PROBLEM DEFINITION

Given a Player class - an instance of which can communicate with other Players.

The requirements are as follows:

1. Create 2 Player instances.
2. One of the players should send a message to the second player (let's call this player "initiator").
3. When a player receives a message, it should reply with a message that contains the received message concatenated with the value of a counter holding the number of messages this player already sent.
4. Finalize the program (gracefully) after the initiator sent 10 messages and received back 10 messages.
5. Both players should run in the same Java process.

PROGRAM SUMMARY

- This is a Java program demonstrating inter-thread communication by using a two-stream message buffer.
- It simulates the communication between 2 distinct players by applying the multithreading principles.
- There are 2 subclasses of the base Player class corresponding to the initiator and receiver players: InitiatorPlayer and ReceiverPlayer.
- InitiatorPlayer and ReceiverPlayer classes have their own threads.
- The initiator player is responsible for starting and finishing the communication. It starts the communication with a **"Hello 0"** message. Once it sends and receives 10 messages back, it sends a **"FINISH"** message to the receiver player and terminates its own thread.
- The receiver player is responsible for responding back to the messages of the initiator player. Once it gets a **"FINISH"** message from the initiator player, it terminates its own thread.
- Except for the first and last messages that the initiator player sends **("Hello 0" and "FINISH")**, all messages are created by adding the message count that the current player already sent to the received message from the opposite player (excluding the message that is being sent in that cycle).
- The initiator player sends its messages via a string allocated for the stream from the initiator player to the receiver player. The receiver player receives the messages from the initiator player by using the same string. Because both players use the same string simultaneously, synchronization is established when they access this string.
- The receiver player sends its messages via a string allocated for the stream from the receiver player to the initiator player. The initiator player receives the messages from the receiver player by using the same string. Because both players use the same string simultaneously, synchronization is established when they access this string.

CLASS SUMMARY

There are 5 classes in the program. Their summaries are given below:

Player

- It is an abstract class that constructs a base for the InitiatorPlayer and ReceiverPlayer classes with common attributes and methods.

- It has a MessageBuffer instance in which 2 distinct strings for the message streams from the initiator player to the receiver player and from the receiver player to the initiator player exist.

InitiatorPlayer

- It extends the Player class and implements the Runnable interface.
- The receivedMessageCount variable was added to this class. It does not occur in the ReceiverPlayer class because the InitiatorPlayer class controls the termination of the threads by using this variable. Once this variable reaches 10, it means that the initiator player sent and received 10 messages. So, it terminates the initiator thread and sends a **"FINISH"** message to the receiver thread.
- The run() method that provides the multithreading capability was added to this class.
- The run() method is responsible for the operations summarized in the program summary.

ReceiverPlayer

- It extends the Player class and implements the Runnable interface.
- The only addition to this class is the run() method which provides the multithreading capability.
- The run() method is responsible for the operations summarized in the program summary. Of course, this run() method is different from the run() method of InitiatorPlayer because their functionalities are different.

MessageBuffer

- It has 2 different strings for 2 different streams explained above.
- It has also 2 boolean instance variables controlling the synchronized access to the strings.
- It has 4 methods that are responsible for putting messages to the strings and receiving messages from them thread-safely. The run() methods of the players use these methods.
- The initiator and receiver players share the same MessagBuffer object.

App

- This is the main class of the program where the main method exists.
- It instantiates all necessary objects.
- It starts the threads of the initiator and receiver objects and waits for their execution to terminate.

NOTES

- This project was developed with Java 1.8.
- This is a Maven project created in Eclipse.
- If you double-click on the **run.sh** file in the root directory, all classes compile, and the program runs.