**Digital Communications**

**Spring Semester Work 2022**

Yfantidis Dimitris (AEM: 3938)

Part A:

The task was implemented in Java (*v. 17.0.2*). The requested program consists of five classes:

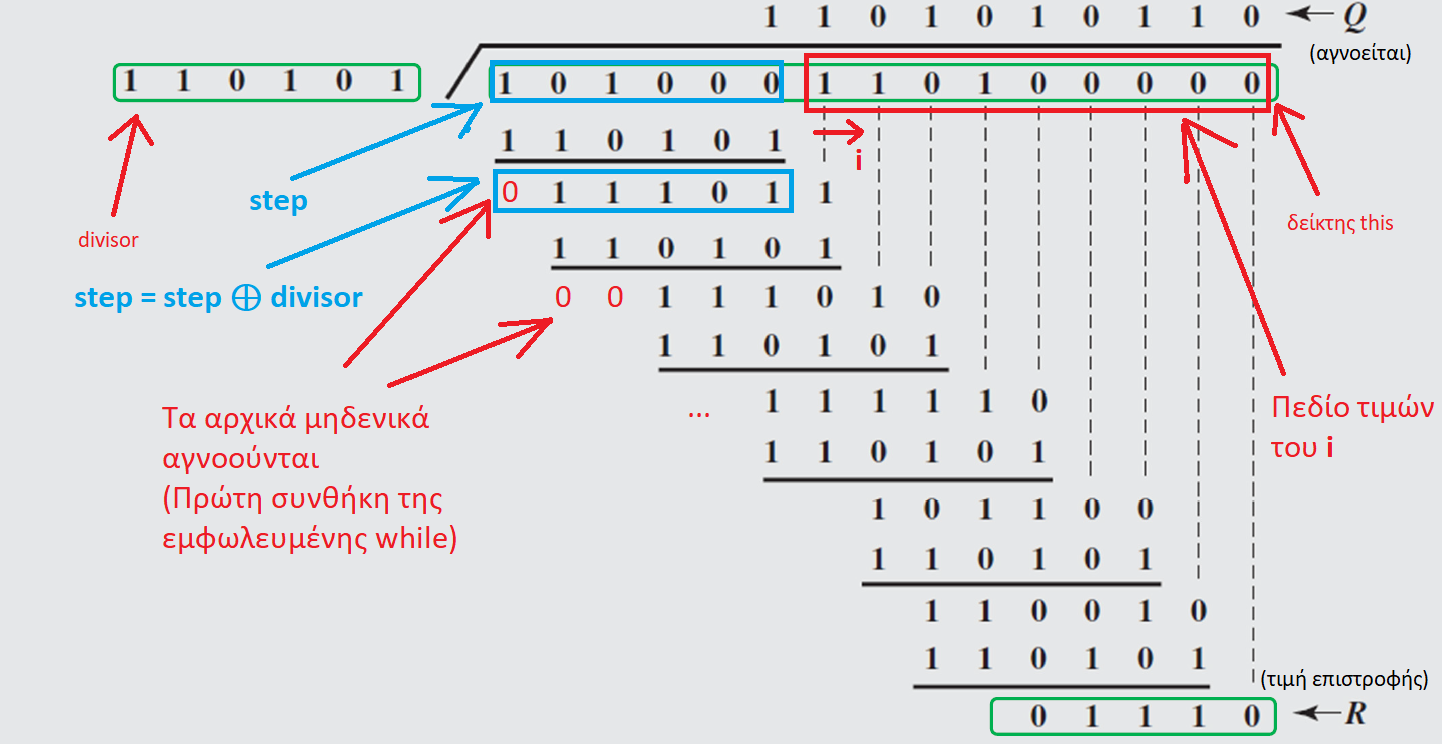
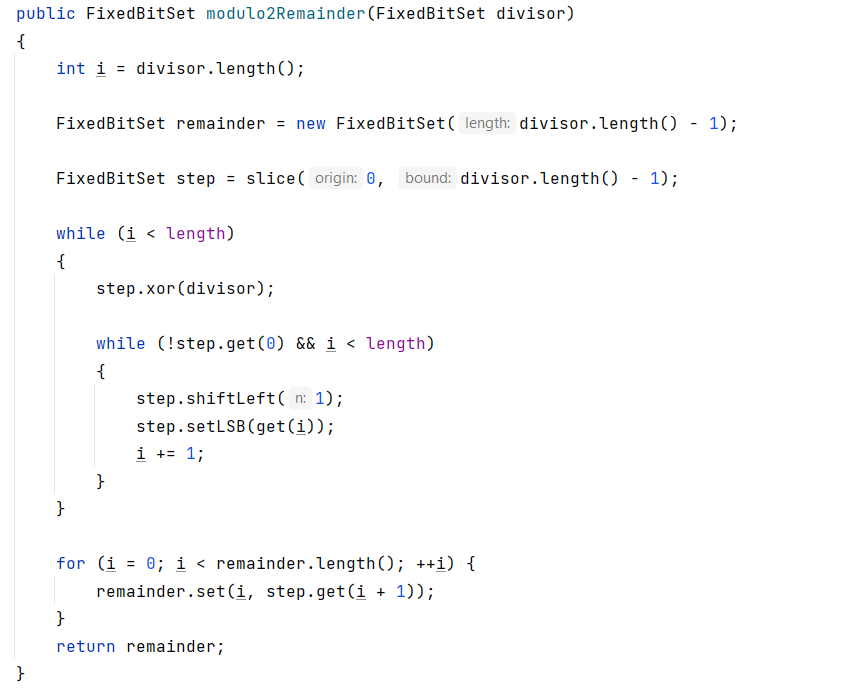
* **FixedBitSet:** The instances of the class represent binary strings. All binary sequences of the task are objects of this class (data blocks, error checking sequences (FCS), predefined divisor ...)
* **SignalTransmitter:** Signal transmitters. Receive a block of data, calculate the error control sequence and send the total sequence to the signal receivers.
* **SignalReceiver:** signal receivers. They receive the total sequence transmitted by a homologous transmitter through a transmission channel. After receiving the signal, they check the validity of the message by means of the cyclic error detection code.
* **SignalChannel:** a channel, of predefined bit error rate, through which messages are transmitted from a SignalTransmitter to a SignalReceiver.
* **Main:** The class of the main program.

More specifically:

**FixedBitSet:**

The class inherits from the java.util.BitSet. It is provided with the **length()** method that returns the length of the string, the **slice(a, b)** method that returns the substring from bit(a) to bit(b). It consists of other methods such as **setLSB(bool)**, **shiftLeft(n)**, toString**()**, **equals(obj), parseBitSet(String).**

More importantly, **modulo2Remainder(FixedBitSet div)** returns the remainder of modulo-2 division of the current binary sequence by the sequence div.



**Explanation:** If the divisor has **d** digits, then we assign the first d bits of the sequence of **this to the variable** step and the value d **to** the variable **i**. Then we make step = step XOR divisor and get the sequence **011101**. We discard the initial zero by left dragging the digits of step once thus:

step = **111010**

Then we add the i-th digit of **this to** the sequence and increase **i** by 1:

step = **111011**

The above procedure is the one-way iteration that continues until the initial digit of the step is one, i.e. the initial zeros of the step are discarded and it is combined with the corresponding number of digits of **this** to have a length equal to the divisor.

Thus the nested iteration terminates and the process step = step XOR divisor is repeated.

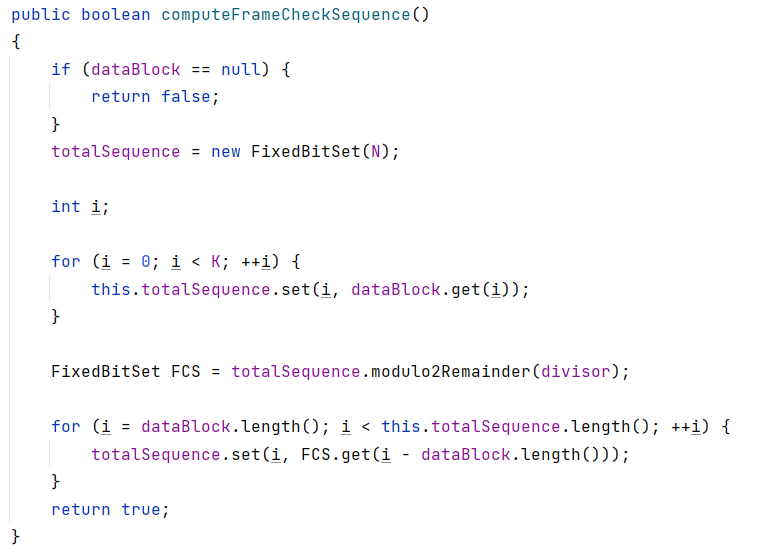
When **i** reaches the end of **this** then the last d-1 of the step is returned as the remainder of the modulo-2 division of **this by** the **divisor**.

**SignalTransmitter:**

An object of this class is a binary string transmitter. The constructor accepts as a parameter a binary string that is the default number for calculating error checking sequences. The properties of the class are:

* The default number (P): **divisor of** length N-K+1 bits
* The data block (D): **dataBlock** of length K bits
* The total sequence (2N-K D + FCS): **totalSequence** of length N bits

First after building a transmitter we define the data block to be sent. Then we call the following function:



**Explanation:** We create the total sequence (T) of N bits and copy the digits of D to the first K digits of T, thus:

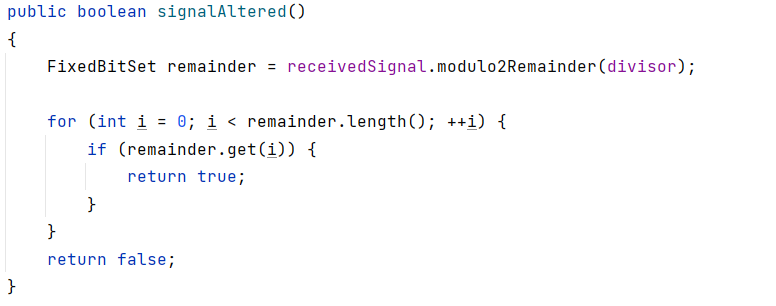
T = 2 DN-K

Then we calculate the remainder of the modulo-2 division of T by the predefined number which is the FCS. Finally we copy the digits of the FCS to the last N-K bits of T, so practically we have:

T = 2N-K D + FCS

**SignalReceiver:**

An object of this class is a binary string receiver. The manufacturer, like transmitters, accepts as a parameter a binary string which is the default number for checking the validity of the received signal. This is done by calling the function:



Returns **true** if there is an ace in the remainder, i.e. if the signal received is not exactly divided by the predefined number. This is how it is detected that the signal was corrupted along the way.

The transmission of a signal from a transmitter to a receiver is done through a transmission channel.

**SignalChannel:**

A SignalChannel is a channel for transmitting binary sequences. The manufacturer accepts the bit error rate as a parameter. The signal is transmitted from a transmitter to a receiver using the

**int SignalChannel.transmitSignal(SignalTransmitter, SignalReceiver)**

Transmission is only performed if the total sequence to be transmitted by the transmitter has been calculated and if the transmitter has the same predefined number as the receiver. First, a copy of the sequence to be transmitted is created. Then, with the help of a repetition structure, each bit of the sequence has a 100 \* BER % chance of changing value. Finally, the receiver receives the possibly corrupted binary string.

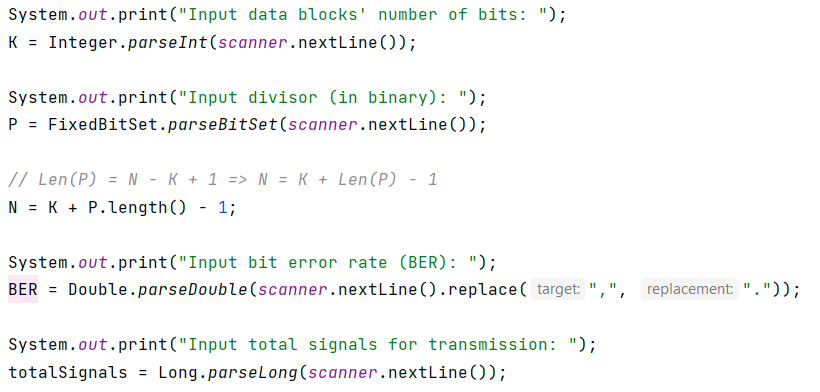
The function returns the number of bit flips that occurred due to noise.



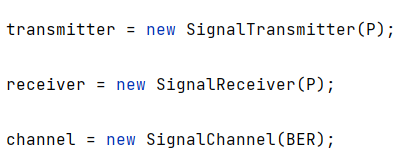
*For the* ***key*** *variable there are sufficient comments in the code.*

**Main:**

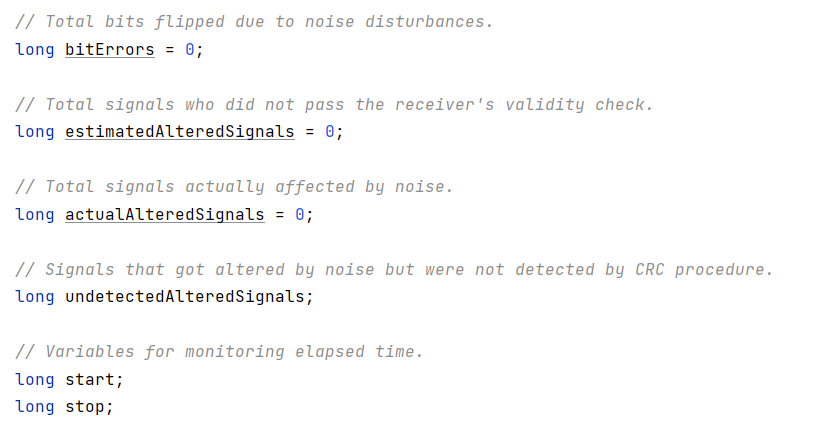
Initially, the program accepts the following values from the user:



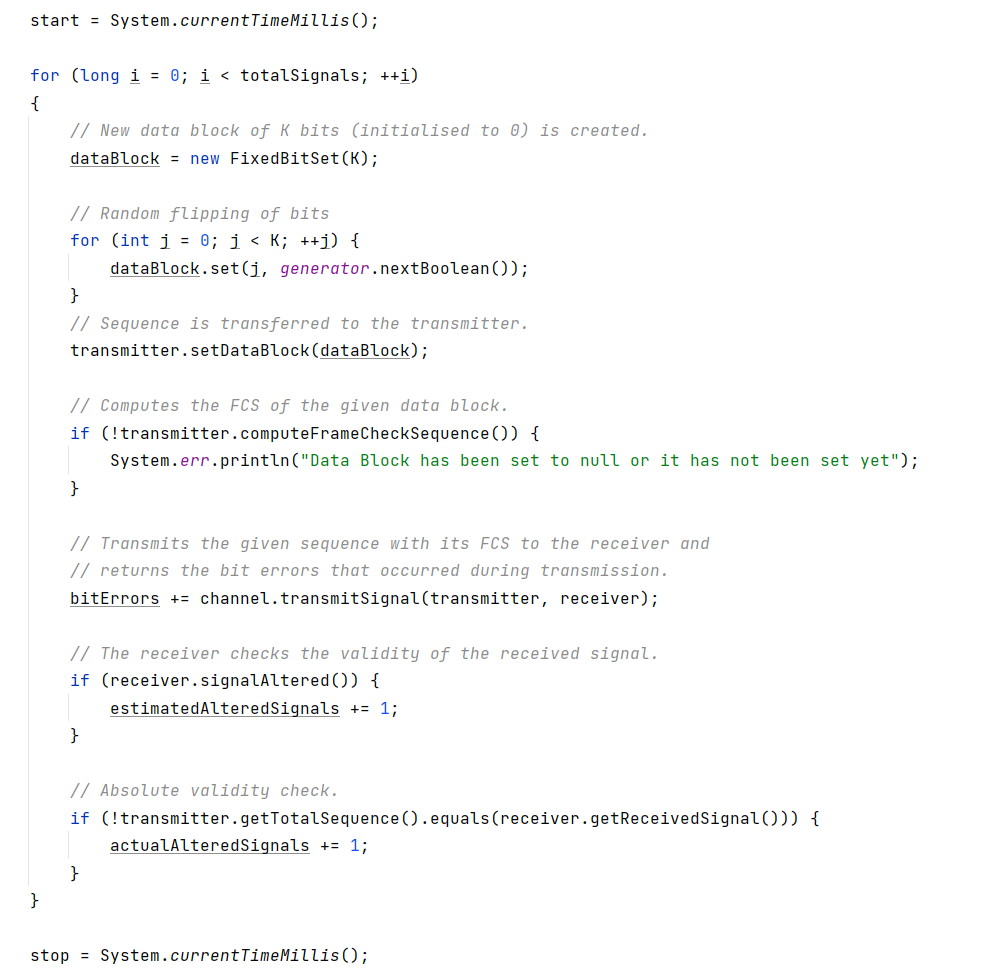
Then, the necessary initializations are made:



The variables for recording metric data are declared.

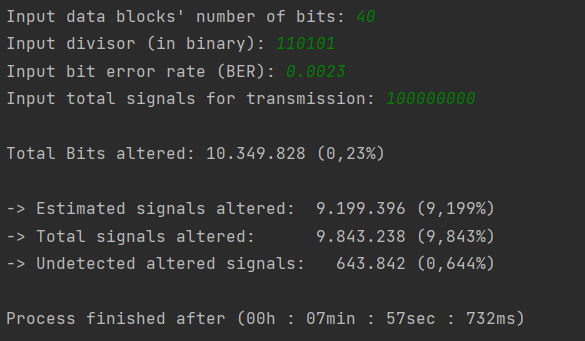


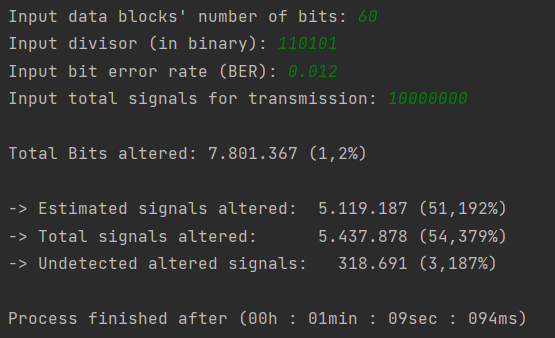
The main loop is performed:

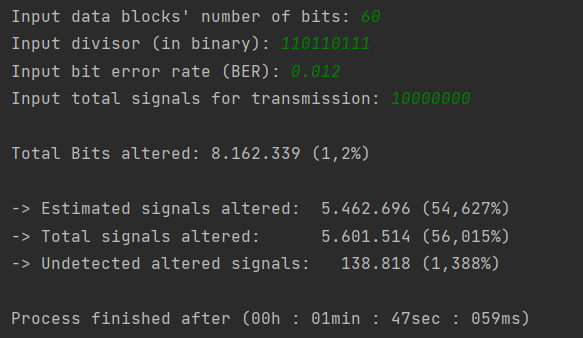


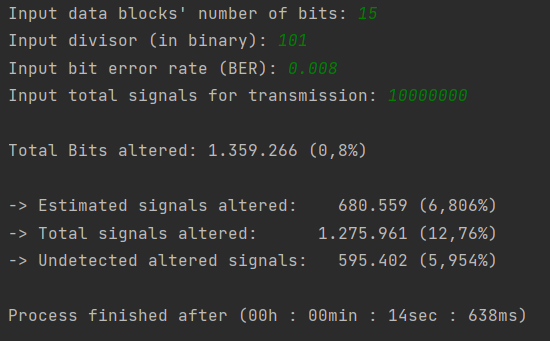
And the results are printed.

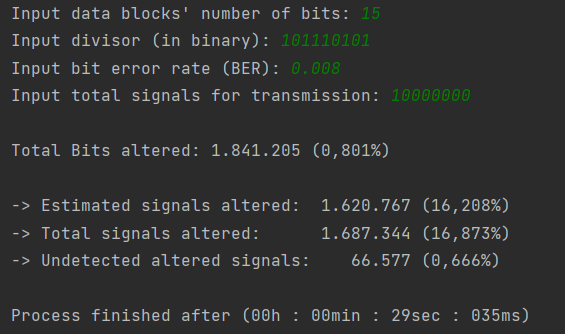
**Indicative Results:**











Part B:

