Christopher Rivas

<https://github.com/Rivas1/CS-380-Exercise-2>

---------------------------------------------------------------------------------------------------------------------

import java.io.\*;

import java.net.\*;

import java.util.zip.CRC32;

public final class Ex2Client

{

public static void main (String[] args) throws IOException

{

String serverOutput = "";

String[] firstHalf = new String[100]; // first 2 bytes of concatenation

String[] secondHalf = new String[100]; // second 2 bytes of concatenation

String[] byteConcantenation = new String[100]; // array that stores merged two halves to form one byte

int insertLine = 0;

CRC32 errorCode = new CRC32();

String hexErrorCode = "";

String serverResponse = ""; // Server's response after CRC32 is sentt

try

{

Socket socket = new Socket ("codebank.xyz", 38102);

if (socket.isConnected())

System.out.println("Connected to server.");

InputStream IS = socket.getInputStream();

System.out.println("Received bytes:");

for (int i = 0; i < 100; i++)

{

firstHalf[i] = Integer.toString( IS.read() );

secondHalf[i] = Integer.toString( IS.read() );

byteConcantenation[i] = dec\_to\_hex( i, firstHalf[i], secondHalf[i] ); // concatenates both halves then returns it as a hex value

errorCode.update( hex\_to\_dec( byteConcantenation[i] ) );

if ( insertLine == 9)

{

System.out.println();

insertLine = 0;

}

else

{

System.out.print(byteConcantenation[i]);

insertLine++;

}

}

// Generate CRC32 Error Code

System.out.println("Generated CRC32: " + errorCode.getValue() + ".");

hexErrorCode = convert\_errorCode\_to\_Hex(errorCode.getValue() );

// send CRC32 code to server

PrintStream PS = new PrintStream(socket.getOutputStream()); // out from client to server

PS.println(hexErrorCode);

InputStreamReader IR = new InputStreamReader(socket.getInputStream()); // listen to server

BufferedReader BR = new BufferedReader(IR);

// Compare CRC32 codes

if (BR.readLine().equals(hexErrorCode))

System.out.println("Response good.");

else

System.out.println("Response bad.");

// Close the connection

socket.close();

System.out.println("Disconnected from server.");

}

catch (IOException e)

{

e.printStackTrace();

}

}

public static String dec\_to\_hex ( int i, String firstHalf, String secondHalf )

{

String hexFirst = "";

String hexSecond = "";

String b = "";

if ( firstHalf.equals("0"))

{ hexFirst = "0"; }

if ( firstHalf.equals("1"))

{ hexFirst = "1"; }

if ( firstHalf.equals("2"))

{ hexFirst = "2"; }

if ( firstHalf.equals("3"))

{ hexFirst = "3"; }

if ( firstHalf.equals("4"))

{ hexFirst = "4"; }

if ( firstHalf.equals("5"))

{ hexFirst = "5"; }

if ( firstHalf.equals("6"))

{ hexFirst = "6"; }

if ( firstHalf.equals("7"))

{ hexFirst = "7"; }

if ( firstHalf.equals("8"))

{ hexFirst = "8"; }

if ( firstHalf.equals("9"))

{ hexFirst = "9"; }

if ( firstHalf.equals("10"))

{ hexFirst = "A"; }

if ( firstHalf.equals("11"))

{ hexFirst = "B"; }

if ( firstHalf.equals("12"))

{ hexFirst = "C"; }

if ( firstHalf.equals("13"))

{ hexFirst = "D"; }

if ( firstHalf.equals("14"))

{ hexFirst = "E"; }

if ( firstHalf.equals("15"))

{ hexFirst = "F"; }

if ( secondHalf.equals("0"))

{ hexSecond = "0"; }

if ( secondHalf.equals("1"))

{ hexSecond = "1"; }

if ( secondHalf.equals("2"))

{ hexSecond = "2"; }

if ( secondHalf.equals("3"))

{ hexSecond = "3"; }

if ( secondHalf.equals("4"))

{ hexSecond = "4"; }

if ( secondHalf.equals("5"))

{ hexSecond = "5"; }

if ( secondHalf.equals("6"))

{ hexSecond = "6"; }

if ( secondHalf.equals("7"))

{ hexSecond = "7"; }

if ( secondHalf.equals("8"))

{ hexSecond = "8"; }

if ( secondHalf.equals("9"))

{ hexSecond = "9"; }

if ( secondHalf.equals("10"))

{ hexSecond = "A"; }

if ( secondHalf.equals("11"))

{ hexSecond = "B"; }

if ( secondHalf.equals("12"))

{ hexSecond = "C"; }

if ( secondHalf.equals("13"))

{ hexSecond = "D"; }

if ( secondHalf.equals("14"))

{ hexSecond = "E"; }

if ( secondHalf.equals("15"))

{ hexSecond = "F"; }

b = hexFirst + hexSecond;

return b;

}

public static int hex\_to\_dec( String fullByte )

{

int i = -1;

int firstIndex = -1;

int secondIndex = -1;

firstIndex = fullByte.charAt( 0 );

secondIndex = fullByte.charAt( 1 ) ;

if ( firstIndex == 'A' )

firstIndex = 10;

else if ( firstIndex == 'B' )

firstIndex = 11;

else if ( firstIndex == 'C' )

firstIndex = 12;

else if ( firstIndex == 'D' )

firstIndex = 13;

else if ( firstIndex == 'E' )

firstIndex = 14;

else if ( firstIndex == 'F' )

firstIndex = 15;

else

{

firstIndex = Character.getNumericValue( fullByte.charAt( 0 ) );

}

if ( secondIndex == 'A' )

secondIndex = 10;

else if ( secondIndex == 'B' )

secondIndex = 11;

else if ( secondIndex == 'C' )

secondIndex = 12;

else if ( secondIndex == 'D' )

secondIndex = 13;

else if ( secondIndex == 'E' )

secondIndex = 14;

else if ( secondIndex == 'F' )

secondIndex = 15;

else

{

secondIndex = Character.getNumericValue( fullByte.charAt( 1 ) );

}

i = (16 \* firstIndex) + ( secondIndex );

return i;

}

public static String convert\_errorCode\_to\_Hex ( long errorCode )

{

String code = Long.toString(errorCode);

String hexMessageCode = Integer.toHexString(Integer.parseInt(code));

return hexMessageCode;

}

}