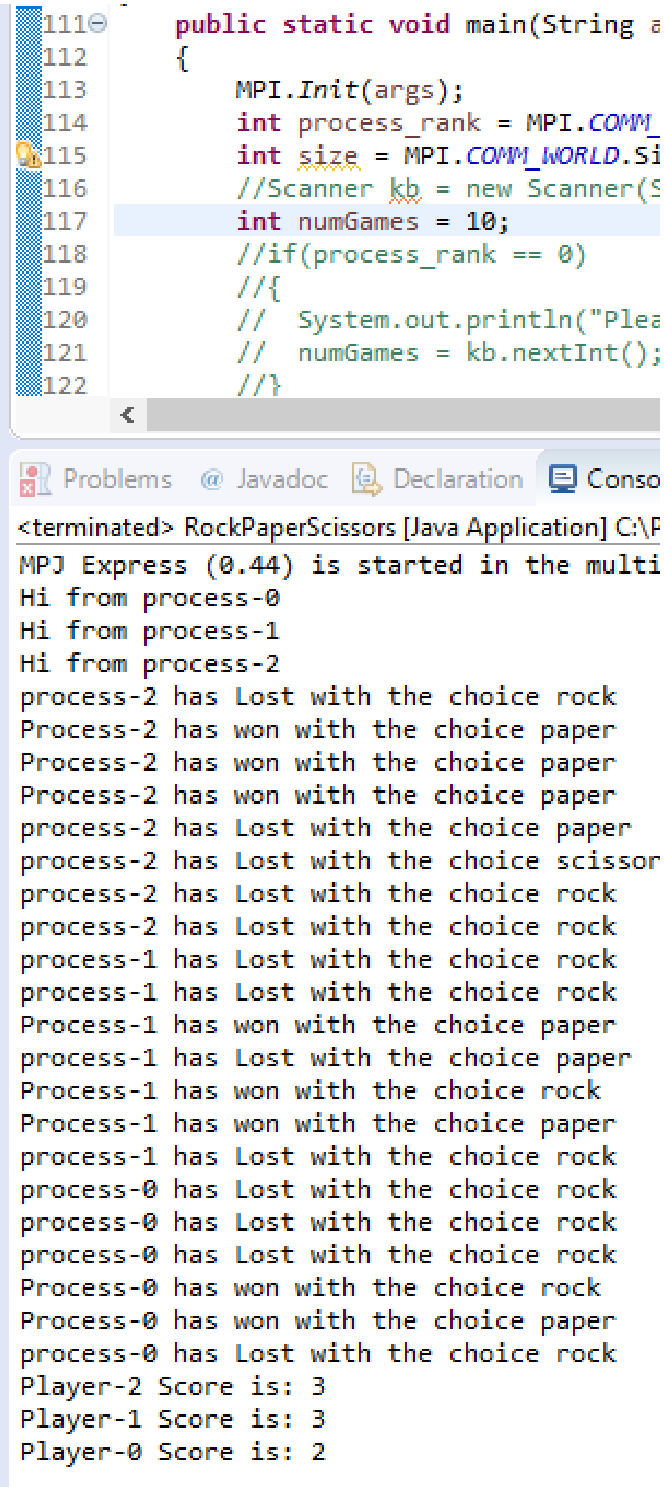
Kyle Kirtland #010860720

https://github.com/Redster11/CS_3700/tree/master/Final_Part_1/RockPaperScissorsMPI

10 games

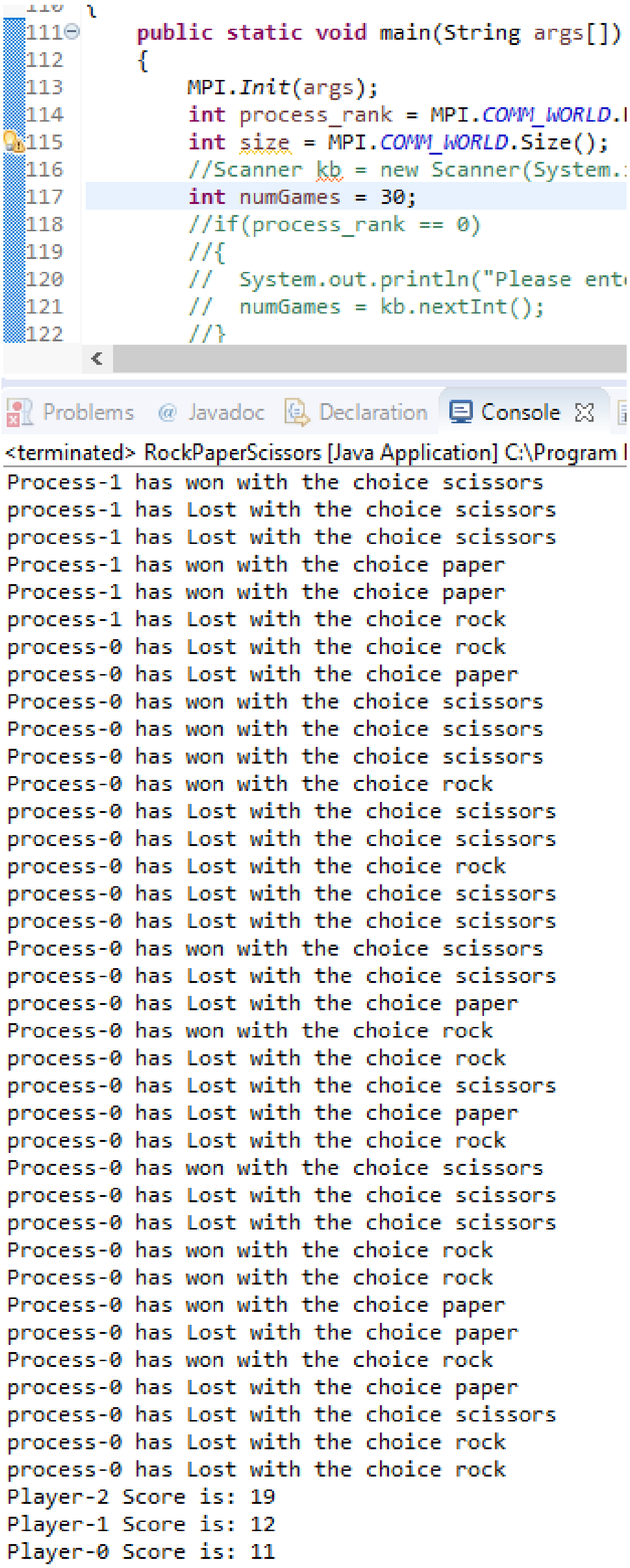


```
    public static void main(String a
    {
        MPI.Init(args);
        int process_rank = MPI.COMM_
        int size = MPI.COMM_WORLD.Si
        //Scanner kb = new Scanner(S
        int numGames = 10;
        //if(process_rank == 0)
        //{
        //  System.out.println("Plea
        //  numGames = kb.nextInt();
        //}
```

<terminated> RockPaperScissors [Java Application] C:\P

```
MPJ Express (0.44) is started in the multi
Hi from process-0
Hi from process-1
Hi from process-2
process-2 has Lost with the choice rock
Process-2 has won with the choice paper
Process-2 has won with the choice paper
Process-2 has won with the choice paper
process-2 has Lost with the choice paper
process-2 has Lost with the choice scissor
process-2 has Lost with the choice rock
process-2 has Lost with the choice rock
process-1 has Lost with the choice rock
process-1 has Lost with the choice rock
Process-1 has won with the choice paper
process-1 has Lost with the choice paper
Process-1 has won with the choice rock
Process-1 has won with the choice paper
process-1 has Lost with the choice rock
process-0 has Lost with the choice rock
process-0 has Lost with the choice rock
process-0 has Lost with the choice rock
Process-0 has won with the choice rock
Process-0 has won with the choice paper
process-0 has Lost with the choice rock
Player-2 Score is: 3
Player-1 Score is: 3
Player-0 Score is: 2
```

30 games



```
    public static void main(String args[])
    {
        MPI.Init(args);
        int process_rank = MPI.COMM_WORLD.I
        int size = MPI.COMM_WORLD.Size();
        //Scanner kb = new Scanner(System.i
        int numGames = 30;
        //if(process_rank == 0)
        //{
        //  System.out.println("Please ent
        //  numGames = kb.nextInt();
        //}
```

```
<terminated> RockPaperScissors [Java Application] C:\Program I
Process-1 has won with the choice scissors
process-1 has Lost with the choice scissors
process-1 has Lost with the choice scissors
Process-1 has won with the choice paper
Process-1 has won with the choice paper
process-1 has Lost with the choice rock
process-0 has Lost with the choice rock
process-0 has Lost with the choice paper
Process-0 has won with the choice scissors
Process-0 has won with the choice scissors
Process-0 has won with the choice scissors
Process-0 has won with the choice rock
process-0 has Lost with the choice scissors
process-0 has Lost with the choice scissors
process-0 has Lost with the choice rock
process-0 has Lost with the choice scissors
process-0 has Lost with the choice scissors
Process-0 has won with the choice scissors
process-0 has Lost with the choice scissors
process-0 has Lost with the choice paper
Process-0 has won with the choice rock
process-0 has Lost with the choice rock
process-0 has Lost with the choice scissors
process-0 has Lost with the choice paper
process-0 has Lost with the choice rock
Process-0 has won with the choice scissors
process-0 has Lost with the choice scissors
process-0 has Lost with the choice scissors
Process-0 has won with the choice rock
Process-0 has won with the choice rock
Process-0 has won with the choice paper
process-0 has Lost with the choice paper
Process-0 has won with the choice rock
process-0 has Lost with the choice paper
process-0 has Lost with the choice scissors
process-0 has Lost with the choice rock
process-0 has Lost with the choice rock
Player-2 Score is: 19
Player-1 Score is: 12
Player-0 Score is: 11
```

Findings:

After some time working on this project, I have found that it is quite hard to take user input to run this type of program. It requires a way to pause all processes but since they are all running at the same time it is not going to allow for a singular input and when an input is used on all of them it makes it not run even when all values are the same. This might be due to the amount of time that passes since it is expecting for the moment that it starts to send to be all at the same point. Maybe if the use of a send function was used at the beginning of the program from the first player then they can all match up in the end.

We were talking about how it is slow to send one value at a time, so I decided to just pass all values for each player to each other making it very fast to start all the processes. Since they all must wait on 2 sends per process. Instead of 2 send processes per play.

The send and receive functions of MPI are very temperamental with ordering… if they are not in the same order as the way they are send receiving the program seemed to break. it might have also been due to not matching tags. To make this run properly I used a few tricks which was ordering the sends and receives up in the code I also made the tags the same number as the player number that it pertained too. These combination of changed seemed to fix the issues that I was running into for the sending and receiving system.

SET UP:

To start of the project I had to setup my environment to be able to run MPI projects/code.

To get this done I learned that using Eclipse IDE makes it easy to work with it.

When you start a new project indicate that you are going to be using an external jar on the second page. Then navigate to the folder that has the MPJ folder in it and go to lib and choose MPJ.jar (if this does not work then you can also include MPI.jar).

After that you are good to go to the next step of setup.

Add a class file and run it.

Right click on the .java file and go under “run as” and choose “run configuration”

In here we need to add a few new items.

First is we need to go to Environment and click the add button to add a variable called “MPJ_HOME” the destination is the location of the folder that holds all of the files for MPI.

Second go to the Arguments section, under VM arguments add the following to it.
“-jar ${MPJ_HOME}/lib/starter.jar -np 3”

The first part makes it so that we can run the MPI.jar and makes the entire system allow for auto filling.

The -np 3 means that we want to start 3 processes. In our cases we want to have 3 players so this was set to 3.