.

What are java Generics and wildcards?

Wildcards in Java are basically the question marks which we use in generic programming, it basically represents the unknown type. We use Java Wildcard widely in situations such as in a type of parameter, local variable, or field and also as a return type.

Unlike arrays, different instantiations of a generic type are not compatible with each other, not even explicitly. We can remove this incompatibility by using wildcard '?' as an actual type parameter.

Wildcards are nothing but the **question mark(?)** that you use in the Java Generics. We can use the Java Wildcard as a local variable, parameter, field or as a return type. But, when the generic class is instantiated or when a generic method is called, we can't use wildcards.
The wildcard is useful to remove the incompatibility between different instantiations of a generic type. This incompatibility is removed by using wildcards ? as an actual type parameter.

What is the difference between array list and enums?

Java Virtual Machine considers enums and arrays as classes.

Enum is a keyword in java and is a type like class or interface and it can be used to define set of enum constants. Enum are collection of named constants, Once you declared Enum constants you cannot change there value . Enum's are type-safe can be used in switch cases.

Eg:

public enum Day { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY }

An array is used to store collection of data of same datatypes like used in other programming languages. You can add,delete,modify array elements.

Eg:

char[] vowels = {'A','E', 'I','O','U'}

int[] numbers = {1,2,3,4,5,6,7,8,9}