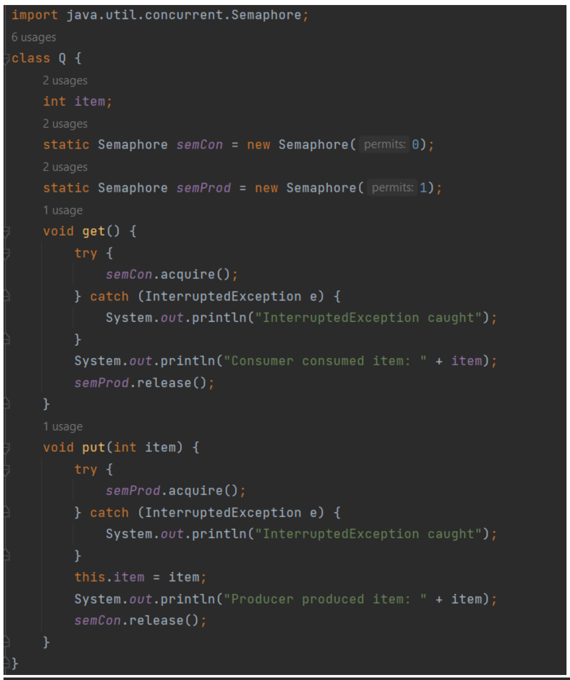
Operating Systems

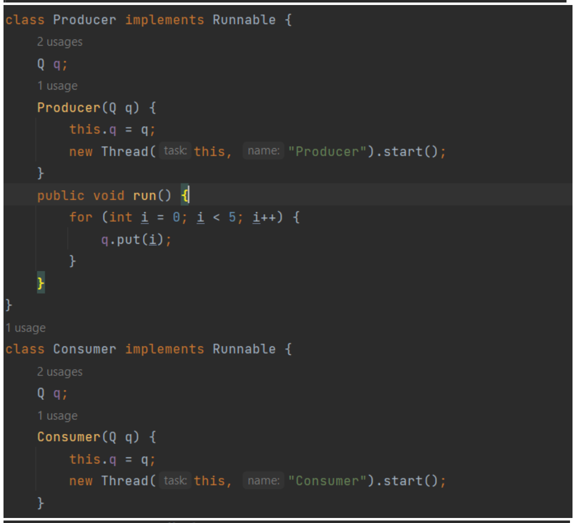
<u>Assignment-6</u>

Implementation of Interprocess Communication

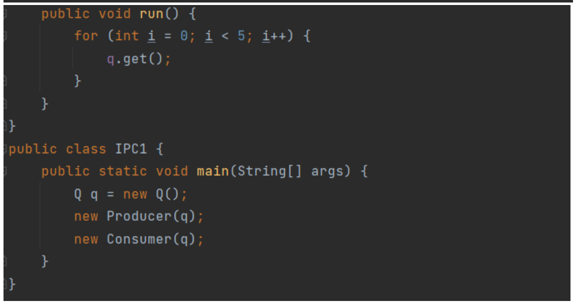
A. IPC Using Semaphore- Producer and Consumer Problem
Code:



```java
import java.util.concurrent.Semaphore;
6 usages
class Q {
    2 usages
    int item;
    2 usages
    static Semaphore semCon = new Semaphore( permits: 0);
    2 usages
    static Semaphore semProd = new Semaphore( permits: 1);
    1 usage
    void get() {
        try {
            semCon.acquire();
        } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
        }
        System.out.println("Consumer consumed item: " + item);
        semProd.release();
    }
    1 usage
    void put(int item) {
        try {
            semProd.acquire();
        } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
        }
        this.item = item;
        System.out.println("Producer produced item: " + item);
        semCon.release();
    }
}
```

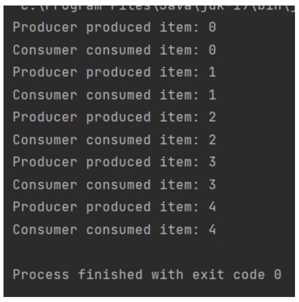


```java
class Producer implements Runnable {
    2 usages
    Q q;
    1 usage
    Producer(Q q) {
        this.q = q;
        new Thread( task: this,  name: "Producer").start();
    }
    public void run() {
        for (int i = 0; i < 5; i++) {
            q.put(i);
        }
    }
}
1 usage
class Consumer implements Runnable {
    2 usages
    Q q;
    1 usage
    Consumer(Q q) {
        this.q = q;
        new Thread( task: this,  name: "Consumer").start();
    }
```



```java
    public void run() {
        for (int i = 0; i < 5; i++) {
            q.get();
        }
    }
}
public class IPC1 {
    public static void main(String[] args) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
    }
}
```

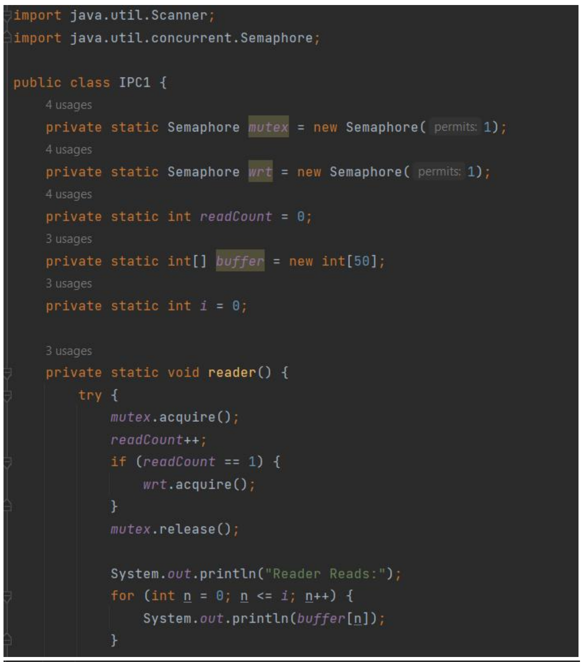
Output:



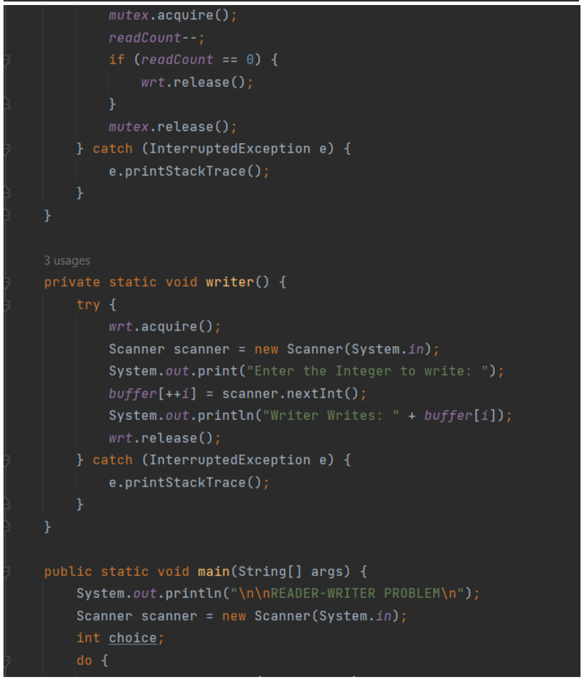
```
C:\Program Files\Java\jdk-17\bin\
Producer produced item: 0
Consumer consumed item: 0
Producer produced item: 1
Consumer consumed item: 1
Producer produced item: 2
Consumer consumed item: 2
Producer produced item: 3
Consumer consumed item: 3
Producer produced item: 4
Consumer consumed item: 4

Process finished with exit code 0
```

B. IPC Using Semaphore- Readers and Writers Problem

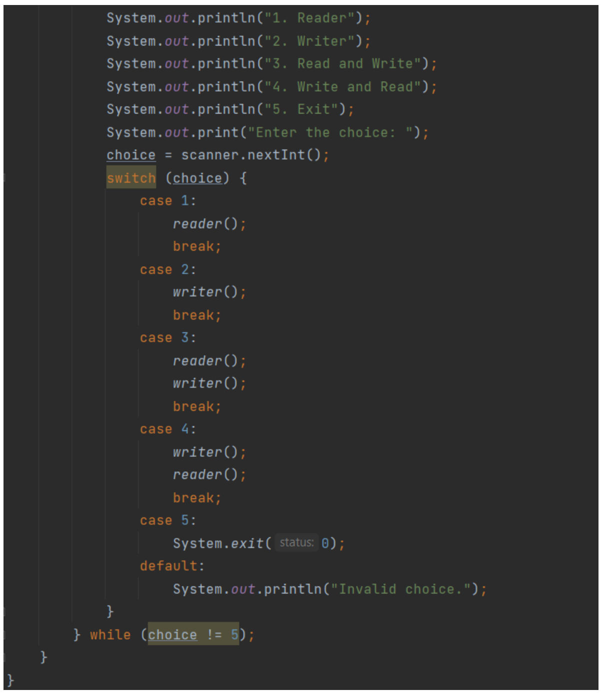
Code:



```java
import java.util.Scanner;
import java.util.concurrent.Semaphore;

public class IPC1 {
    4 usages
    private static Semaphore mutex = new Semaphore( permits: 1);
    4 usages
    private static Semaphore wrt = new Semaphore( permits: 1);
    4 usages
    private static int readCount = 0;
    3 usages
    private static int[] buffer = new int[50];
    3 usages
    private static int i = 0;

    3 usages
    private static void reader() {
        try {
            mutex.acquire();
            readCount++;
            if (readCount == 1) {
                wrt.acquire();
            }
            mutex.release();

            System.out.println("Reader Reads:");
            for (int n = 0; n <= i; n++) {
                System.out.println(buffer[n]);
            }
```

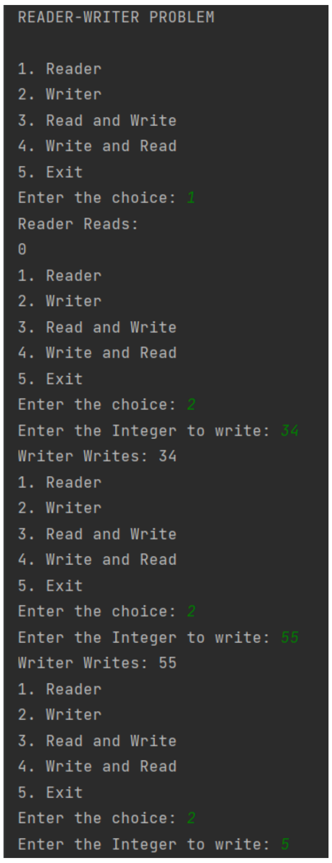


```java
            mutex.acquire();
            readCount--;
            if (readCount == 0) {
                wrt.release();
            }
            mutex.release();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    3 usages
    private static void writer() {
        try {
            wrt.acquire();
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter the Integer to write: ");
            buffer[++i] = scanner.nextInt();
            System.out.println("Writer Writes: " + buffer[i]);
            wrt.release();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        System.out.println("\n\nREADER-WRITER PROBLEM\n");
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
```

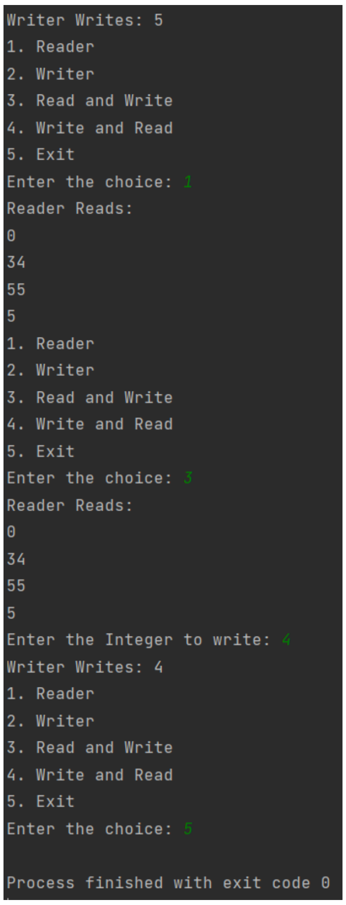


```java
                System.out.println("1. Reader");
                System.out.println("2. Writer");
                System.out.println("3. Read and Write");
                System.out.println("4. Write and Read");
                System.out.println("5. Exit");
                System.out.print("Enter the choice: ");
                choice = scanner.nextInt();
                switch (choice) {
                    case 1:
                        reader();
                        break;
                    case 2:
                        writer();
                        break;
                    case 3:
                        reader();
                        writer();
                        break;
                    case 4:
                        writer();
                        reader();
                        break;
                    case 5:
                        System.exit( status: 0);
                    default:
                        System.out.println("Invalid choice.");
                }
            } while (choice != 5);
        }
}
```

Output:



```
READER-WRITER PROBLEM

1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 1
Reader Reads:
0
1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 2
Enter the Integer to write: 34
Writer Writes: 34
1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 2
Enter the Integer to write: 55
Writer Writes: 55
1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 2
Enter the Integer to write: 5
```


```
Writer Writes: 5
1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 1
Reader Reads:
0
34
55
5
1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 3
Reader Reads:
0
34
55
5
Enter the Integer to write: 4
Writer Writes: 4
1. Reader
2. Writer
3. Read and Write
4. Write and Read
5. Exit
Enter the choice: 5

Process finished with exit code 0
```

Submitted by:
Harshita Pasupuleti
21BCE8421