```
public class FilaDePrioridade {

	static NodoDaFila first, last;
	static int numNodos = 0;

	//Construtor
	public FilaDePrioridade (HufPar[] entrada) {
		for (HufPar x : entrada) {
			this.insere( new NodoBinario (x) );
		}
	}
	/** Insere um NodoBinario na lista */
	void insere (NodoBinario n) {
		if (this.numNodos == 0)this.first = this.last = new NodoDaFila (n);
		else {
			NodoDaFila insere = new NodoDaFila (n);
			insere.prev = this.last;
			this.last.next = insere;
			this.last = insere;
		}
		this.numNodos++;
	}
	/** Remove um NodoBinario da lista */
	void remove (NodoDaFila n) {
		//Caso seja o ultimo
		if ( (n == this.first) && (n == this.last) ) {
			n.next = n.prev = null;
			this.first = this.last = null;
		} else {

			if (this.first == n) {
				n.next.prev = null;
				this.first = n.next;
				n.next = null;
			} else if (this.last == n) {
				n.prev.next = null;
				this.last = n.prev;
				n.prev = null;
			} else {
				n.prev.next = n.next;
				n.next.prev = n.prev;
				n.next = n.prev = null;
			}
		}
		this.numNodos--;
	}
	/** Extrai o nodo binario que guarda o menor valor */
	NodoBinario extraiMenor() {

		if ( this.numNodos < 1) {
			return null;
		}
		else {
			NodoDaFila retira = this.first;

			if (this.numNodos > 1) {

				double freqAtual = retira.info.getHufPar().obtemFreq();

				NodoDaFila atual = this.first.next;

				for ( int i=1; i<this.numNodos; i++ ) {

					if ( atual.info.getHufPar().obtemFreq() <
freqAtual ) {
						retira = atual;
```

```
                                                freqAtual =
retira.info.getHufPar().obtemFreq();
                                        }
                                        atual = atual.next;
                                }
                        } else if ( this.numNodos == 1 ) {

                                //Retira arvore inteira
                                retira = this.first;
                        }

                        this.remove(retira);
                        return retira.info;
                }
        }
        /** Imprime os elementos da fila */
        void printFila() {
                NodoDaFila atual = this.first;
                if (this.numNodos > 0) {
                        System.out.println( this.numNodos+"\n");
                        while (true) {
                                atual.info.printNodo();
                                if (atual != this.last) atual=atual.next;
                                else break;
                        }
                }
        }
}
class NodoDaFila {

                NodoBinario info;
                NodoDaFila prev, next;

                NodoDaFila (NodoBinario x) { info = x; prev = next = null; }


                /** Retorna o NodoBinario dentro do NodoDaLista */
                public NodoBinario getNodoBin() {
                        return this.info;
                }
}
```