UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Disciplina: Estruturas de Dados I

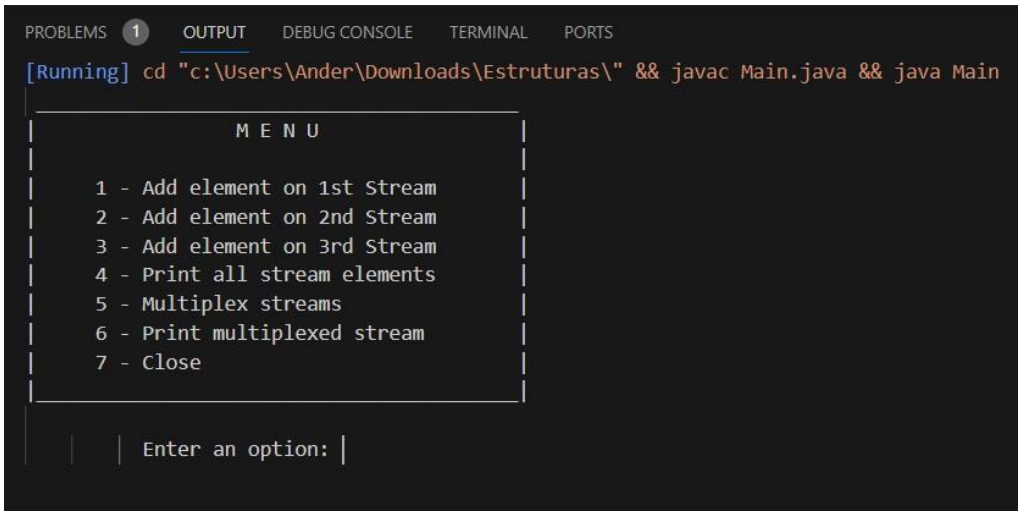
Atividade Multiplexação – Fila Circular
Dupla ou Trio

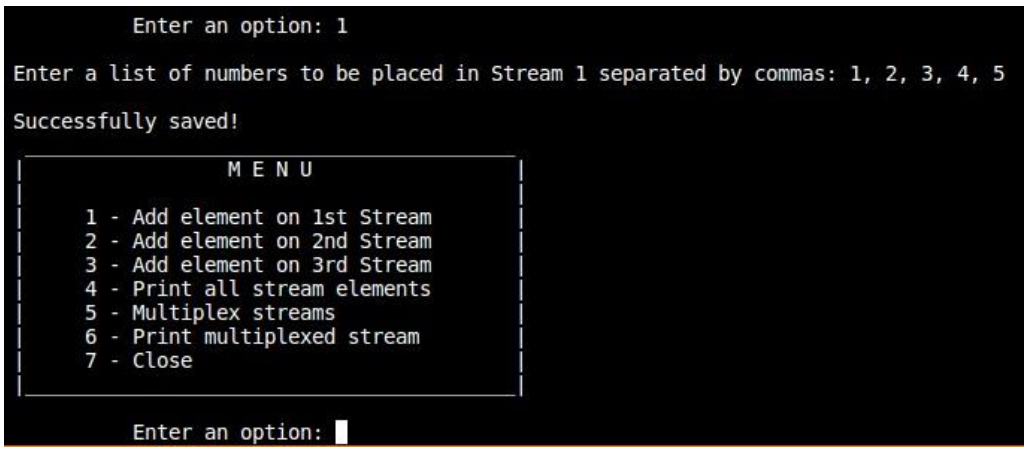
Nome do(a) aluno(a)	TIA
Anderson Correa Nicodemo	32285671
Gustavo Garabetti Munhoz	42211956

Utilizando a classe Fila Circular que disponibilizada em aula, realizar a implementação da atividade Multiplexação.

Realizar dois (2) testes de cada item de menu, colocá-los no relatório a seguir (*Printscreen*) e incluir os códigos fontes desenvolvidos em uma seção denominada Apêndice, além disso, enviar o código fonte compactado.

Resolução





```
Enter a list of numbers to be placed in Stream 1 separated by commas: 1, 2, 3, 4, 5

Successfully saved!
 _____________________________________
|                M E N U              |
|                                     |
|     1 - Add element on 1st Stream   |
|     2 - Add element on 2nd Stream   |
|     3 - Add element on 3rd Stream   |
|     4 - Print all stream elements   |
|     5 - Multiplex streams           |
|     6 - Print multiplexed stream    |
|     7 - Close                       |
|_____________________________________|

         Enter an option: 4

Stream Elements:
Stream 1: [1, 2, 3, 4, 5]
Stream 2 is empty!
Stream 3 is empty!
```

```
         Enter an option: 2

Enter a list of numbers to be placed in Stream 2 separated by commas:6, 7, 8, 9

Successfully saved!
 _____________________________________
|                M E N U              |
|                                     |
|     1 - Add element on 1st Stream   |
|     2 - Add element on 2nd Stream   |
|     3 - Add element on 3rd Stream   |
|     4 - Print all stream elements   |
|     5 - Multiplex streams           |
|     6 - Print multiplexed stream    |
|     7 - Close                       |
|_____________________________________|

         Enter an option: 4

Stream Elements:
Stream 1: [1, 2, 3, 4, 5]
Stream 2: [6, 7, 8, 9]
Stream 3 is empty!
```

```
         Enter an option: 3

Enter a list of numbers to be placed in Stream 3 separated by commas:10, 348534

Successfully saved!
 _____________________________________
|                M E N U              |
|                                     |
|     1 - Add element on 1st Stream   |
|     2 - Add element on 2nd Stream   |
|     3 - Add element on 3rd Stream   |
|     4 - Print all stream elements   |
|     5 - Multiplex streams           |
|     6 - Print multiplexed stream    |
|     7 - Close                       |
|_____________________________________|

         Enter an option: 4

Stream Elements:
Stream 1: [1, 2, 3, 4, 5]
Stream 2: [6, 7, 8, 9]
Stream 3: [10, 348534]
```

```
          Enter an option: 5

Successfully multiplexed streams!

 ________________________________________
|                M E N U                 |
|                                        |
|     1 - Add element on 1st Stream      |
|     2 - Add element on 2nd Stream      |
|     3 - Add element on 3rd Stream      |
|     4 - Print all stream elements      |
|     5 - Multiplex streams              |
|     6 - Print multiplexed stream       |
|     7 - Close                          |
|________________________________________|

          Enter an option: 4

Stream Elements:

All the Streams are empty! Please add elements to the streams and try multiplexing again!
```

```
 ________________________________________
|                M E N U                 |
|                                        |
|     1 - Add element on 1st Stream      |
|     2 - Add element on 2nd Stream      |
|     3 - Add element on 3rd Stream      |
|     4 - Print all stream elements      |
|     5 - Multiplex streams              |
|     6 - Print multiplexed stream       |
|     7 - Close                          |
|________________________________________|

          Enter an option: 6

Multiplexed stream: [(1,1), (6,2), (10,3), (2,1), (7,2), (348534,3), (3,1), (8,2), (4,1), (9,2), (5,1)]
```

```
          Enter an option: 6

Multiplexed stream are Empty! Please add elements to the streams and try multiplexing again!
 ________________________________________
|                M E N U                 |
|                                        |
|     1 - Add element on 1st Stream      |
|     2 - Add element on 2nd Stream      |
|     3 - Add element on 3rd Stream      |
|     4 - Print all stream elements      |
|     5 - Multiplex streams              |
|     6 - Print multiplexed stream       |
|     7 - Close                          |
|________________________________________|

          Enter an option: 4

Stream Elements:

All the Streams are empty! Please add elements to the streams and try multiplexing again!
```

```
          Enter an option: 1

Enter a list of numbers to be placed in Stream 1 separated by commas: 1, 2, 3, 4

Successfully saved!

 ________________________________________
|                MENU                    |
|                                        |
|     1 - Add element on 1st Stream      |
|     2 - Add element on 2nd Stream      |
|     3 - Add element on 3rd Stream      |
|     4 - Print all stream elements      |
|     5 - Multiplex streams              |
|     6 - Print multiplexed stream       |
|     7 - Close                          |
|________________________________________|

          Enter an option: 4

Stream Elements:
Stream 1: [1, 2, 3, 4]
Stream 2 is empty!
Stream 3 is empty!
```

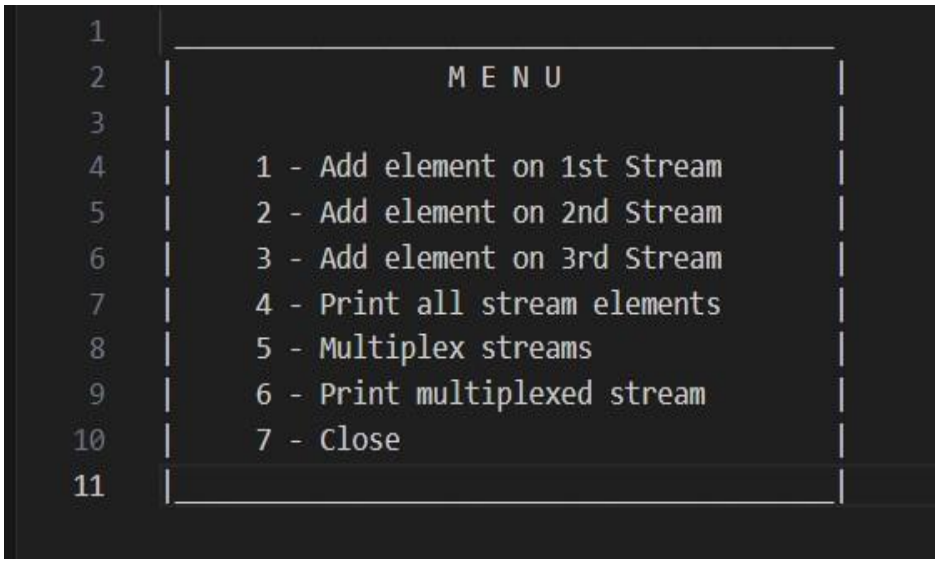
```
          Enter an option: 5

Successfully multiplexed streams!


 ________________________________________
|                MENU                    |
|                                        |
|     1 - Add element on 1st Stream      |
|     2 - Add element on 2nd Stream      |
|     3 - Add element on 3rd Stream      |
|     4 - Print all stream elements      |
|     5 - Multiplex streams              |
|     6 - Print multiplexed stream       |
|     7 - Close                          |
|________________________________________|

          Enter an option: 6

Multiplexed stream: [(1,1), (2,1), (3,1), (4,1)]
```

```
 ________________________________________
|                MENU                    |
|                                        |
|     1 - Add element on 1st Stream      |
|     2 - Add element on 2nd Stream      |
|     3 - Add element on 3rd Stream      |
|     4 - Print all stream elements      |
|     5 - Multiplex streams              |
|     6 - Print multiplexed stream       |
|     7 - Close                          |
|________________________________________|

          Enter an option: 7

Program Closed! Thank you very much for using the Multiplexer!
gustavo@garabs-ubuntu:~/Documentos/3o-Periodo-Computacao/Estruturas/Proj1$
```

UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Disciplina: Estruturas de Dados I

Apêndice

Menu



Classe IO Functions

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class IOfunctions{
    public static void Show_Menu(){ //Realiza a leitura do menu através de um TXT
        File file = new File("menu.txt");

        try {
            Scanner scan = new Scanner(file);

            while(scan.hasNextLine()) System.out.println(scan.nextLine());

            scan.close();
        } catch (FileNotFoundException err){
            System.out.println("Error opening file:" + err.getMessage());
        }
    }

    public static int choice(){ //Salva a choice do usuario
        Show_Menu();
        Scanner input = new Scanner(System.in);
        System.out.print("\n          Enter an option: ");
        int choice = Integer.parseInt(input.nextLine());
        while(choice > 7 || choice < 1){
            Show_Menu();
            System.out.print("\nInvalid option, type again: ");
            choice = input.nextInt();
        }
        return choice;
    }
}
```

UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Disciplina: Estruturas de Dados I

Classe CircularQueue

```java
public class CircularQueue{
    private int begin, end, qtd;
    private StreamObject queue[];
    private static final int DEFAULT_TAM = 100;

    public CircularQueue(int size_queue){
        this.begin = this.end = this.qtd = 0;
        this.queue = new StreamObject[size_queue];
    }

    public CircularQueue(){ this(DEFAULT_TAM); }

    public int size(){ return this.qtd; }

    public boolean isEmpty(){ return this.size() == 0; }

    public boolean isFull(){ return this.size() == this.queue.length; }

    public void enqueue(StreamObject object){
        if(!this.isFull()){
            this.queue[this.end++] = object;
            this.end %= this.queue.length;
            this.qtd++;
        }else System.out.println("Queue overStreamObject!");
    }

    public StreamObject dequeue(){
        StreamObject aux = new StreamObject();
        if(!this.isEmpty()){
            this.qtd--;
            this.begin = ++this.begin % this.queue.length;
            return queue[(this.begin - 1 + this.queue.length) % this.queue.length];
        }else System.out.println("Queue underStreamObject!");
        return aux;
    }

    public StreamObject front(){
        StreamObject aux = new StreamObject();
```

```
        if(!this.isEmpty()) return this.queue[this.begin];
        System.out.println("Queue underStreamObject");
        return aux;
    }

    public StreamObject rear(){
        StreamObject aux = new StreamObject();
        if(!this.isEmpty()) return this.queue[(this.end - 1 + this.queue.length) % this.queue.length];
        System.out.println("Queue underStreamObject");
        return aux;
    }

    public void AddElementsOnStream(int elements[], int ID){ //Adiciona os elementos a fila
        for(int i = 0; i < elements.length; i++){
            StreamObject obj = new StreamObject(ID, elements[i]);
            this.enqueue(obj);
        }
    }

    public boolean MultiplexStreams(CircularQueue F[]){ //Realiza a multiplexação
        if(F[0].isEmpty() && F[1].isEmpty() && F[2].isEmpty()){
            return false;
        }
        int cont = 0; //Inicializa contador e um vetor interatores
        while(!F[0].isEmpty() || !F[1].isEmpty() || !F[2].isEmpty()){
            if(!F[cont % 3].isEmpty()){ //Verifica se ainda há elementos não processados na fila
                this.enqueue(F[cont % 3].dequeue()); //Insere o elemento da frente da fila atual para a fila MUXF
            }
            cont++;
        }
        return true;
    }

    public int PrintElementsOfStream(int ID){ //Imprime os elementos da fila
        if(this.isEmpty()){
            return 0;
        }
        System.out.printf("Stream %d: [", ID+1);
        for(int i = 0; i < this.size(); i++){
            System.out.printf("%d", this.front().getData()); //Pega o elemento inicial da fila
            if(i != this.size()-1) System.out.printf(", "); //Separa o elemento com virgula
            this.enqueue(this.dequeue());
        }
        System.out.printf("]\n");
        return 1;
    }

    public void PrintElementsOfMUXStream(){ //Imprime os elementos da fila
        if(this.isEmpty()){
            System.out.print("\nMultiplexed stream are Empty! Please add elements to the streams and try multiplexing again!\n");
            return;
        }
        System.out.print("\nMultiplexed stream: [");
        int tam = this.size();
        for(int i = 0; i < tam; i++){
            System.out.printf("(%d,%d)", this.front().getData(), this.front().getID()+1);
            if(i != tam-1) System.out.printf(", "); //Separa o elemento com virgula
            this.dequeue();
        }
        System.out.printf("]\n");
    }

}
```

Classe AuxiliarMethods

```java
public class AuxiliarMethods {
    public static int[] split(String values){
        values = values.trim();
        int[] aux = new int[values.length()];
        int cont = 0;
        String buffer = new String(""); //Define um buffer para salvar o elemento
        for(int i = 0; i < values.length(); i++){  //Adiciona os elementos a fila, identificando-os atráves da separação por virgula
            while(i < values.length() && values.charAt(i) != ','){
                buffer += values.charAt(i);
                i++;
            }
            int valor = Integer.parseInt(buffer.trim());
            aux[cont++] = valor;
            buffer = ""; //Reseta o buffer
        }
        int[] ans = new int[cont];
        for(int i = 0; i < cont; i++) ans[i] = aux[i];
        return ans;
    }
}
```

Classe StreamObject

```java
import java.util.Scanner;

public class StreamObject{ //Classe fila
    private int streamID, streamData;
    public static final int ID_DEFAULT = 0, DATA_DEFAULT = 0;
    Scanner input = new Scanner(System.in);

    public StreamObject(int ID, int data){ //Construtores da fila
        setID(ID);
        setData(data);
    }

    public StreamObject(){ this(ID_DEFAULT, DATA_DEFAULT); }

    public int getData(){ return this.streamData; }

    public void setData(int newData){ this.streamData = newData; }

    public int getID(){ return this.streamID; }

    public void setID(int newID){ this.streamID = newID; }
}
```

UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Disciplina: Estruturas de Dados I

Main

```java
import java.util.*;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        CircularQueue[] stream = new CircularQueue[3]; //Cria os fluxos
        CircularQueue MUXstream = new CircularQueue(); //Cria a fila para o multiplexador

        for(int i = 0; i < 3; i++) stream[i] = new CircularQueue(); //Cria 3 filas circulares

        boolean EndOfProgram = false;
        do {
            int choice = IOfunctions.choice(); //Passa a choice do menu realizada pelo usuario

            switch (choice) {
                case 1:
                    System.out.print("\nEnter a list of numbers to be placed in Stream 1 separated by commas: ");
                    String values = input.nextLine(); //Coloca os elementos em values
                    System.out.println();
                    int elements[] = AuxiliarMethods.split(values);
                    stream[(choice-1)%3].AddElementsOnStream(elements, (choice-1)%3);
                    System.out.print("Successfully saved!\n");
                    break;
                case 2:
                    System.out.print("\nEnter a list of numbers to be placed in Stream 2 separated by commas:");
                    values = input.nextLine(); //Coloca os elementos em values
                    System.out.println();
                    elements = AuxiliarMethods.split(values);
                    stream[(choice-1)%3].AddElementsOnStream(elements, (choice-1)%3);
                    System.out.print("Successfully saved!\n");
                    break;
                case 3:
                    System.out.print("\nEnter a list of numbers to be placed in Stream 3 separated by commas:");
                    values = input.nextLine(); //Coloca os elementos em values retirando os espaços
                    System.out.println();
                    elements = AuxiliarMethods.split(values);
                    stream[(choice-1)%3].AddElementsOnStream(elements, (choice-1)%3);
                    System.out.print("Successfully saved!\n");
                    break;
                case 4: //choice 4 imprime os valores das 3 filas
                    int aux[] = new int[3];
                    int EveryStreamIsEmpty = 0;
                    System.out.println("\nStream Elements: ");
                    for(int i = 0; i < 3; i++){
                        aux[i] = stream[i].PrintElementsOfStream(i);
                        EveryStreamIsEmpty += aux[i];
                    }
                    if(EveryStreamIsEmpty == 0) System.out.println("\nAll the Streams are empty! Please add elements to the streams and try multiplexing again!\n");
                    else for(int i = 0; i < 3; i++){
                        if(aux[i] == 0) System.out.printf("Stream %d is empty!\n", i+1);
                    }
                    break;
                case 5: //choice 5 realiza a multiplexação
                    if(MUXstream.MultiplexStreams(stream)){
                        System.out.println("\nSuccessfully multiplexed streams!\n");
                    }else System.out.println("\nAll the Streams are empty! Please add elements to the streams and try multiplexing again!\n");
                    break;
                case 6: //choice 6 imprime os numeros já multiplexados
                    MUXstream.PrintElementsOfMUXStream();
                    break;
                case 7: //choice 7 finaliza o programa
                    EndOfProgram = true;
                    System.out.printf("\nProgram Closed! Thank you very much for using the Multiplexer!\n");
                    break;
                default:
                    break;
            }
        } while (!EndOfProgram); //Se final do programa for diferente de verdadeiro fica em loop

        input.close(); //Fecha o input
    }
}
```