XU_Ziyang_cpp_doc

Generated by Doxygen 1.12.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- Group . . . ??
- InputBuffer . . . ??
- Multimedia . . . ??
  - Film . . . ??
  - Photo . . . ??
  - Video . . . ??
- MultimediaManager . . . ??
- std::runtime_error
  - MultimediaException . . . ??
    - DuplicateNameException . . . ??
    - InvalidDurationException . . . ??
    - InvalidNameException . . . ??
    - NotFoundException . . . ??
- ServerSocket . . . ??
- Socket . . . ??
- SocketBuffer . . . ??
- SocketCnx . . . ??
- TCPServer . . . ??

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

DuplicateNameException . . . . . ??
Film . . . . . ??
Group . . . . . ??
InputBuffer . . . . . ??
InvalidDurationException . . . . . ??
InvalidNameException . . . . . ??
Multimedia . . . . . ??
MultimediaException . . . . . ??
MultimediaManager . . . . . ??
NotFoundException . . . . . ??
Photo . . . . . ??
ServerSocket . . . . . ??
Socket . . . . . ??
SocketBuffer . . . . . ??
SocketCnx
    Connection with a given client. Each SocketCnx uses a different thread . . . . . ??
TCPServer . . . . . ??
Video . . . . . ??

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

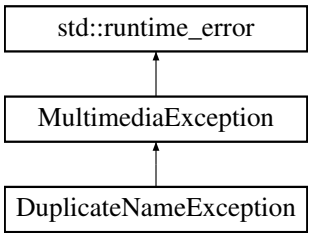
cpp/ccsocket.h . . . ??
cpp/client.cpp
Main client application to interact with the server . . . ??
cpp/Film.cpp
Implementation of the Film class . . . ??
cpp/Film.h . . . ??
cpp/Group.cpp
Implementation of the Group class . . . ??
cpp/Group.h . . . ??
cpp/Multimedia.cpp
Implementation of the Multimedia class . . . ??
cpp/Multimedia.h . . . ??
cpp/MultimediaException.h . . . ??
cpp/MultimediaManager.cpp
Implementation of the MultimediaManager class . . . ??
cpp/MultimediaManager.h . . . ??
cpp/Photo.cpp
Implementation of the Photo class . . . ??
cpp/Photo.h . . . ??
cpp/server.cpp
Main server application that handles multimedia requests . . . ??
cpp/tcpserver.h . . . ??
cpp/Video.cpp
Implementation of the Video class . . . ??
cpp/Video.h . . . ??

Chapter 4

Class Documentation

4.1 DuplicateNameException Class Reference

Inheritance diagram for DuplicateNameException:



Public Member Functions

- **DuplicateNameException** (const std::string &message)

Public Member Functions inherited from MultimediaException

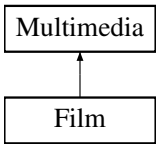
- **MultimediaException** (const std::string &message)

The documentation for this class was generated from the following file:

- cpp/MultimediaException.h

4.2 Film Class Reference

Inheritance diagram for Film:



Public Member Functions

- Film (const std::string &name, const std::string &path, int duration, const std::vector< int > &chapters)

  Constructs a Film object.
- void display (std::ostream &os) const override

  Displays film information.
- void play () const override

  Plays the film.
- void serialize (std::ofstream &ofs) const override

  Serializes the film to a file.
- void deserialize (std::ifstream &ifs) override

  Deserializes the film from a file.
- std::string getClassName () const override

  Gets the class name of the film.

Public Member Functions inherited from Multimedia

- Multimedia (const std::string &name, const std::string &path)

  Constructs a Multimedia object.
- virtual ~**Multimedia** ()

  Destroys the Multimedia object.
- std::string getName () const

  Returns the name of the multimedia.

Additional Inherited Members

Protected Attributes inherited from Multimedia

- std::string **name**
- std::string **path**

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Film()

```
Film::Film (
            const std::string & name,
            const std::string & path,
            int duration,
            const std::vector< int > & chapters)
```

Constructs a Film object.

Parameters

name	The name of the film.
path	The file path of the film.
duration	The duration of the film in seconds.
chapters	The chapters of the film.

4.2.2 Member Function Documentation

4.2.2.1 deserialize()

```
void Film::deserialize (
            std::ifstream & ifs)  [override], [virtual]
```

Deserializes the film from a file.

Parameters

ifs	The input file stream to read the serialized film.

Implements Multimedia.

4.2.2.2 display()

```
void Film::display (
            std::ostream & os) const  [override], [virtual]
```

Displays film information.

Parameters

os	The output stream to display the information.

Reimplemented from Multimedia.

4.2.2.3 getClassName()

```
std::string Film::getClassName () const  [override], [virtual]
```

Gets the class name of the film.

Returns

The class name.

Implements Multimedia.

4.2.2.4 play()

```
void Film::play () const  [override], [virtual]
```

Plays the film.

Implements Multimedia.

4.2.2.5 serialize()

```
void Film::serialize (
            std::ofstream & ofs) const  [override], [virtual]
```

Serializes the film to a file.

Parameters

ofs	The output file stream to write the serialized film.

Implements Multimedia.

The documentation for this class was generated from the following files:

- cpp/Film.h
- cpp/Film.cpp

4.3 Group Class Reference

Public Member Functions

- Group (const std::string &name)
  Constructs a Group object.
- ~**Group** ()
  Destroys a Group object.
- void push_back (const std::shared_ptr< Multimedia > &item)
  Adds an item to the group.
- void remove (const std::shared_ptr< Multimedia > &item)
  Removes an item from the group.
- void display (std::ostream &os) const
  Displays group information.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Group()

```
Group::Group (
            const std::string & name)
```

Constructs a Group object.

Parameters

name	The name of the group.

4.3.2 Member Function Documentation

4.3.2.1 display()

```
void Group::display (
            std::ostream & os) const
```

Displays group information.

Parameters

os	The output stream to display the information.

4.3.2.2 push_back()

```
void Group::push_back (
            const std::shared_ptr< Multimedia > & item)
```

Adds an item to the group.

Parameters

item	The item to add.

4.3.2.3 remove()

```
void Group::remove (
            const std::shared_ptr< Multimedia > & item)
```

Removes an item from the group.

Parameters

item	The item to remove.

The documentation for this class was generated from the following files:

- cpp/Group.h
- cpp/Group.cpp

4.4 InputBuffer Struct Reference

Public Member Functions

- **InputBuffer** (size_t size)

Public Attributes

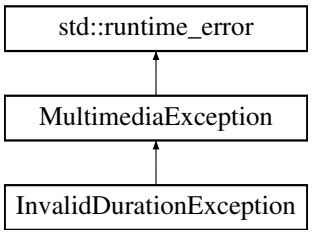
- char * **buffer**
- char * **begin**
- char * **end**
- SOCKSIZE **remaining**

The documentation for this struct was generated from the following file:

- cpp/ccsocket.cpp

4.5 InvalidDurationException Class Reference

Inheritance diagram for InvalidDurationException:



Public Member Functions

- **InvalidDurationException** (const std::string &message)

Public Member Functions inherited from MultimediaException

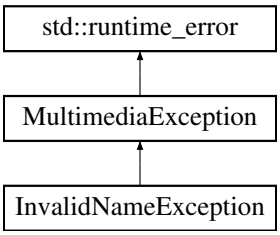
- **MultimediaException** (const std::string &message)

The documentation for this class was generated from the following file:

- cpp/MultimediaException.h

4.6 InvalidNameException Class Reference

Inheritance diagram for InvalidNameException:



Public Member Functions

- **InvalidNameException** (const std::string &message)

Public Member Functions inherited from MultimediaException

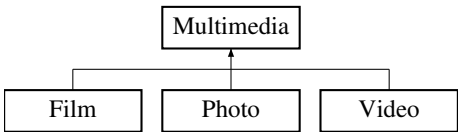
- **MultimediaException** (const std::string &message)

The documentation for this class was generated from the following file:

- cpp/MultimediaException.h

4.7 Multimedia Class Reference

Inheritance diagram for Multimedia:



Public Member Functions

- Multimedia (const std::string &name, const std::string &path)

  Constructs a Multimedia object.
- virtual ~**Multimedia** ()

  Destroys the Multimedia object.
- virtual void display (std::ostream &os) const

  Displays multimedia information.
- virtual void play () const =0
- std::string getName () const

  Returns the name of the multimedia.
- virtual void serialize (std::ofstream &ofs) const =0
- virtual void deserialize (std::ifstream &ifs)=0
- virtual std::string getClassName () const =0

Protected Attributes

- std::string **name**
- std::string **path**

Friends

- std::ofstream & operator<< (std::ofstream &ofs, const Multimedia &multimedia)

  Serializes the multimedia to a file.
- std::ifstream & operator>> (std::ifstream &ifs, Multimedia &multimedia)

  Deserializes the multimedia from a file.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Multimedia()

```
Multimedia::Multimedia (
            const std::string & name,
            const std::string & path)
```

Constructs a Multimedia object.

Parameters

name	The name of the multimedia.
path	The file path of the multimedia.

4.7.2 Member Function Documentation

4.7.2.1 deserialize()

```
virtual void Multimedia::deserialize (
            std::ifstream & ifs)  [pure virtual]
```

Implemented in Film, Photo, and Video.

4.7.2.2 display()

```
void Multimedia::display (
            std::ostream & os) const  [virtual]
```

Displays multimedia information.

Parameters

os	The output stream to display the information.

Reimplemented in Film, Photo, and Video.

4.7.2.3 getClassName()

```
virtual std::string Multimedia::getClassName () const  [pure virtual]
```

Implemented in Film, Photo, and Video.

4.7.2.4 getName()

```
std::string Multimedia::getName () const
```

Returns the name of the multimedia.

Returns

The name of the multimedia.

4.7.2.5 play()

```
virtual void Multimedia::play () const  [pure virtual]
```

Implemented in Film, Photo, and Video.

4.7.2.6 serialize()

```
virtual void Multimedia::serialize (
            std::ofstream & ofs) const  [pure virtual]
```

Implemented in Film, Photo, and Video.

4.7.3 Friends And Related Symbol Documentation

4.7.3.1 operator<<

```
std::ofstream & operator<< (
            std::ofstream & ofs,
            const Multimedia & multimedia)  [friend]
```

Serializes the multimedia to a file.

Parameters

ofs	The output file stream to write the serialized multimedia.

4.7.3.2 operator>>

```
std::ifstream & operator>> (
            std::ifstream & ifs,
            Multimedia & multimedia)  [friend]
```

Deserializes the multimedia from a file.

Parameters

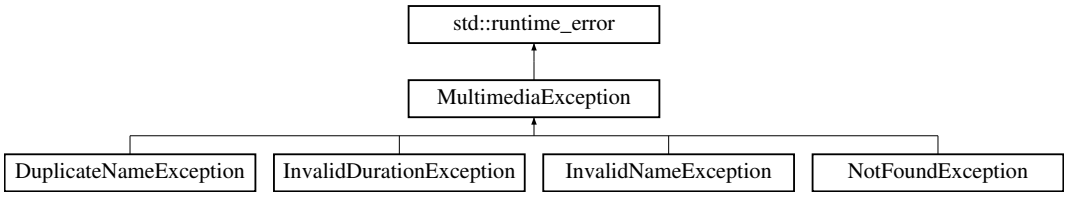
ifs	The input file stream to read the serialized multimedia.

The documentation for this class was generated from the following files:

- cpp/Multimedia.h
- cpp/Multimedia.cpp

4.8 MultimediaException Class Reference

Inheritance diagram for MultimediaException:



Public Member Functions

- **MultimediaException** (const std::string &message)

The documentation for this class was generated from the following file:

- cpp/MultimediaException.h

4.9 MultimediaManager Class Reference

Public Member Functions

- void createPhoto (const std::string &name, const std::string &path, double latitude, double longitude)

  Creates a photo and adds it to the manager.
- void createVideo (const std::string &name, const std::string &path, int duration)

  Creates a video and adds it to the manager.
- void createFilm (const std::string &name, const std::string &path, int duration, const std::vector< int > &chapters)

  Creates a film and adds it to the manager.
- void createGroup (const std::string &name)

  Creates a group and adds it to the manager.
- void displayMultimedia (const std::string &name) const

  Displays multimedia information.
- void displayGroup (const std::string &name) const

  Displays group information.
- void playMultimedia (const std::string &name) const

  Plays multimedia.
- void save (const std::string &filename) const

  Saves multimedia objects to a file.
- void load (const std::string &filename)

  Loads multimedia objects from a file.

4.9.1 Member Function Documentation

4.9.1.1 createFilm()

```
void MultimediaManager::createFilm (
            const std::string & name,
            const std::string & path,
            int duration,
            const std::vector< int > & chapters)
```

Creates a film and adds it to the manager.

Parameters

name	The name of the film.
path	The file path of the film.
duration	The duration of the film in seconds.
chapters	The chapters of the film.

4.9.1.2 createGroup()

```
void MultimediaManager::createGroup (
            const std::string & name)
```

Creates a group and adds it to the manager.

Parameters

name	The name of the group.

4.9.1.3 createPhoto()

```
void MultimediaManager::createPhoto (
            const std::string & name,
            const std::string & path,
            double latitude,
            double longitude)
```

Creates a photo and adds it to the manager.

Parameters

name	The name of the photo.
path	The file path of the photo.
latitude	The latitude where the photo was taken.
longitude	The longitude where the photo was taken.

4.9.1.4 createVideo()

```
void MultimediaManager::createVideo (
            const std::string & name,
            const std::string & path,
            int duration)
```

Creates a video and adds it to the manager.

Parameters

name	The name of the video.
path	The file path of the video.
duration	The duration of the video in seconds.

4.9.1.5 displayGroup()

```
void MultimediaManager::displayGroup (
            const std::string & name) const
```

Displays group information.

Parameters

name	The name of the group.

4.9.1.6 displayMultimedia()

```
void MultimediaManager::displayMultimedia (
            const std::string & name) const
```

Displays multimedia information.

Parameters

name	The name of the multimedia.

4.9.1.7 load()

```
void MultimediaManager::load (
            const std::string & filename)
```

Loads multimedia objects from a file.

Parameters

filename	The name of the file to load from.

4.9.1.8 playMultimedia()

```
void MultimediaManager::playMultimedia (
            const std::string & name) const
```

Plays multimedia.

Parameters

name	The name of the multimedia.

4.9.1.9 save()

```
void MultimediaManager::save (
            const std::string & filename) const
```

Saves multimedia objects to a file.

Parameters

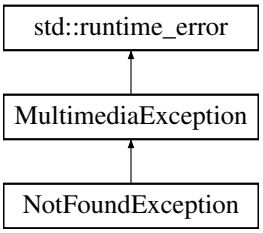
filename	The name of the file to save to.

The documentation for this class was generated from the following files:

- cpp/MultimediaManager.h
- cpp/MultimediaManager.cpp

4.10 NotFoundException Class Reference

Inheritance diagram for NotFoundException:



Public Member Functions

- **NotFoundException** (const std::string &message)

Public Member Functions inherited from MultimediaException

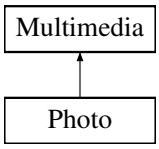
- **MultimediaException** (const std::string &message)

The documentation for this class was generated from the following file:

- cpp/MultimediaException.h

4.11 Photo Class Reference

Inheritance diagram for Photo:



Public Member Functions

- Photo (const std::string &name, const std::string &path, double latitude, double longitude)

  Constructs a Photo object.
- void display (std::ostream &os) const override

  Displays photo information.
- void play () const override

  Plays the photo.
- void serialize (std::ofstream &ofs) const override

  Serializes the photo to a file.
- void deserialize (std::ifstream &ifs) override

  Deserializes the photo from a file.
- std::string getClassName () const override

  Returns the class name.

Public Member Functions inherited from Multimedia

- Multimedia (const std::string &name, const std::string &path)

  Constructs a Multimedia object.
- virtual ~**Multimedia** ()

  Destroys the Multimedia object.
- std::string getName () const

  Returns the name of the multimedia.

Additional Inherited Members

Protected Attributes inherited from Multimedia

- std::string **name**
- std::string **path**

4.11.1 Constructor & Destructor Documentation

4.11.1.1 Photo()

```
Photo::Photo (
            const std::string & name,
            const std::string & path,
            double latitude,
            double longitude)
```

Constructs a Photo object.

Parameters

name	The name of the photo.
path	The file path of the photo.
latitude	The latitude where the photo was taken.
longitude	The longitude where the photo was taken.

4.11.2 Member Function Documentation

4.11.2.1 deserialize()

```
void Photo::deserialize (
            std::ifstream & ifs)  [override], [virtual]
```

Deserializes the photo from a file.

Parameters

ifs	The input file stream to read the serialized photo.

Implements Multimedia.

4.11.2.2 display()

```
void Photo::display (
            std::ostream & os) const  [override], [virtual]
```

Displays photo information.

Parameters

os	The output stream to display the information.

Reimplemented from Multimedia.

4.11.2.3 getClassName()

```
std::string Photo::getClassName () const  [override], [virtual]
```

Returns the class name.

Returns

The class name.

Implements Multimedia.

4.11.2.4 play()

```
void Photo::play () const  [override], [virtual]
```

Plays the photo.

Implements Multimedia.

4.11.2.5 serialize()

```
void Photo::serialize (
            std::ofstream & ofs) const  [override], [virtual]
```

Serializes the photo to a file.

Parameters

ofs	The output file stream to write the serialized photo.

Implements Multimedia.

The documentation for this class was generated from the following files:

- cpp/Photo.h
- cpp/Photo.cpp

4.12 ServerSocket Class Reference

`#include <ccsocket.h>`

Public Member Functions

- **ServerSocket** ()

  Creates a listening socket that waits for connection requests by TCP/IP clients.
- Socket * accept ()
- int bind (int port, int backlog=50)
- int **close** ()

  Closes the socket.
- bool **isClosed** () const

  Returns true if the socket was closed.
- SOCKET **descriptor** ()

  Returns the descriptor of the socket.
- int **setReceiveBufferSize** (int size)

  Sets the SO_RCVBUF option to the specified value.
- int **setReuseAddress** (bool)

  Enables/disables the SO_REUSEADDR socket option.
- int **setSoTimeout** (int timeout)

  Enables/disables SO_TIMEOUT with the specified timeout (in milliseconds).
- int **setTcpNoDelay** (bool)

  Turns on/off TCP coalescence (useful in some cases to avoid delays).

4.12.1 Detailed Description

TCP/IP IPv4 server socket. Waits for requests to come in over the network. TCP/IP sockets do not preserve record boundaries but SocketBuffer solves this problem.

4.12.2 Member Function Documentation

4.12.2.1 accept()

```
Socket * ServerSocket::accept ()
```

Accepts a new connection request and returns a socket for exchanging data with this client. This function blocks until there is a connection request.

Returns

the new Socket or nullptr on error.

4.12.2.2 bind()

```
int ServerSocket::bind (
            int port,
            int backlog = 50)
```

Assigns the server socket to localhost.

Returns

0 on success or a negative value on error, see Socket::Errors

The documentation for this class was generated from the following files:

- cpp/ccsocket.h
- cpp/ccsocket.cpp

4.13 Socket Class Reference

```
#include <ccsocket.h>
```

Public Types

- enum Errors { **Failed** = -1 , **InvalidSocket** = -2 , **UnknownHost** = -3 }

Public Member Functions

- Socket (int type=SOCK_STREAM)
- **Socket** (int type, SOCKET sockfd)

  Creates a Socket from an existing socket file descriptor.
- ~**Socket** ()

  Destructor (closes the socket).
- int connect (const std::string &host, int port)
- int bind (int port)
- int bind (const std::string &host, int port)
- int **close** ()

  Closes the socket.
- bool **isClosed** () const

  Returns true if the socket has been closed.
- SOCKET **descriptor** ()

  Returns the descriptor of the socket.
- void **shutdownInput** ()

  Disables further receive operations.
- void **shutdownOutput** ()

  Disables further send operations.
- SOCKSIZE send (const SOCKDATA *buf, size_t len, int flags=0)
- SOCKSIZE receive (SOCKDATA *buf, size_t len, int flags=0)
- SOCKSIZE **sendTo** (void const *buf, size_t len, int flags, SOCKADDR const *to, socklen_t addrlen)

  Sends data to a datagram socket.
- SOCKSIZE **receiveFrom** (void *buf, size_t len, int flags, SOCKADDR *from, socklen_t *addrlen)

  Receives data from datagram socket.
- int **setReceiveBufferSize** (int size)

  Set the size of the TCP/IP input buffer.
- int **setReuseAddress** (bool)

  Enable/disable the SO_REUSEADDR socket option.
- int **setSendBufferSize** (int size)

  Set the size of the TCP/IP output buffer.
- int **setSoLinger** (bool, int linger)

  Enable/disable SO_LINGER with the specified linger time in seconds.
- int **setSoTimeout** (int timeout)

  Enable/disable SO_TIMEOUT with the specified timeout (in milliseconds).
- int **setTcpNoDelay** (bool)

  Enable/disable TCP_NODELAY (turns on/off TCP coalescence).
- int **getReceiveBufferSize** () const

  Return the size of the TCP/IP input buffer.
- bool **getReuseAddress** () const

  Return SO_REUSEADDR state.
- int **getSendBufferSize** () const

  Return the size of the TCP/IP output buffer.
- bool **getSoLinger** (int &linger) const

  Return SO_LINGER state and the specified linger time in seconds.
- int **getSoTimeout** () const

  Return SO_TIMEOUT value.
- bool **getTcpNoDelay** () const

  Return TCP_NODELAY state.

Static Public Member Functions

- static void startup ()
- static void **cleanup** ()

Friends

- class **ServerSocket**

4.13.1 Detailed Description

TCP/IP or UDP/Datagram IPv4 socket. AF_INET connections following the IPv4 Internet protocol are supported.

Note

- ServerSocket should be used on the server side.
- SIGPIPE signals are ignored when using Linux, BSD or MACOSX.
- TCP/IP sockets do not preserve record boundaries but SocketBuffer solves this problem.

4.13.2 Member Enumeration Documentation

4.13.2.1 Errors

```
enum Socket::Errors
```

Socket errors.

- Socket::Failed (-1): could not connect, could not bind, etc.
- Socket::InvalidSocket (-2): invalid socket or wrong socket type
- Socket::UnknownHost (-3): could not reach host

4.13.3 Constructor & Destructor Documentation

4.13.3.1 Socket()

```
Socket::Socket (
            int type = SOCK_STREAM)
```

Creates a new Socket. Creates a AF_INET socket using the IPv4 Internet protocol. Type can be:

- SOCK_STREAM (the default) for TCP/IP connected stream sockets
- SOCK_DGRAM for UDP/datagram sockets (available only or Unix/Linux)

4.13.4 Member Function Documentation

4.13.4.1 bind() [1/2]

```
int Socket::bind (
            const std::string & host,
            int port)
```

Assigns the socket to an IP address. On Unix/Linux host can be a hostname, on Windows it can only be an IP address.

Returns

0 on success or a negative value on error, see Socket::Errors

4.13.4.2 bind() [2/2]

```
int Socket::bind (
            int port)
```

Assigns the socket to localhost.

Returns

0 on success or a negative value on error, see Socket::Errors

4.13.4.3 connect()

```
int Socket::connect (
            const std::string & host,
            int port)
```

Connects the socket to an address. Typically used for connecting TCP/IP clients to a ServerSocket. On Unix/Linux host can be a hostname, on Windows it can only be an IP address.

Returns

0 on success or a negative value on error which is one of Socket::Errors

4.13.4.4 receive()

```
SOCKSIZE Socket::receive (
            SOCKDATA * buf,
            size_t len,
            int flags = 0)  [inline]
```

Receives data from a connected (TCP/IP) socket. Reads at most *len* bytes fand stores them in *buf*. By default, this function blocks the caller until thre is availbale data.

Returns

the number of bytes that were received, or 0 or shutdownOutput() was called on the other side, or Socket::↩ Failed (-1) if an error occured.

4.13.4.5 send()

```
SOCKSIZE Socket::send (
            const SOCKDATA * buf,
            size_t len,
            int flags = 0)  [inline]
```

Send sdata to a connected (TCP/IP) socket. Sends the first *len* bytes in *buf*.

Returns

the number of bytes that were sent, or 0 or shutdownInput() was called on the other side, or Socket::Failed (-1) if an error occured.

Note

TCP/IP sockets do not preserve record boundaries, see SocketBuffer.

4.13.4.6 startup()

```
void Socket::startup ()  [static]
```

initialisation and cleanup of sockets on Widows.

Note

startup is automaticcaly called when a Socket or a ServerSocket is created

The documentation for this class was generated from the following files:

- cpp/ccsocket.h
- cpp/ccsocket.cpp

4.14 SocketBuffer Class Reference

```
#include <ccsocket.h>
```

Public Member Functions

- SOCKSIZE readLine (std::string &message)
- SOCKSIZE writeLine (const std::string &message)
- SOCKSIZE read (char *buffer, size_t len)
- SOCKSIZE write (const char *str, size_t len)
- Socket * **socket** ()

  Returns the associated socket.

- SocketBuffer (Socket *, size_t inputSize=8192, size_t ouputSize=8192)
- **SocketBuffer** (Socket &, size_t inputSize=8192, size_t ouputSize=8192)

- size_t **insize_** {}
- size_t **outsize_** {}
- int **insep_** {}
- int **outsep_** {}
- Socket * **sock_** {}
- struct InputBuffer * **in_** {}
- void setReadSeparator (int separ)
- int **readSeparator** () const
- void setWriteSeparator (int separ)
- int **writeSeparator** () const
- bool **retrieveLine** (std::string &str, SOCKSIZE received)

4.14.1 Detailed Description

Preserves record boundaries when exchanging messages between connected TCP/IP sockets. Ensures that one call to readLine() corresponds to one and exactly one call to writeLine() on the other side. By default, writeLine() adds
at the end of each message and readLine() searches for
, \r or
\r so that it can retreive the entire record. Beware messages should thus not contain these charecters.

```
int main() {
   Socket sock;
   SocketBuffer sockbuf(sock);

   int status = sock.connect("localhost", 3331);
   if (status < 0) {
     cerr « "Could not connect" « endl;
     return 1;
   }

   while (cin) {
     string request, response;
     cout « "Request: ";
     getline(cin, request);

     if (sockbuf.writeLine(request) < 0) {
        cerr « "Could not send message" « endl;
        return 2;
     }
     if (sockbuf.readLine(response) < 0) {
        cerr « "Couldn't receive message" « endl;
        return 3;
     }
   }
  return 0;
}
```

4.14.2 Constructor & Destructor Documentation

4.14.2.1 SocketBuffer()

```
SocketBuffer::SocketBuffer (
             Socket * sock,
             size_t inputSize = 8192,
             size_t ouputSize = 8192)
```

Constructor. *socket* must be a connected TCP/IP Socket. It should **not** be deleted as long as the SocketBuffer is used. *inputSize* and *ouputSize* are the sizes of the buffers that are used internally for exchanging data.

4.14.3 Member Function Documentation

4.14.3.1 read()

```
SOCKSIZE SocketBuffer::read (
             char * buffer,
             size_t len)
```

Reads exactly *len* bytes from the socket, blocks otherwise.

Returns

see readLine()

4.14.3.2 readLine()

```
SOCKSIZE SocketBuffer::readLine (
            std::string & message)
```

Read a message from a connected socket. readLine() receives one (and only one) message sent by writeLine() on the other side, ie, a call to writeLine() corresponds to one and exactly one call to readLine() on the other side. The received data is stored in *message*. This method blocks until the message is fully received.

Returns

The number of bytes that were received or one of the following values:

- 0: shutdownOutput() was called on the other side
- Socket::Failed (-1): a connection error occured
- Socket::InvalidSocket (-2): the socket is invalid.

Note

the separator (eg
) is counted in the value returned by readLine().

4.14.3.3 setReadSeparator()

```
void SocketBuffer::setReadSeparator (
            int separ)
```

Returns/changes the separator used by readLine(). setReadSeparator() changes the symbol used by readLine() to separate successive messages:

- if *separ* $<$ 0 (the default) readLine() searches for \n, \r or \n\r.
- if *separ* $>=$ 0, readLine() searches for this character to separate messages,

4.14.3.4 setWriteSeparator()

```
void SocketBuffer::setWriteSeparator (
            int separ)
```

Returns/changes the separator used by writeLine(). setWriteSeparator() changes the character(s) used by writeLine() to separate successive messages:

- if *separ* $<$ 0 (the default) writeLine() inserts \n\r between successive lines.
- if *separ* $>=$ 0, writeLine() inserts *separ* between successive lines,

4.14.3.5 write()

```
SOCKSIZE SocketBuffer::write (
            const char * str,
            size_t len)
```

Writes *len* bytes to the socket.

Returns

see readLine()

4.14.3.6 writeLine()

```
SOCKSIZE SocketBuffer::writeLine (
            const std::string & message)
```

Send a message to a connected socket. writeLine() sends a message that will be received by a single call of readLine() on the other side,

Returns

see readLine()

Note

if *message* contains one or several occurences of the separator, readLine() will be called as many times on the other side.

The documentation for this class was generated from the following files:

- cpp/ccsocket.h
- cpp/ccsocket.cpp

4.15 SocketCnx Class Reference

Connection with a given client. Each SocketCnx uses a different thread.

Public Member Functions

- **SocketCnx** (TCPServer &, Socket *)
- void **processRequests** ()

Public Attributes

- TCPServer & **server_**
- Socket * **sock_**
- SocketBuffer * **sockbuf_**
- std::thread **thread_**

4.15.1 Detailed Description

Connection with a given client. Each SocketCnx uses a different thread.

The documentation for this class was generated from the following file:

- cpp/tcpserver.cpp

4.16 TCPServer Class Reference

```
#include <tcpserver.h>
```

Public Types

- using Callback

Public Member Functions

- TCPServer (Callback const &callback)
- virtual int run (int port)

Friends

- class **TCPLock**
- class **SocketCnx**

4.16.1 Detailed Description

TCP/IP IPv4 server. Supports TCP/IP AF_INET IPv4 connections with multiple clients. One thread is used per client.

4.16.2 Member Typedef Documentation

4.16.2.1 Callback

```
using TCPServer::Callback
```

Initial value:

```
  std::function< bool(std::string const& request, std::string& response) >
```

4.16.3 Constructor & Destructor Documentation

4.16.3.1 TCPServer()

```
TCPServer::TCPServer (
            Callback const & callback)
```

initializes the server. The callback function will be called each time the server receives a request from a client.

- *request* contains the data sent by the client
- *response* will be sent to the client as a response The connection with the client is closed if the callback returns false.

4.16.4 Member Function Documentation

4.16.4.1 run()

```
int TCPServer::run (
            int port)  [virtual]
```

Starts the server. Binds an internal ServerSocket to *port* then starts an infinite loop that processes connection requests from clients.

Returns

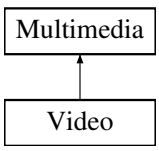
0 on normal termination, or a negative value if the ServerSocket could not be bound (value is then one of Socket::Errors).

The documentation for this class was generated from the following files:

- cpp/tcpserver.h
- cpp/tcpserver.cpp

4.17 Video Class Reference

Inheritance diagram for Video:



Public Member Functions

- Video (const std::string &name, const std::string &path, int duration)
  Constructs a Video object.
- void display (std::ostream &os) const override
  Displays video information.
- void play () const override
  Plays the video.
- void serialize (std::ofstream &ofs) const override
  Serializes the video to a file.
- void deserialize (std::ifstream &ifs) override
  Deserializes the video from a file.
- std::string getClassName () const override
  Returns the class name.

Public Member Functions inherited from Multimedia

- Multimedia (const std::string &name, const std::string &path)
  Constructs a Multimedia object.
- virtual ~**Multimedia** ()
  Destroys the Multimedia object.
- std::string getName () const
  Returns the name of the multimedia.

Additional Inherited Members

Protected Attributes inherited from Multimedia

- std::string **name**
- std::string **path**

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Video()

```
Video::Video (
            const std::string & name,
            const std::string & path,
            int duration)
```

Constructs a Video object.

Parameters

name	The name of the video.
path	The file path of the video.
duration	The duration of the video in seconds.

4.17.2 Member Function Documentation

4.17.2.1 deserialize()

```
void Video::deserialize (
            std::ifstream & ifs)  [override], [virtual]
```

Deserializes the video from a file.

Parameters

ifs	The input file stream to read the serialized video.

Implements Multimedia.

4.17.2.2 display()

```
void Video::display (
            std::ostream & os) const  [override], [virtual]
```

Displays video information.

Parameters

os	The output stream to display the information.

Reimplemented from Multimedia.

4.17.2.3 getClassName()

```
std::string Video::getClassName () const  [override], [virtual]
```

Returns the class name.

Returns

The class name.

Implements Multimedia.

4.17.2.4 play()

```
void Video::play () const  [override], [virtual]
```

Plays the video.

Implements Multimedia.

4.17.2.5 serialize()

```
void Video::serialize (
            std::ofstream & ofs) const  [override], [virtual]
```

Serializes the video to a file.

Parameters

ofs	The output file stream to write the serialized video.

Implements Multimedia.

The documentation for this class was generated from the following files:

- cpp/Video.h
- cpp/Video.cpp

Chapter 5

File Documentation

5.1 ccsocket.h

```
00001 //
00002 //  ccsocket: C++ Classes for TCP/IP and UDP Datagram INET Sockets.
00003 //  (c) Eric Lecolinet 2016/2020 - https://www.telecom-paris.fr/~elc
00004 //
00005 //  - Socket: TCP/IP or UDP/Datagram IPv4 socket
00006 //  - ServerSocket: TCP/IP Socket Server
00007 //  - SocketBuffer: preserves record boundaries when exchanging data
00008 //   between TCP/IP sockets.
00009 //
00010
00011 #ifndef ccuty_ccsocket
00012 #define ccuty_ccsocket 1
00013
00014 #include <string>
00015
00016 #if defined(_WIN32) || defined(_WIN64)
00017 #include <winsock2.h>
00018 #define SOCKSIZE int
00019 #define SOCKDATA char
00020
00021 #else
00022 #include <sys/types.h>
00023 #include <sys/socket.h>
00024 #define SOCKET int
00025 #define SOCKADDR struct sockaddr
00026 #define SOCKADDR_IN struct sockaddr_in
00027 #define INVALID_SOCKET -1
00028 #define SOCKSIZE ssize_t
00029 #define SOCKDATA void
00030 #endif
00031
00032 // ignore SIGPIPES when possible
00033 #if defined(MSG_NOSIGNAL)
00034 #  define NO_SIGPIPE_(flags) (flags | MSG_NOSIGNAL)
00035 #else
00036 #  define NO_SIGPIPE_(flags) (flags)
00037 #endif
00038
00046 class Socket {
00047 public:
00052   enum Errors { Failed = -1, InvalidSocket = -2, UnknownHost = -3 };
00053
00057   static void startup();
00058   static void cleanup();
00060
00065   Socket(int type = SOCK_STREAM);
00066
00068   Socket(int type, SOCKET sockfd);
00069
00071   ~Socket();
00072
00077   int connect(const std::string& host, int port);
00078
00081   int bind(int port);
00082
00086   int bind(const std::string& host, int port);
00087
00089   int close();
```

```
00090
00092   bool isClosed() const { return sockfd_ == INVALID_SOCKET; }
00093
00095   SOCKET descriptor() { return sockfd_; }
00096
00098   void shutdownInput();
00099
00101   void shutdownOutput();
00102
00108   SOCKSIZE send(const SOCKDATA* buf, size_t len, int flags = 0) {
00109     return ::send(sockfd_, buf, len, NO_SIGPIPE_(flags));
00110   }
00111
00117   SOCKSIZE receive(SOCKDATA* buf, size_t len, int flags = 0) {
00118     return ::recv(sockfd_, buf, len, flags);
00119   }
00120
00121 #if !defined(_WIN32) && !defined(_WIN64)
00122
00124   SOCKSIZE sendTo(void const* buf, size_t len, int flags,
00125                   SOCKADDR const* to, socklen_t addrlen) {
00126     return ::sendto(sockfd_, buf, len, NO_SIGPIPE_(flags), to, addrlen);
00127   }
00128
00130   SOCKSIZE receiveFrom(void* buf, size_t len, int flags,
00131                        SOCKADDR* from, socklen_t* addrlen) {
00132     return ::recvfrom(sockfd_, buf, len, flags, from, addrlen);
00133   }
00134
00136   int setReceiveBufferSize(int size);
00137
00139   int setReuseAddress(bool);
00140
00142   int setSendBufferSize(int size);
00143
00145   int setSoLinger(bool, int linger);
00146
00148   int setSoTimeout(int timeout);
00149
00151   int setTcpNoDelay(bool);
00152
00154   int getReceiveBufferSize() const;
00155
00157   bool getReuseAddress() const;
00158
00160   int getSendBufferSize() const;
00161
00163   bool getSoLinger(int& linger) const;
00164
00166   int getSoTimeout() const;
00167
00169   bool getTcpNoDelay() const;
00170
00171 #endif
00172
00173 private:
00174   friend class ServerSocket;
00175
00176   // Initializes a local INET4 address, returns 0 on success, -1 otherwise.
00177   int setLocalAddress(SOCKADDR_IN& addr, int port);
00178   // Initializes a remote INET4 address, returns 0 on success, -1 otherwise.
00179   int setAddress(SOCKADDR_IN& addr, const std::string& host, int port);
00180
00181   SOCKET sockfd_{};
00182   Socket(const Socket&) = delete;
00183   Socket& operator=(const Socket&) = delete;
00184   Socket& operator=(Socket&&) = delete;
00185 };
00186
00187
00188
00192 class ServerSocket {
00193 public:
00195   ServerSocket();
00196
00197   ~ServerSocket();
00198
00202   Socket* accept();
00203
00206   int bind(int port, int backlog = 50);
00207
00209   int close();
00210
00212   bool isClosed() const { return sockfd_ == INVALID_SOCKET; }
00213
00215   SOCKET descriptor() { return sockfd_; }
00216
```

```
00217 #if !defined(_WIN32) && !defined(_WIN64)
00218
00220   int setReceiveBufferSize(int size);
00221
00223   int setReuseAddress(bool);
00224
00226   int  setSoTimeout(int timeout);
00227
00229   int setTcpNoDelay(bool);
00230
00231 #endif
00232
00233 private:
00234   Socket* createSocket(SOCKET);
00235   SOCKET sockfd_{};  // listening socket.
00236   ServerSocket(const ServerSocket&) = delete;
00237   ServerSocket& operator=(const ServerSocket&) = delete;
00238   ServerSocket& operator=(ServerSocket&&) = delete;
00239 };
00240
00241
00276 class SocketBuffer {
00277 public:
00283   SocketBuffer(Socket*, size_t inputSize = 8192, size_t ouputSize = 8192);
00284   SocketBuffer(Socket&, size_t inputSize = 8192, size_t ouputSize = 8192);
00286
00287   ~SocketBuffer();
00288
00300   SOCKSIZE readLine(std::string& message);
00301
00309   SOCKSIZE writeLine(const std::string& message);
00310
00313   SOCKSIZE read(char* buffer, size_t len);
00314
00317   SOCKSIZE write(const char* str, size_t len);
00318
00320   Socket* socket() { return sock_; }
00321
00327   void setReadSeparator(int separ);
00328   int readSeparator() const { return insep_; }
00329   // @}
00330
00336   void setWriteSeparator(int separ);
00337   int writeSeparator() const { return outsep_; }
00338   // @}
00339
00340 private:
00341   SocketBuffer(const SocketBuffer&) = delete;
00342   SocketBuffer& operator=(const SocketBuffer&) = delete;
00343   SocketBuffer& operator=(SocketBuffer&&) = delete;
00344
00345 protected:
00346   bool retrieveLine(std::string& str, SOCKSIZE received);
00347   size_t insize_{}, outsize_{};
00348   int insep_{}, outsep_{};
00349   Socket* sock_{};
00350   struct InputBuffer* in_{};
00351 };
00352
00353 #endif
```

5.2 cpp/client.cpp File Reference

Main client application to interact with the server.

```
#include <iostream>
#include <string>
#include <algorithm>
#include "ccsocket.h"
```

Functions

- int main ()

  Main function to run the client.

5.2.1 Detailed Description

Main client application to interact with the server.

5.2.2 Function Documentation

5.2.2.1 main()

```
int main ()
```

Main function to run the client.

Returns

Exit status.

5.3 cpp/Film.cpp File Reference

Implementation of the Film class.

```
#include "Film.h"
```

5.3.1 Detailed Description

Implementation of the Film class.

5.4 Film.h

```
00001 #ifndef FILM_H
00002 #define FILM_H
00003
00004 #include "Multimedia.h"
00005 #include <vector>
00006
00007 class Film : public Multimedia {
00008 public:
00009     Film(const std::string& name, const std::string& path, int duration, const std::vector<int>&
      chapters);
00010     void display(std::ostream& os) const override;
00011     void play() const override;
00012     void serialize(std::ofstream& ofs) const override;
00013     void deserialize(std::ifstream& ifs) override;
00014     std::string getClassName() const override;
00015
00016 private:
00017     int duration;
00018     std::vector<int> chapters;
00019 };
00020
00021 #endif // FILM_H
```

5.5 cpp/Group.cpp File Reference

Implementation of the Group class.

```
#include "Group.h"
```

5.5.1 Detailed Description

Implementation of the Group class.

5.6 Group.h

```
00001 #ifndef GROUP_H
00002 #define GROUP_H
00003
00004 #include <vector>
00005 #include <memory>
00006 #include <iostream>
00007 #include "Multimedia.h"
00008
00009 class Group {
00010 private:
00011     std::string name;
00012     std::vector<std::shared_ptr<Multimedia» items;
00013
00014 public:
00015     Group(const std::string &name);
00016     ~Group();
00017
00018     void push_back(const std::shared_ptr<Multimedia> &item);
00019     void remove(const std::shared_ptr<Multimedia> &item);
00020     void display(std::ostream &os) const;
00021 };
00022
00023 using GroupPtr = std::shared_ptr<Group>;
00024
00025 #endif // GROUP_H
```

5.7 cpp/Multimedia.cpp File Reference

Implementation of the Multimedia class.

```
#include "Multimedia.h"
```

Functions

- std::ofstream & operator<< (std::ofstream &ofs, const Multimedia &multimedia)

  Serializes the multimedia to a file.
- std::ifstream & operator>> (std::ifstream &ifs, Multimedia &multimedia)

  Deserializes the multimedia from a file.

5.7.1 Detailed Description

Implementation of the Multimedia class.

5.7.2 Function Documentation

5.7.2.1 operator<<()

```
std::ofstream & operator<< (
            std::ofstream & ofs,
            const Multimedia & multimedia)
```

Serializes the multimedia to a file.

Parameters

ofs	The output file stream to write the serialized multimedia.

5.7.2.2 operator>>()

```
std::ifstream & operator>> (
            std::ifstream & ifs,
            Multimedia & multimedia)
```

Deserializes the multimedia from a file.

Parameters

ifs	The input file stream to read the serialized multimedia.

5.8 Multimedia.h

```
00001 #ifndef MULTIMEDIA_H
00002 #define MULTIMEDIA_H
00003
00004 #include <string>
00005 #include <iostream>
00006 #include <fstream>
00007
00008 class Multimedia {
00009 public:
00010     Multimedia(const std::string& name, const std::string& path);
00011     virtual ~Multimedia();
00012     virtual void display(std::ostream& os) const;
00013     virtual void play() const = 0;
00014     std::string getName() const;
00015
00016     virtual void serialize(std::ofstream& ofs) const = 0;
00017     virtual void deserialize(std::ifstream& ifs) = 0;
00018     virtual std::string getClassName() const = 0;
00019
00020     friend std::ofstream& operator«(std::ofstream& ofs, const Multimedia& multimedia);
00021     friend std::ifstream& operator»(std::ifstream& ifs, Multimedia& multimedia);
00022
00023 protected:
00024     std::string name;
00025     std::string path;
00026 };
00027
00028 #endif // MULTIMEDIA_H
```

5.9 MultimediaException.h

```
00001 // File: MultimediaException.h
00002 #ifndef MULTIMEDIAEXCEPTION_H
00003 #define MULTIMEDIAEXCEPTION_H
00004
00005 #include <stdexcept>
00006 #include <string>
00007
00008 class MultimediaException : public std::runtime_error {
00009 public:
00010     explicit MultimediaException(const std::string& message) : std::runtime_error(message) {}
00011 };
00012
00013 class DuplicateNameException : public MultimediaException {
00014 public:
00015     explicit DuplicateNameException(const std::string& message) : MultimediaException(message) {}
00016 };
00017
00018 class NotFoundException : public MultimediaException {
00019 public:
00020     explicit NotFoundException(const std::string& message) : MultimediaException(message) {}
00021 };
00022
00023 class InvalidNameException : public MultimediaException {
00024 public:
00025     explicit InvalidNameException(const std::string& message) : MultimediaException(message) {}
00026 };
00027
00028 class InvalidDurationException : public MultimediaException {
00029 public:
00030     explicit InvalidDurationException(const std::string& message) : MultimediaException(message) {}
00031 };
00032
00033 #endif // MULTIMEDIAEXCEPTION_H
```

5.10 cpp/MultimediaManager.cpp File Reference

Implementation of the MultimediaManager class.

```
#include "MultimediaManager.h"
#include "Photo.h"
#include "Video.h"
#include "Film.h"
#include "Group.h"
#include "MultimediaException.h"
#include <fstream>
#include <iostream>
#include <regex>
```

5.10.1 Detailed Description

Implementation of the MultimediaManager class.

5.11 MultimediaManager.h

```
00001 // File: MultimediaManager.h
00002 #ifndef MULTIMEDIAMANAGER_H
00003 #define MULTIMEDIAMANAGER_H
00004
00005 #include <string>
00006 #include <vector>
00007 #include <memory>
00008 #include <map>
00009 #include "Multimedia.h"
```

```
00010 #include "Group.h"
00011 #include "MultimediaException.h"
00012
00013 class MultimediaManager {
00014 public:
00015     void createPhoto(const std::string& name, const std::string& path, double latitude, double
      longitude);
00016     void createVideo(const std::string& name, const std::string& path, int duration);
00017     void createFilm(const std::string& name, const std::string& path, int duration, const
      std::vector<int>& chapters);
00018     void createGroup(const std::string& name);
00019     void displayMultimedia(const std::string& name) const;
00020     void displayGroup(const std::string& name) const;
00021     void playMultimedia(const std::string& name) const;
00022     void save(const std::string& filename) const;
00023     void load(const std::string& filename);
00024
00025 private:
00026     std::map<std::string, std::shared_ptr<Multimedia» multimediaMap;
00027     std::map<std::string, std::shared_ptr<Group» groupMap;
00028     bool isValidName(const std::string& name) const;
00029 };
00030
00031 #endif // MULTIMEDIAMANAGER_H
```

5.12 cpp/Photo.cpp File Reference

Implementation of the Photo class.

```
#include "Photo.h"
```

5.12.1 Detailed Description

Implementation of the Photo class.

5.13 Photo.h

```
00001 #ifndef PHOTO_H
00002 #define PHOTO_H
00003
00004 #include "Multimedia.h"
00005
00006 class Photo : public Multimedia {
00007 public:
00008     Photo(const std::string& name, const std::string& path, double latitude, double longitude);
00009     void display(std::ostream& os) const override;
00010     void play() const override;
00011     void serialize(std::ofstream& ofs) const override;
00012     void deserialize(std::ifstream& ifs) override;
00013     std::string getClassName() const override;
00014
00015 private:
00016     double latitude;
00017     double longitude;
00018 };
00019
00020 #endif // PHOTO_H
```

5.14 cpp/server.cpp File Reference

Main server application that handles multimedia requests.

```
#include <iostream>
#include <sstream>
#include <string>
#include <memory>
#include "MultimediaManager.h"
#include "ccsocket.h"
#include "tcpserver.h"
```

Macros

- #define **PORT** 3333

Functions

- void processRequest (const std::string &request, MultimediaManager &manager, std::string &response)

  Processes a request from the client.
- int main ()

  Main function to run the server.

5.14.1 Detailed Description

Main server application that handles multimedia requests.

5.14.2 Function Documentation

5.14.2.1 main()

```
int main ()
```

Main function to run the server.

Returns

Exit status.

5.14.2.2 processRequest()

```
void processRequest (
            const std::string & request,
            MultimediaManager & manager,
            std::string & response)
```

Processes a request from the client.

Parameters

request	The request string from the client.
manager	The multimedia manager to handle the request.
response	The response string to be sent back to the client.

5.15 tcpserver.h

```
00001 //
00002 //  tcpserver: TCP/IP INET Server.
00003 //  (c) Eric Lecolinet - Telecom ParisTech - 2016.
00004 //  http://www.telecom-paristech.fr/~elc
00005 //
00006
00007 #ifndef __tcpserver__
00008 #define __tcpserver__
00009 #include <memory>
00010 #include <string>
00011 #include <functional>
00012 #include "ccsocket.h"
00013
00014 class TCPConnection;
00015 class TCPLock;
00016
00019 class TCPServer {
00020 public:
00021
00022   using Callback =
00023   std::function< bool(std::string const& request, std::string& response) >;
00024
00030   TCPServer(Callback const& callback);
00031
00032   virtual ~TCPServer();
00033
00039   virtual int run(int port);
00040
00041 private:
00042   friend class TCPLock;
00043   friend class SocketCnx;
00044
00045   TCPServer(TCPServer const&) = delete;
00046   TCPServer& operator=(TCPServer const&) = delete;
00047   void error(std::string const& msg);
00048
00049   ServerSocket servsock_;
00050   Callback callback_{};
00051 };
00052
00053 #endif
```

5.16 cpp/Video.cpp File Reference

Implementation of the Video class.

```
#include "Video.h"
```

5.16.1 Detailed Description

Implementation of the Video class.

5.17 Video.h

```
00001 #ifndef VIDEO_H
00002 #define VIDEO_H
00003
00004 #include "Multimedia.h"
00005
00006 class Video : public Multimedia {
00007 public:
00008     Video(const std::string& name, const std::string& path, int duration);
00009     void display(std::ostream& os) const override;
00010     void play() const override;
00011     void serialize(std::ofstream& ofs) const override;
00012     void deserialize(std::ifstream& ifs) override;
00013     std::string getClassName() const override;
00014
00015 private:
00016     int duration;
00017 };
00018
00019 #endif // VIDEO_H
```