Clase Auxiliar

Lun 4/Jun/07

Ordenamiento y Estructuras

Recordando

- QuickSort
  - O(n logn)
  - Se elije un pivote y se ordena, en función a el.
- MergeSort
  - O(n log n)
  - Divide al arreglo en dos partes, las ordena y las une
- Pila
  - Contenedor de datos
  - Primero en entrar, Ultimo en salir
  - **push,pop,reset,isEmpty,isFull**
- Cola
  - Contenedor de datos
  - Primero en entrar, Primero en salir
  - **enQueue,deQueue,reset,isEmpty,isFull**

Ejercicio 1

La clase Cola tiene definidas las siguientes operaciones:

- c=new Cola(), Constructor que inicializa la cola como vacía (sin elementos)
- c.poner(x), Agregar String x al final de la cola c.
- c.sacar(), Extraer (y devolver) primer String de la cola c.
- c.vacia(), Devolver true si cola C está vacía (o false si no)

Utilice (sin escribirla) la clase Cola en un programa que simule la operación del casino de alumnos de la Escuela. Al respecto, el archivo "casino.txt" contiene líneas que registraron la operación del casino en un día normal en la forma indicada en el siguiente ejemplo:

L1200 llegó un alumno a las 12:00 y se puso en la cola
L1201 llegó un alumno a las 12:01 y se puso en la cola
A1202 se atendió al primer alumno de la cola a las 12:02
L1204 llegó un alumno y se puso en la cola
A1205 se atendió al primer alumno de la cola
...

Ejercicio 1 (cont.)

El programa debe leer el archivo y producir los siguientes resultados:
* No total de alumnos atendidos
* Largo máximo de la cola (expresado en # de alumnos)
* Tiempo de espera promedio, es decir, el promedio de minutos que esperaron los alumnos
* Tiempo máximo de espera, es decir, la cantidad de minutos que más tuvo que esperar un alumno.

Para fines prácticos de la clase, se considera que existe la funcion:

int restaTiempo(int t1,int t2), que devuelve el tiempo diferencia entre t1 y t2 en minutos.

Pregunta 3 del control 3-2001

Solución 1

```
public static void main(String[] args) throws IOException{

    Cola cola = new Cola();
    BufferedReader in = new BufferedReader(new
FileReader("casino.txt"))
    String linea = in.readLine();
    int cant = 0;      int adentro = 0;          int maximo = 0;
    int minutos = 0;       int maximoEspera = 0;

    while( linea != null ){

        if( linea.startsWith("L") ){
           cola.poner( linea.substring(1) );
           adentro = adentro + 1;
           if( adentro > maximo )
             maximo = adentro;
        }
     ...
```

Solución 1 (cont.)

```
        else{

            cant = cant + 1;
            adentro = adentro - 1;

            int entrada = Integer.parseInt(cola.sacar());
            int salida = Integer.parseInt(linea.substring(1));
            int delta = restaTiempo(entrada, salida);
            minutos = minutos + delta;
            if( maximoEspera < delta )    maximoEspera = delta;
        }
        linea = in.readLine();
    }

    U.println("Alumnos atendidos: "+cant);
    U.println("Largo máximo cola: "+maximo);
    U.println("Tiempo espera promedio: "+(minutos/cant));
    U.println("Tiempo máximo espera: "+maximoEspera);
}
```

Ejercicio 2

Pregunta 1, control 3 del 2004.

Escribir un programa que reciba un archivo que contiene una lista de palabras ordenadas alfabéticamente y que las regrabe, en el mismo archivo, pero ordenadas por tamaño, es decir, primero la más corta y al final la más larga.

- Cada palabra está grabada en una línea del archivo "palabras.txt" (sin espacios a la derecha).
- El archivo contiene un máximo de 1000 palabras
- Las palabras del mismo tamaño deben mantener entre sí el orden alfabético

Solución 2

```
int N=1000, n=0;
String[]palabra = new String[N];
BufferedReader a=new BufferedReader(new FileReader("palabras.txt"));
String linea;
while((linea=a.readLine())!=null){
  int l = linea.length();
  String largo = (l<10 ? "0" : "") + l;
  palabra[n] = largo+linea;
  ++n;
}

ordenar(palabra,n);
PrintWriter b=new PrintWriter(new FileWriter("palabras.txt"));//0.2
for(int i=0; i<n; ++i)
  b.println(palabra[i].substring(2));
b.close();
```

Solución 2 (cont)

```
static public void ordenar(String[]x,int n){
  if(n<2) return;
  intercambiar(x,n-1,indiceMayor(x,n));
  ordenar(x,n-1);
}
static public void intercambiar(String[]x,int i,int j){
  String aux=x[i]; x[i]=x[j]; x[j]=aux;
}
static public int indiceMayor(String[]x,int n){
  int im=0;
  for(int i=1; i<n; ++i)
    if(x[i].compareTo(x[im])<0) im=i;
  return im;
}
```

Ejercicio 3, con Nota

Pregunta 1, control 3 del 2005.

Escriba un programa que determine los nombres y los puntajes acumulados finales de los atletas que obtuvieron las medallas de oro, plata y bronce en el Decatlon del ultimo campeonato mundial de atletismo, donde un grupo de 24 atletas participaron en 10 competencias distintas

Los resultados de cada una de las 10 pruebas están grabadas en archivos de texto de nombres p1,p2,...,p10 respectivamente. Cada archivo contiene el resultado final de una competencia: los 24 atletas ordenados descendentemente por el puntaje obtenido en la prueba. Cada linea del archivo contiene nombre del atleta(20 caracteres) y el puntaje(numero entero de 4 dígitos)

FIN

Profesor: Andrés Muñoz

Auxiliares: Oscar Alvarez
Pedro Valencia