## Delay
<<Java Class>>
**Delay**
(default package)

- ▫ delay: int
- ▵ data: byte[]
- ▵ addr: InetAddress
- ▵ port: int
- ▵ socket: DatagramSocket
- ▵ sendPacket: DatagramPacket

- ● Delay(byte[],InetAddress,int,DatagramSocket,int)
- ● run():void
- ● threadName():String
- ● twoBytesToInt(byte,byte):int

## UserInput
<<Java Class>>
**UserInput**
(default package)

- ▫ input: Scanner

- ● UserInput()
- ● run():void

## Quit
<<Java Class>>
**Quit**
(default package)

- ▫ input: Scanner

- ● Quit()
- ● run():void

## Server
<<Java Class>>
**Server**
(default package)

- ▵ receivePacket: DatagramPacket
- ▵ receiveSocket: DatagramSocket
- ▪ MAX_DATA: int

- ● Server()
- ● main(String[]):void
- ● getSocket():DatagramSocket
- ● listener():void
- ● makeConnection(DatagramPacket):void
- ● makeConnection(DatagramPacket,byte[]):void
- ● parseRequest(byte[]):String
- ● getFilename(byte[]):String
- ● getMode(byte[]):String
- ● receive():DatagramPacket
- ● processDatagram(DatagramPacket):byte[]
- ● isValidPacket(DatagramPacket):boolean
- ● parseError(byte[]):String
- ● twoBytesToInt(byte,byte):int
- ● createError(int,String):byte[]

## ClientConnection
<<Java Class>>
**ClientConnection**
(default package)

- ▪ MAX_DATA: int
- ▵ requestPacket: DatagramPacket
- ▵ sendPacket: DatagramPacket
- ▵ receivePacket: DatagramPacket
- ▵ sendReceiveSocket: DatagramSocket
- ▪ TIMEOUT: int
- ▫ blockNumberWrap: boolean
- ▪ fileDirectory: String
- ▵ op: int
- ▵ filename: String
- ▵ addr: InetAddress
- ▵ port: int
- ▵ error: byte[]
- ▫ in: BufferedInputStream

- ● ClientConnection(DatagramPacket)
- ● ClientConnection(DatagramPacket,byte[])
- ● run():void
- ● threadName():String
- ● readReq():void
- ● writeReq():void
- ● closeConnection():void
- ● twoBytesToInt(byte,byte):int
- ● getFilename(DatagramPacket):String
- ● getData(byte[]):byte[]
- ● send(byte[]):void
- ● receive():DatagramPacket
- ● processDatagram(DatagramPacket):byte[]
- ● parseAck(byte[]):String
- ● parseData(byte[]):String
- ● parseError(byte[]):String
- ● createAck(int):byte[]
- ● createData(int,byte[]):byte[]
- ● createError(int,String):byte[]
- ● isValidPacket(DatagramPacket):boolean

## Client
<<Java Class>>
**Client**
(default package)

- ▵ sendPacket: DatagramPacket
- ▵ receivePacket: DatagramPacket
- ▵ sendReceiveSocket: DatagramSocket
- ▪ TIMEOUT: int
- ▫ input: Scanner
- ▵ filename: String
- ▪ fileDirectory: String
- ▵ mode: String
- ▫ in: BufferedInputStream
- ▪ MAX_DATA: int
- ▵ addr: InetAddress
- ▵ port: int
- ▫ req: byte
- ▫ blockNumberWrap: boolean

- ● Client()
- ● main(String[]):void
- ● ui(int,InetAddress):void
- ● connection():void
- ● readReq():void
- ● writeReq():void
- ● createRequest(int,String,String):byte[]
- ● createAck(int):byte[]
- ● createData(int,byte[]):byte[]
- ● createError(int,String):byte[]
- ● parseRequest(byte[]):String
- ● parseAck(byte[]):String
- ● parseData(byte[]):String
- ● parseError(byte[]):String
- ● getData(byte[]):byte[]
- ● send(byte[],InetAddress,int):void
- ● receive():DatagramPacket
- ● processDatagram(DatagramPacket):byte[]
- ● twoBytesToInt(byte,byte):int
- ● isValidPacket(DatagramPacket):boolean

## Opcode
<<Java Enumeration>>
**Opcode**
(default package)

- ▪ RRQ: Opcode
- ▪ WRQ: Opcode
- ▪ DATA: Opcode
- ▪ ACK: Opcode
- ▪ ERROR: Opcode
- ▫ op: byte

- ▵ Opcode(byte)
- ▪ op():byte

## Opcode
<<Java Enumeration>>
**Opcode**
(default package)

- ▪ RRQ: Opcode
- ▪ WRQ: Opcode
- ▪ ACK: Opcode
- ▪ DATA: Opcode
- ▪ ERROR: Opcode

- ● Opcode()