java.net

Class DatagramPacket

```
java.lang.Object
  └ java.net.DatagramPacket
```

```
public final class DatagramPacket
extends Object
```

This class represents a datagram packet.

Datagram packets are used to implement a connectionless packet delivery service. Each message is routed from one machine to another based solely on information contained within that packet. Multiple packets sent from one machine to another might be routed differently, and might arrive in any order. Packet delivery is not guaranteed.

Since:
JDK1.0

Constructor Summary

`DatagramPacket(byte[] buf, int length)` Constructs a `DatagramPacket` for receiving packets of length `length`.
`DatagramPacket(byte[] buf, int length, InetAddress address, int port)` Constructs a datagram packet for sending packets of length `length` to the specified port number on the specified host.
`DatagramPacket(byte[] buf, int offset, int length)` Constructs a `DatagramPacket` for receiving packets of length `length`, specifying an offset into the buffer.
`DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)` Constructs a datagram packet for sending packets of length `length` with offset `ioffset`to the specified port number on the specified host.
`DatagramPacket(byte[] buf, int offset, int length, SocketAddress address)` Constructs a datagram packet for sending packets of length `length` with offset `ioffset`to the specified port number on the specified host.
`DatagramPacket(byte[] buf, int length, SocketAddress address)` Constructs a datagram packet for sending packets of length `length` to the specified port number on the specified host.

Method Summary

InetAddress	**getAddress**() Returns the IP address of the machine to which this datagram is being sent or from which the datagram was received.
byte[]	**getData**() Returns the data buffer.
int	**getLength**() Returns the length of the data to be sent or the length of the data received.
int	**getOffset**() Returns the offset of the data to be sent or the offset of the data received.
int	**getPort**() Returns the port number on the remote host to which this datagram is being sent or from which the datagram was received.
SocketAddress	**getSocketAddress**() Gets the SocketAddress (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.
void	**setAddress**(InetAddress iaddr) Sets the IP address of the machine to which this datagram is being sent.
void	**setData**(byte[] buf) Set the data buffer for this packet.
void	**setData**(byte[] buf, int offset, int length) Set the data buffer for this packet.
void	**setLength**(int length) Set the length for this packet.
void	**setPort**(int iport) Sets the port number on the remote host to which this datagram is being sent.
void	**setSocketAddress**(SocketAddress address) Sets the SocketAddress (usually IP address + port number) of the remote host to which this datagram is being sent.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DatagramPacket

```
public DatagramPacket(byte[] buf,
                      int offset,
                      int length)
```

Constructs a `DatagramPacket` for receiving packets of length `length`, specifying an offset into the buffer.

The `length` argument must be less than or equal to `buf.length`.

Parameters:
`buf` - buffer for holding the incoming datagram.
`offset` - the offset for the buffer
`length` - the number of bytes to read.
Since:
JDK1.2

DatagramPacket

```
public DatagramPacket(byte[] buf,
                      int length)
```

Constructs a `DatagramPacket` for receiving packets of length `length`.

The `length` argument must be less than or equal to `buf.length`.

Parameters:
`buf` - buffer for holding the incoming datagram.
`length` - the number of bytes to read.

DatagramPacket

```
public DatagramPacket(byte[] buf,
                      int offset,
                      int length,
                      InetAddress address,
                      int port)
```

Constructs a datagram packet for sending packets of length `length` with offset `ioffset`to the specified port number on the specified host. The `length` argument must be less than or equal to `buf.length`.

Parameters:
`buf` - the packet data.
`offset` - the packet data offset.
`length` - the packet data length.
`address` - the destination address.
`port` - the destination port number.
Since:
JDK1.2
See Also:
`InetAddress`

DatagramPacket

```
public DatagramPacket(byte[] buf,
                      int offset,
                      int length,
                      SocketAddress address)
```

```
        throws SocketException
```

Constructs a datagram packet for sending packets of length `length` with offset `ioffset`to the specified port number on the specified host. The `length` argument must be less than or equal to `buf.length`.

Parameters:

`buf` - the packet data.
`offset` - the packet data offset.
`length` - the packet data length.
`address` - the destination socket address.

Throws:

`IllegalArgumentException` - if address type is not supported
`SocketException`

Since:

1.4

See Also:

`InetAddress`

DatagramPacket

```
public DatagramPacket(byte[] buf,
                      int length,
                      InetAddress address,
                      int port)
```

Constructs a datagram packet for sending packets of length `length` to the specified port number on the specified host. The `length` argument must be less than or equal to `buf.length`.

Parameters:

`buf` - the packet data.
`length` - the packet length.
`address` - the destination address.
`port` - the destination port number.

See Also:

`InetAddress`

DatagramPacket

```
public DatagramPacket(byte[] buf,
                      int length,
                      SocketAddress address)
               throws SocketException
```

Constructs a datagram packet for sending packets of length `length` to the specified port number on the specified host. The `length` argument must be less than or equal to `buf.length`.

Parameters:

`buf` - the packet data.
`length` - the packet length.
`address` - the destination address.

Throws:
`IllegalArgumentException` - if address type is not supported
`SocketException`

Since:
1.4

See Also:
`InetAddress`

Method Detail

getAddress

```
public InetAddress getAddress()
```

Returns the IP address of the machine to which this datagram is being sent or from which the datagram was received.

Returns:
the IP address of the machine to which this datagram is being sent or from which the datagram was received.

See Also:
`InetAddress`, `setAddress(java.net.InetAddress)`

---

getPort

```
public int getPort()
```

Returns the port number on the remote host to which this datagram is being sent or from which the datagram was received.

Returns:
the port number on the remote host to which this datagram is being sent or from which the datagram was received.

See Also:
`setPort(int)`

---

getData

```
public byte[] getData()
```

Returns the data buffer. The data received or the data to be sent starts from the `offset` in the buffer, and runs for `length` long.

Returns:
the buffer used to receive or send data

See Also:
`setData(byte[], int, int)`

---

getOffset

```
public int getOffset()
```

Returns the offset of the data to be sent or the offset of the data received.

Returns:
the offset of the data to be sent or the offset of the data received.
Since:
JDK1.2

---

getLength

```
public int getLength()
```

Returns the length of the data to be sent or the length of the data received.

Returns:
the length of the data to be sent or the length of the data received.
See Also:
`setLength(int)`

---

setData

```
public void setData(byte[] buf,
                    int offset,
                    int length)
```

Set the data buffer for this packet. This sets the data, length and offset of the packet.

Parameters:
`buf` - the buffer to set for this packet
`offset` - the offset into the data
`length` - the length of the data and/or the length of the buffer used to receive data
Throws:
`NullPointerException` - if the argument is null
Since:
JDK1.2
See Also:
`getData()`, `getOffset()`, `getLength()`

---

setAddress

```
public void setAddress(InetAddress iaddr)
```

Sets the IP address of the machine to which this datagram is being sent.

Parameters:

`iaddr` - the `InetAddress`

Since:

JDK1.1

See Also:

`getAddress()`

---

setPort

```
public void setPort(int iport)
```

Sets the port number on the remote host to which this datagram is being sent.

Parameters:

`iport` - the port number

Since:

JDK1.1

See Also:

`getPort()`

---

setSocketAddress

```
public void setSocketAddress(SocketAddress address)
```

Sets the SocketAddress (usually IP address + port number) of the remote host to which this datagram is being sent.

Parameters:

`address` - the `SocketAddress`

Throws:

`IllegalArgumentException` - if address is null or is a SocketAddress subclass not supported by this socket

Since:

1.4

See Also:

`getSocketAddress()`

---

getSocketAddress

```
public SocketAddress getSocketAddress()
```

Gets the SocketAddress (usually IP address + port number) of the remote host that this packet is being sent to or is coming from.

Returns:

the `SocketAddress`

Since:

1.4

See Also:

`setSocketAddress(java.net.SocketAddress)`

---

setData

```
public void setData(byte[] buf)
```

Set the data buffer for this packet. With the offset of this DatagramPacket set to 0, and the length set to the length of `buf`.

Parameters:
`buf` - the buffer to set for this packet.
Throws:
`NullPointerException` - if the argument is null.
Since:
JDK1.1
See Also:
`getLength()`, `getData()`

---

setLength

```
public void setLength(int length)
```

Set the length for this packet. The length of the packet is the number of bytes from the packet's data buffer that will be sent, or the number of bytes of the packet's data buffer that will be used for receiving data. The length must be lesser or equal to the offset plus the length of the packet's buffer.

Parameters:
`length` - the length to set for this packet.
Throws:
`IllegalArgumentException` - if the length is negative of if the length is greater than the packet's data buffer length.
Since:
JDK1.1
See Also:
`getLength()`, `setData(byte[], int, int)`
