Rojin Atefimehr

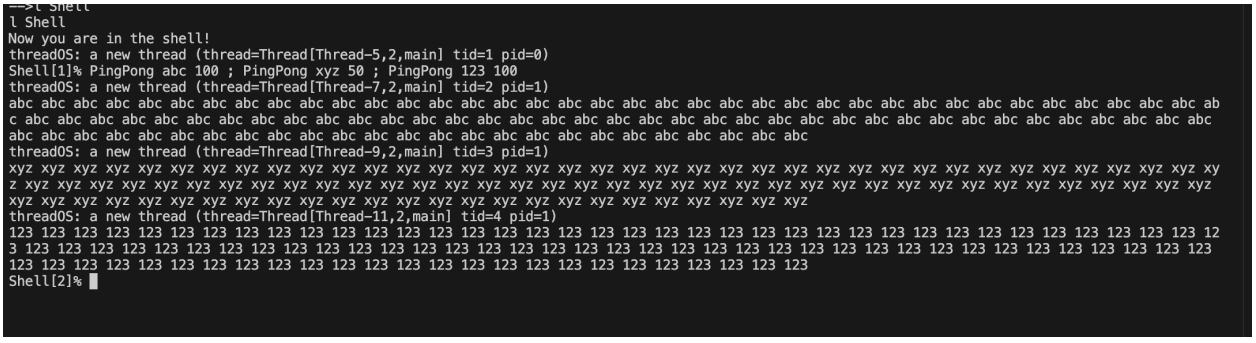
Program2

1) For testing the Shell.java, first we need to make sure that we are in the right directory and cd to the directory where the Shell.java exists. After being sure that we are in the right directory we need to use “javac Shell.java” to compile the program, then “java Boot”, finally “l Shell”. Now we can test the program with the given test cases.

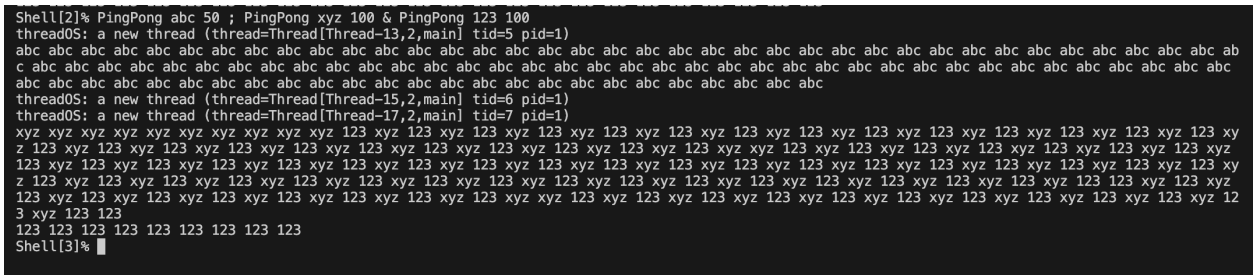
2) In the run method I used a while loop for continually running the program and waits for user to enter their input. I used stringbuffer because it is much more flexible and takes less space in the memory. After that I converted the buffer format to the string. It also shows the shell[i]% for entering the input and increases the number (i) as the user enter more input (number++). When the user enter exit, I used sysLib.exit() to exit from the shell otherwise we go to process the input commands.
for processing the commands, it takes the user input and separate it based on ‘;’ or ‘&’.
For ‘;’, it executes a command in the order they were entered. It waits for one command to complete before starting the next one.
For ‘&’, it is meant to run simultaneously. I used SysLib.exec(args) to start each command in a new thread, and for keeping track of the thread IDs. After executing all concurrent commands, it waits for all of them to finish by SysLib.join() for each thread ID to make sure all concurrent process is completed before proccing another one.

3)

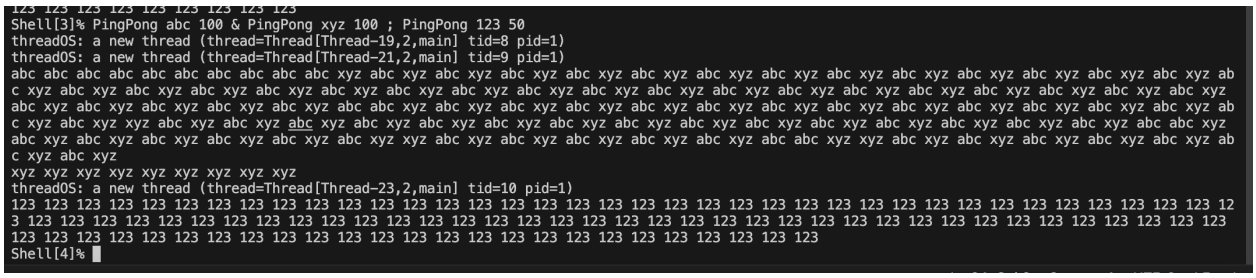
Test 1) `PingPong abc 100 ; PingPong xyz 50 ; PingPong 123 100`



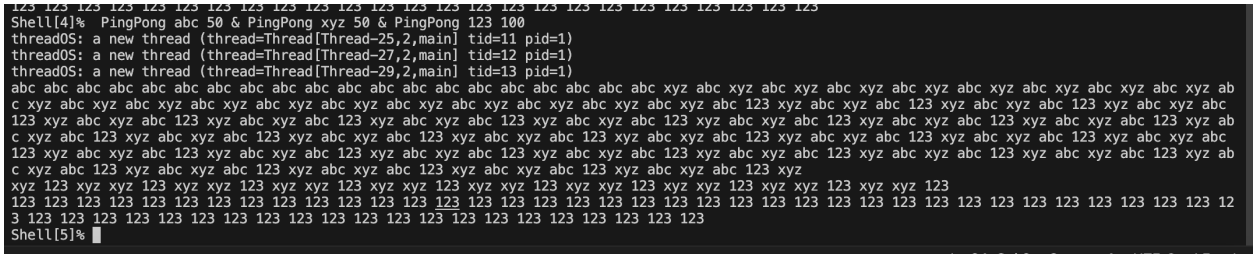
Test2) `PingPong abc 50 ; PingPong xyz 100 & PingPong 123 100`



Test3) `PingPong abc 100 & PingPong xyz 100 ; PingPong 123 50`



Test4) `PingPong abc 50 & PingPong xyz 50 & PingPong 123 100`



I also added my program here:

```
public class Shell extends Thread{

// shell starts at 1 like shell[1]
static int number = 1;
static boolean run = true;

public Shell( ) {

}

// it has to have run function, if the name was somethingelse, it would fail
public void run(){
// keep the shell running like infinity
while (run){
    StringBuffer buffer = new StringBuffer();
    // to print shell[i]%
    SysLib.cout("Shell["+ number +"]% ");
    // read the input from the user
    SysLib.cin(buffer);
    // we need to convert the buffer into a string
    String input =  buffer.toString();
```

```
    // when user do not enter their input
    if (input.isEmpty()){
        SysLib.cout("you did not eneter your input, try again!");
        SysLib.cout("\n");
        continue;
    }
     // when the user eneter exit, it should exit from the shell
    if (input.equalsIgnoreCase("exit")){
        SysLib.cout("you exit the shell!");
        SysLib.exit();
        break;
    }
    // processes the input
    else{
    number++;

    String[] sequential = input.split(";");

    for (String seq: sequential) {
        // split the sequential when has & while going through it
        String[] concurrent = seq.split("&");
        // keep track of thread IDs for concurrent
        int length = concurrent.length;
        int[] thread_ids = new int[length];

    for (int i = 0; i< length; i++) {
        String command = concurrent[i];
        if (!command.isEmpty()) {
            // convert command string into arguments
            String[] args = SysLib.stringToArgs(command);
            // run the command and store the thread ID
            thread_ids[i] = SysLib.exec(args);
        }
    }
        // for concuurent we need to wait for thread to complete
            for (int tid : thread_ids) {
                // if the invalid thread id is -1
                if (tid != -1) {
                    SysLib.join();
                }
            }


    }
}

}
}
```

```
}
```