```java
import java.util.Random;
public class Deque {
    Celula primeira;
    Celula ultima;
    int totalElementos = 0;

    Random gerador = new Random();


// //sortList() will sort nodes of the list in ascending order
//     public void sortList() {
//         //Node current will point to head
//         Node current = head, index = null;
//         int temp;

//         if(head == null) {
//             return;
//         }
//         else {
//             while(current != null) {
//                 //Node index will point to node next to current
//                 index = current.next;

//                 while(index != null) {
//                     //If current node's data is greater than index's n
ode data, swap the data between them
//                     if(current.data > index.data) {
//                         temp = current.data;
//                         current.data = index.data;
//                         index.data = temp;
//                     }
//                     index = index.next;
//                 }
//                 current = current.next;
//             }
//         }
//     }

    public void sortList() {
        //Node current will point to head
        Celula atual = primeira;
        Celula proximo;
        Pessoa temp;

        if(primeira == null) {
            return;
        }
        else {
            while(atual.proxima != primeira) {
```

```
                //Node index will point to node next to current
                proximo = atual.proxima;

                while(proximo != primeira) {
                    //If current node's data is greater than index's node
 data, swap the data between them
                    if(atual.elemento.prioridade < proximo.elemento.prior
idade) {
                        temp = atual.elemento;
                        atual.elemento = proximo.elemento;
                        proximo.elemento = temp;
                    }
                    proximo = proximo.proxima;
                }
                atual = atual.proxima;
            }
        }
    }

    public void inserirFim(Pessoa pessoa) {
        // If the list is empty, create a single node
        // circular and doubly list
        if (totalElementos == 0) {
            Celula atual = new Celula(pessoa);
            atual.proxima = atual.anterior = atual;
            primeira = atual;
            totalElementos++;
            return;
        }


        ultima = primeira.anterior;
        // Create Node dynamically
        Celula atual = new Celula(pessoa);

        // Start is going to be next of new_node
        atual.proxima = primeira;

        // Make new node previous of start
        primeira.anterior = atual;

        // Make last previous of new node
        atual.anterior = ultima;

        // Make new node next of old last
        ultima.proxima = atual;
        //será?
        ultima = atual;
        totalElementos++;
```

```
    }


    public void inserirComeço(Pessoa pessoa) {
        // Pointer points to last Node
        Celula ultimo = primeira.anterior;

        Celula atual = new Celula(pessoa);
        // setting up previous and next of new node
        atual.proxima = primeira;
        atual.anterior = ultimo;

        // Update next and previous pointers of start
        // and last.
        ultimo.proxima = primeira.anterior = atual;

        // Update start pointer
        primeira = atual;
    }

    public void gerarDez() {
        //gerar 5 pessoas normais
        for (int a = 0; a <5; a++) {
            //Pessoa(int id, int idade, boolean sexo, boolean gestante, b
oolean lactante, boolean neceEspecial)
            Pessoa pessoaNormal = new Pessoa(gerador.nextInt(100), gerado
r.nextInt(60), gerador.nextBoolean(), false, false, false);

            inserirFim(pessoaNormal);
        }
        Pessoa pessoaEspecial = new Pessoa((gerador.nextInt(100) + 100),
gerador.nextInt(60), gerador.nextBoolean(), false, false, true);
        inserirFim(pessoaEspecial);

        Pessoa pessoaGestante = new Pessoa((gerador.nextInt(100) + 200),
gerador.nextInt(42)+18, true, true, true, false);
        inserirFim(pessoaGestante);

        for(int b = 0; b < 3; b++) {
            Pessoa pessoaIdosa = new Pessoa((gerador.nextInt(100) + 300),
 gerador.nextInt(40) + 60, gerador.nextBoolean(), false, false, false);
            inserirFim(pessoaIdosa);
        }
    }
    public Celula removerInicio() {
        // If list is empty
        if (totalElementos == 0)
            return null;
```

```
        // Check if node is the only node in list
        if (totalElementos == 1) {
            primeira = null;
            totalElementos--;

            return primeira;
        }
        // If list has more than one node,
        // check if it is the first node
        // Move prev_1 to last node
        // Move start ahead
        Celula aux = primeira;
        primeira = primeira.proxima;

        // Adjust the pointers of prev_1 and start node
        ultima.proxima = primeira;
        primeira.anterior = ultima;
        totalElementos--;
        return aux;
    }
    public Celula removerFim() {
        Celula temp = ultima;
        Celula penultimo = ultima.anterior;
        penultimo.proxima = primeira;
        primeira.anterior = penultimo;

        ultima = penultimo;

        return temp;
    }
    public String imprimir() {
        if (this.totalElementos == 0) {
            return "[]";
        } else {
            StringBuilder builder = new StringBuilder("[");
            Celula atual = this.primeira;
            while(atual.proxima !=primeira) {
                builder.append(atual.getElemento());
                builder.append(", ");
                atual = atual.getProxima();
            }
            builder.append(atual.getElemento());
            builder.append("]");
            return builder.toString();
        }
    }
}
```