Guía de Estudio

Control 1

Profesor: Andrés Muñoz
Auxiliares: Agustín Almonte
Marcelo Muñoz
Patricio Salles

CC10A - 2004

Tabla de Contenidos

1. Promedio de Notas (Pregunta 2 Control 1, 1994)

Unos profesores usan la siguiente política en sus ramos: "*La nota final se calcula promediando todas las notas, pero reemplazando la peor por la mejor*" (o sea se elimina la peor y la mejor vale por dos).

Escriba un programa que lea una lista de notas reales (terminadas por un 0), e imprima el promedio según esta política.

Solución:

Comenzamos declarando las variables necesarias de nuestro programa. Las variables "mejor y "peor" son inicializadas de manera que cualquier nota ingresada y sea mas baja que la peor y superior a la mejor inicial, de esta manera nos aseguramos de que cambien a valores reales. EN el caso de ser inicializadas en 0 debemos preocuparnos que tomen el valor de la primera nota como mejor y peor para después comenzar la comparación con el resto.
La variable notas será nuestra variable receptora de las notas ingresadas, suma será la variable acumuladora y cont el contador de la cantidad de notas.

```
class Peor {
	static public void main (String args[]) {
		// Declaraciones necesarias
		Console c = new Console();

		double mejor = 1;
		double peor = 7;
		double notas = 0;
		double suma = 0;
		int cont = 0;
```

El siguiente paso es generar el ciclo que solicitará las notas al usuario. Dentro de este ciclo debemos procuparnos por sumar la nota recibida a nuestra variable acumuladora y de actualizar los valores para la mejor y peor nota.

```
		c.println("Ingrese las notas");
		c.print("?");
		while((notas = c.readDouble())!=0) {
			++cont;
			if (peor>notas)
				peor = notas;
			if (mejor<notas)
				mejor = notas;
			suma+=notas;
			c.print("?");
		}
```

Por ultimo, debemos restar a nuestra suma la peor nota y sumar la mejor en su lugar para desplegar el promedio resultante.

```
            suma = suma - peor + mejor;

            // Imprime el promedio
            c.println("El promedio es "+suma/cont);
      }
}
```

2. Separador de Palabras (Pregunta 3 Control 2, 1994)

Se pide leer de la pantalla 2 strings: uno con una lista de caracteres considerados *Separadores* y otro que es una línea de texto cualquiera.

El programa debe imprimir las palabras contenidas en la línea de texto, sin los separadores. Finalmente, debe imprimir el número de palabras que hay en esta.

Una secuencia de separadores es lo mismo que tener uno solo.

Ejemplo:

```
Separadores? ,. ;
      Linea? ,hola... como te va?
Palabras:
       hola
       como
       te
       va?
(4 palabras)
```

Solución:

Este es un clasico problema de manejo de strings. Lo primero que necesitamos hacer es obtener los parametros que utilizaremos. Las variables seran dos strings uno para separadores y otro para la linea de texto, un contador para contar las palabras separadas y otras tres variables para manejar la separación del string cuyon uso explicaremos durante el codigo.

```
class Tokenizador {
      static public void main (String args[]) {
            Console c = new Console();
            String seps;
            String linea;
            int contador=0, indice=0;
            boolean sep = true;
```

Obtenemos la información desde el usuario.

```
            c.print("Separadores? ");
            seps = c.readLine();
            c.print("Linea? ");
            linea = c.readLine();
```

Ahora creamos una variable en que guardaremos los caracteres que no son separadores a medida que recorremos la linea de texto.
La variable indice nos servira como memoria para recordar en que carácter estamos parados y tambien en la condicion de termino del ciclo.

```
            String palabra="";
```

```
while (indice <= linea.length()) {
```

dentro del ciclo que recorre los caracteres de la linea de texto tenemos dos casos, uno que el carácter en "indice" sea un separador o que sea un carácter permitido. Luego la base del ciclo es un "if y else" que separan estos casos. Cuando encontramos un separador debemos imprimir la palabra aumentar el contador y vaciar la variable de acumulación de caracteres "palabra". Por otro lado si el carácter que encontramos no es un separador debemos agregarlo a "palabra".
La mayor dificultad de este problema es el caso en que tenemos separadores juntos, para esto usamos la variable de tipo boolean "sep" que nos indicará si el ultimo carácter visitado fue un separador o no. Entonces ante el encuentro de un separador verificamos si el carácter anterior fue o no un separador antes de imprimir y aumentar el contador, si no fue asi realizamos todas las tareas mencionadas y seteamos la variable "sep" a true (verdadero) para saber en la proxima iteración que vimos un separador.
En caso de no encontrar un separador lo primero es avisar que ya no estamos sobre separadores y luego agregar el carácter.
Y la ultima instrucción del ciclo, lo que nunca debemos olvidar, aumentar el contador.

```
                if (seps.indexOf(linea.charAt(indice))>=0) {
                //si el ultimo carácter no fue un separador.
                        if (!sep){
                                c.println("     " + palabra);
                                contador++;
                                sep=true;
                                palabra = "";
                        }
                }
                else {
                        sep=false;
                        palabra = palabra + linea.charAt(indice);
                }
                indice++;
        }//while
```

Como en nuestro ciclo imprimimos la variables palabra cada vez que encontramos una secuencia de separadores, si la linea de texto ingresada no termina con un separador la varible "palabra" quedara con caracteres que no han sido impresos ni contados como una palabra mas.
Para resolver esto cosultamos el largo de la variable que nos indicara si tiene algun contenido en cuyo caso imprimimos y contamos.

```
        if(palabra.length()>0){
                c.println("     " + palabra);
                contador++;
        }
```

Y terminamos imprimiendo el contador.

```
        c.println("("+contador+" palabras)");
    }
}
```

3. Niveles de Contaminación (Pregunta 1 Control 1, 1997)

En la entrada de Avda. Tupper del Parque O'Higgins se ubica la estación que presenta siempre los niveles más altos de contaminación en Santiago. Al respecto, escriba un programa que permita realizar las siguientes operaciones con la información del índice de contaminación:

Oper.	Significado
1	Solicitar y guardar el valor del índice y la hora en que fue medido
2	Mostrar el último valor del índice y la hora en que se registró
3	Mostrar el valor máximo del índice hasta ahora medido y la hora en la cual se produjo
4	Mostrar el promedio de los valores obtenidos para el índice
0	Fin de las operaciones

Notas:

En el caso que el índice alcance el valor 400 escriba el mensaje "Emergencia". Posteriormente, la primera vez que el índice registre un valor menor que 400 escriba "Fin Emergencia".
El siguiente diálogo muestra en ejemplo de las operaciones:

```
Operacion? 1
Indice y Hora? 200 1030

Operacion? 1
Indice y Hora? 420 1115
Emergencia

Operacion? 1
Indice y Hora? 280 1210
Fin Emergencia

Operacion? 2
Indice=280   Hora=1210

Operacion? 4
Promedio Indice=300

Operacion? 3
Maximo Indice=420   Hora=1115

Operacion? 0
Fin Operaciones
```

Solución:

Para este problema necesitaremos variables para guardar los valores que requieren cada una de las operaciones, debemos guardar :

- ultimo indice con su hora de ingreso.
- índice máximo con su hora de ingreso (inicializados en 0 para asegurarnos de que cambiaran frente al primer ingreso).
- la suma de los indices ingresados y la cantidad (Para calcular el promedio).

Además utilizaremos una variable oper que recibira la operación deseada y una variable de tipo boolean emergencia que indicará si estamos en estado de emergencia o no.

```
class Estacion {
      static public void main (String args[]) {
            // Declaraciones
            Console c = new Console();
            int oper = 10;  // Solo como valor inicial

            int ind_max=0, hor_max=0;
            int ind_ult=0, hor_ult=0;
            int ind_sum=0, tot_ind=0;
            boolean emergencia=false;
```

Ahora creamos el ciclo de operaciones que continuara mientras se ingrese una operación valida. Al inicio del ciclo consultamos la operación deseada.

```
            while (oper>0) {
                  c.print("Operacion? ");
                  oper = c.readInt();
```

Para distinguir la operación a realizar utilizaremos un bloque switch que compara su parámetro contra distintos casos, en caso de igualdad ejecuta las instrucciones correspondientes a ese caso y luego sigue comparando con los demas casos a menos de recibir la instrucción break. Existe un caso por defecto (default) que se ejecuta en cualquier caso que se llegue a consultar, es como un "else". El uso de este bloque switch puede ser fácilmente reemplazado por instrucciones "if" seguidos de "else if" y finalizado por un "else" en que se comnparen los valores de oper contra los de los casos de "switch".
Entonces comenzamos el switch y como primer caso tenemos oper==0, en este caso debemos terminar el programa lo que hacemos quebrando el "switch" y el ciclo terminará por la condición del "while".

```
                  switch (oper) {
                  case 0:
                        break;
```

El segundo caso es oper==1. En este caso se debe solicitar el indice y la hora a ingresar, actualizando con estos valores las variables de ultimo indice y hora, indice máximo y hora, suma de indices y cantidad. Otra operación que debemos hacer con los datos obtenidos es verificar el estado de emergencia y notificarlo, para esto comparamos el indice recibido con el limite (400) para entrar en emergencia en caso de emergencia==false o salir de emergencia en caso de emergencia=true.
Para finalizar este caso quebramos el bloque "switch" para que los siguientes casos no sean considerados.

```
                  case 1:
                        c.print("Indice y Hora? ");
                        ind_ult = c.readInt();
                        hor_ult = c.readInt();

                        // Vemos si hay maximo
```

```
                        if (ind_max<ind_ult) {
                              ind_max=ind_ult;
                              hor_max=hor_ult;
                        }

                        // Sumamos a los totales
                        ind_sum += ind_ult;
                        ++tot_ind;

                        // Vemos si hay emergencia
                        if (ind_ult>=400 && !emergencia) {
                              c.println("Emergencia");
                              emergencia=true;
                        }
                        else if (ind_ult<400 && emergencia) {
                              c.println("Fin Emergencia");
                              emergencia=false;
                        }
                        break;
```

El tercer caso es oper==2. En esta operación se nos solicita el valor de el ultimo indice ingresado y su hora, valores que gracias a nuestra definición de variables tenemos a la mano y solo debemos imprimirlas y quebrar el bloque para no considerar los casos posteriores.

```
                  case 2:
                        c.print("Indice="+ind_ult+
                                " Hora="+hor_ult);
                        break;
```

El cuarto caso es oper==3. En este caso debemos imprimir el indice máximo y su hora de ingreso y quebrar.

```
                  case 3:
                        c.print("Indice="+ind_max+
                                " Hora="+hor_max);
                        break;
```

El quinto y ultimo caso a considerar es oper==4. Ahora debemos desplegar el indice promedio de los registros y quebrar.

```
                  case 4:
                        c.print("Indice="+ind_sum/tot_ind);
                        break;
```

Como ultimo caso del bloque "switch" tenemos el caso por defecto al que llegaremos en caso de ingresar una operación que no corresponde a ninguna anterior y notificaremos al usuario de su error. En el último caso no es necesario quebrar porque no hay nada después.

```
                  default:
                        c.println ("Error de Operacion");
                  }//cerramos switch
            }//cerramos while
```

Y en caso de salir del ciclo while despedimos al usuario con el mensaje "Fin Operaciones" y cerramos el programa.

```
            c.println("Fin Operaciones");
      }
}
```

4. Distancia 2 Puntos (Problema 1 Control 2, 1998)

a) La distancia entre dos puntos de coordenadas cartesianas (x1,y1) y (x2,y2) se define como:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Escriba un método llamado distancia que reciba las coordenadas (reales) de 2 puntos y devuelva su distancia.

b) El algoritmo de Montecarlo permite obtener una aproximación de PI de la siguiente manera:

- Considerar un circulo inscrito en un cuadrado de lado 2
- Generar al azar varios puntos (x,y) dentro del cuadrado (usando Math.random())
- Contar el número de puntos que caen dentro del círculo (cuyo centro está en (1,1))

Aproximar PI considerando la proporción de entre los puntos que "cayeron" dentro del círculo y el total de puntos, y la proporción entre las áreas del círculo y el cuadrado.

Solución

a)

Para esta parte del problema debemos declarar un metodo de la manera vista en clases:

[tipo metodo] [tipo retorno] [nombre metodo] ([tipo parámetro] [parámetro], ...) {
...
}

En este caso el tipo del metodo sera publico "public", el tipo de retorno "double", el nombre "distancia" y recibirá de parámetros cuatro coordenadas de tipo "double".

```
public double distancia (double x1, double y1, double x2, double y2)
{
```

En el cuerpo del metodo debemos aplicar la fórmula para distancia, mostrada en el enunciado, por medio de las funciones de la libreria matemática y retornar su valor.

```
      double d;
      d = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
      return d;
}
```

b)

Creamos las variables necesarias: la consola, la variable pi en que guardaremos la estimacion, el contador de puntos dentro del círculo y la cantidad de puntos a utilizar en la estimación.

```
Console c = new Console();
```

```
double pi = 0;
double ndentro = 0, npuntos = 1000;
```

A continuación debemos generar los puntos al azar dentro del cuadro de lado 2, esto nos deja un rango [0,2[para valores de x e y. Los puntos son generados dentro de un ciclo "for" para "npuntos" iteraciones.
Dentro del ciclo debemos contar los puntos que caen dentro del circulo con centro en (1,1), para esto calculamos la distancia entre el centro y el punto generado al azar. Si la distancia es menor al radio del circulo el punto esta dentro y sumamos el contador de puntos.

```
for (int i=0; i<npuntos; i++) {
      double x = Math.random() * 2;
      double y = Math.random() * 2;
      npuntos++;
      if (distancia(1, 1, x, y) <= 1) { // DENTRO
            ndentro++;
      }
}
```

Por último solo nos queda calcular la estimación y desplegar su resultado.

```
pi = ndentro / npuntos;
c.println("PI = " + pi);
```

5. Declaración de Impuestos (Pregunta 1 Control 1, 2000)

Escriba un programa que calcule el monto de impuestos que debe pagar una persona, ciñéndose al diálogo que se muestra en el siguiente ejemplo:

```
Mes 1
Ingresos Percibidos en $ ? 300000
Factor de Actualizacion ? 1.020
Ingresos Actualizados = $ 306000

...

Mes 12
Ingresos Percibidos en $ ? 250000
Factor de Actualizacion ? 1.000
Ingresos Actualizados = $ 250000

Suma de ingresos actualizados = $ xxxxxxxxxx
Impuestos a pagar = $ xxxxxx
```

Nota:
- Los ingresos actualizados de cada mes se obtienen aplicando el factor a los ingresos percibidos.
- Para calcular los impuestos a pagar deben aplicar las siguientes reglas (UTM = $15000)

- Si Total < 10 UTM => Exento de Impuestos
- Si Total >= 10 UTM y < 30 UTM => 5% de Impuestos
- Si Total >= 30 UTM y < 50 UTM => 10% de Impuestos
- Si Total >= 50 UTM => 15% de Impuestos

Solución

Comenzamos con unn ciclo "for" para capturar los datos de los 12 meses. Teniendo los valores para ingreso y factor de cada mes, calculamos el ingreso actualizado (ingreso * factor) y lo sumamos a los ingresos totales.

```
class Impuestos {
      public static void main (String args[]) {
            Console c = new Console() ;
            int suma = 0 ;
            for ( int i = 1 ; i <= 12 ; ++i ) {
                  c.println("Mes "+i) ;
                  c.print("Ingresos percibidos en $ ? ") ;
                  int ingresos = c.readInt() ;
                  c.print("Factor de actualizacion ? ") ;
                  double factor = c.readDouble() ;
                  suma = suma + (int) (ingresos * factor) ;
              }
```

Ahora que debemos calcular los impuestos a pagar usamos un valor de 15000 para la utm y calculamos el porcentaje de los ingresos totales a pagar de impuesto usando una secuencia de comparaciones de acuerdo a las reglas dispuestas en el enunciado.

```
            int impuestos = 0 ;
            int utm = 15000 ;
```

```
            if ( suma < 10 * utm )
                  impuestos = 0 ;
            else if ( suma < 30 * utm )
                  impuestos = (int)(suma * 0.05) ;
            else if ( suma < 50 * utm )
                  impuestos = (int)(suma * 0.10) ;
            else
                  impuestos = (int)(suma * 0.15) ;
```

Por último informamos el resultado de los calculos y terminamos el programa.

```
            c.println("Suma de ingresos actualizados = $ "+suma) ;
            c.println("Impuestos a pagar = $ "+impuestos);
      }
}
```

6. Subrayado de Palabras (Pregunta 2 Control 1, 2000)

Escribir un programa que lea una linea y que le ponga un subrayado debajo de la palabra que comienza con un &, es decir:

```
Ingrese frase:
? Se que me ira &bien en el control de &Computacion
Frase subrayada:
> Se que me ira bien en el control de Computacion
                ----                  -----------
```

Nota: La palabra subrayada comienza con un & y termina con un ESPACIO. Los & no aparecen en el resultado.

Solución

Lo primero es lo primero, leemos la linea ingresada por el usuario.

```
class Subrayado {
      public static void main (String args[]) {
            Console c = new Console() ;
            c.print("? ") ;
            String linea = c.readLine() ;
```

Para procesar el texto ingresado utilizaremos tres variables:un string con el texto resultante, un string con el subrayado y una variable boolean que indicara el estado en que estamos (subrayando o no).

```
            boolean sub = false ;
            String linea1 = "", linea2 = "" ;
```

Una de la formas que tenemos para lograr el resultado buscado es recorrer el string de entrada letra a letra. Al usar esta forma tendremos dos caracteres que nos indicaran la acción a realizar, el carácter "&" nos indicara que debemos comenzar el subrayado y " " (espacio) que debemos detenerlo.
Comenzamos el procesamiento entrando a un ciclo que recorre todas las letras del texto y capturando cada una de ellas.

```
            for ( int i = 0 ; i < linea.length() ; ++i ) {
                  char d = linea.charAt(i) ;
```

Como ya dijimos tenemos dos acciones basicas, pero para saber cual de ellas esperamos realizar es que utilizamos la variable de tipo boolean sub. Cuando "sub" sea true, estamos subrayando, debemos esperar por " " para detener el subrayado y cuando sub sea false, no estamos subrayando, debemos esperar por un "&" para comenzar a subrayar.
Veamos el caso en que "sub" es true. Tomamos cada carácter, si no es un espacio lo agregamos a "linea1" y agregamos el "-" correspondiente a ese carácter en "linea2". Cuando sub es true y encontramos un espacio debemos detener el subrayado, copiamos el carácter en "linea1" y en "linea2" y cambiamos el valor de sub a false para indicar que el subrayado esta detenido.

```
                    if ( sub ) {
                          if ( d != ' ' ) {
                                linea1 += d ;
                                linea2 += "-" ;
                            }
                          else {
                                linea1 += d ;
                                linea2 += " " ;
                                sub = false ;
                            }
                      }
```

Ahora si sub es false, debemos esperar el carácter "&" para comenzar el subrayado. Mientras el carácter que vemos no es "&" debemos agregar espacios en "linea2" y al mismo carácter en "linea1". Al momento de encontrar el carácter "&" solo debemos cambiar el valor de "sub" a true porque este carácter no estará incluido en el texto de salida por lo que sera mas corto.

```
                    else {
                          if ( d != '&' ) {
                                linea1 += d ;
                                linea2 += " " ;
                            }
                          else {
                                sub = true ;
                            }
                      }
                }//while
```

Y terminamos imprimiendo las lineas resultado.

```
            c.println("> "+linea1) ;
            c.println("  "+linea2) ;
        }
    }
```

7. Competitividad de Paises (Pregunta 3 Control 1, 2000)

Recientemente se publicó un informe de competitividad mundial de los paises en que Chile aparece en el lugar 26 (año 2000, pero ahora aparece 24... GUAU). Para determinar el lugar que ocupa un país se calcula un puntaje de acuerdo a la evaluacion (buena, regular, mala) en 50 aspectos distintos (ej: gasto en salud, educacion, etc).

Al respecto escriba un programa que escriba los nombres de los países que ocuparon el primer y segundo lugar en el ranking. Los dados se deben obtener del archivo "paises.txt", que en cada línea contiene la siguiente información de un país:

- Nombre del país (20 primeros caracteres)
- Evaluación en cada uno de los 50 aspectos (como B, R o M).

Por ejemplo:

```
Australia            BRBMMMRRBMRRMBBRMMBRRMMMBBBBRMBRMMBRMBRBMRMBBMRMBR
Alemania             BMRMMBBBBBBBMBMBMBMRRRRRRBMMRMBRMBMRBMRMBMRMBRMBMR
...
```

Nota: Para calcular el puntaje de un país se deben sumar las evaluaciones en cada aspecto. Para ello considere que (B)ueno, (R)egular y (M)alo equivale a valores 3, 2 y 1 respectivamente.

Solución

Para aplicar esta solución, cabe explicar que, leer una línea desde un archivo es muy similar a leerla desde el teclado. Para ello utilizas la clase **BufferedReader** y la clase **FileReader** de la siguiente forma:

1. Abres un archivo para leer su contenido:

```
BufferedReader file = new BufferedReader(
       new FileReader("<nombre del archivo>")) ;
```

2. Lees una línea desde el archivo:

```
String linea = file.readLine();
```

Con estas 2 líneas, quedamos con la siguiente solución:

Necesitamos tener variables para recordar cuales son los paises que tienen el primer y segundo lugar.
Creamos las variables necesarias, consola, lectro de archivo, string "linea" para guardar la linea leida, strings "pais1" y "pais2" con los dos primeros paises y las dos variables int nota1 y 2 para el puntaje correspondiente a los primeros paises.

```
import java.io.* ;
class Ranking {
       public static void main (String args[]) throws IOException {
              Console c = new Console() ;
```

```
BufferedReader file = new BufferedReader(
      new FileReader("paises.txt")) ;
String linea ;
String pais1 = "", pais2 = "" ;
int nota1 = 0, nota2 = 0 ;
```

Ahora comenzamos a leer el archivo, la linea que estamos leyendo queda almacenada en la variable "linea" que debemos separar en los primeros 20 caracteres, que contienen el nombre del pais, y el resto con su evaluación.
La condición de ruptura del ciclo de lectura es llegar al fin del archivo en donde "linea" tomara un valor null (nulo).

```
while ( (linea = file.readLine()) != null ) {
      String pais = linea.substring(0,20) ;
      String eval = linea.substring(20) ;
```

Creamos una variable interna al ciclo en al que almacenaremos la nota del pais que estamos leyendo.
Para obtener la evaluación debemos recorrer carácter por carácter el string que contiene la votación y sumar la puntuación correspondiente a ella. Esto lo hacemos con un ciclo "for" con condición "mientras el contador sea menor al largo de la evaluación". Entonces tomamos un carácter de la linea de evaluación y sumamos su puntaje a la nota acumulada del pais.

```
int nota = 0 ;
for ( int i = 0 ; i < eval.length() ; ++i ) {
      char d = eval.charAt(i) ;
      if (d == 'B')
            nota += 3 ;
      else if (d == 'R')
            nota += 2 ;
      else
            nota += 1 ;
  }
```

El siguiente paso es actualizar el primer y segundo lugar entre los paises. Debemos comparar el pais que acabamos de leer con los paises que ocupan el primer y segundo lugar por la eventualidad que el nuevo pais tenga un puntaje mayor a alguno de ellos.
En el caso extremo el nuevo pais tomará el primer lugar si su puntaje es mayor al actual primer pais, si esto pasa el pais actualmente primero tomará el segundo lugar y el nuevo pais el primero. Recordemos que solo nos interesan las dos primeras posiciones asi que nos olvidamos del pais que estaba antes en la segunda posición.

```
if ( nota > nota1 )  {
      nota2 = nota1 ;
      pais2 = pais1 ;
      nota1 = nota ;
      pais1 = pais ;
  }
```

El otro caso es que el puntaje del nuevo pais no sea mayor que el del actual primero pero si mayor que el del actual segundo. En este caso solo cambia el pais en segundo lugar siendo reemplazado por el nuevo pais.

```
                    else if ( nota > nota2 ) {
                          nota2 = nota ;
                          pais2 = pais ;
                       }
                }//while
```

No debemos olvidar que al cambiar de lugar a un pais tambien debemos cambiar su puntaje relacionado.
Por último solo nos queda informar el resultado de la evaluación.

```
            c.println("Primer lugar : "+pais1) ;
            c.println("Segundo lugar: "+pais2) ;
        }
    }
```

8. Eliminador de Strings (Pregunta 2 Control 1, 2001)

a) Escribir un método (función) de encabezamiento

```
static public String borrar(String x, String y)
```

que elimine todas las apariciones de x en y. Por ejemplo, la invocación borrar("bra","abracadabra") entrega el string "acada"

b) Utilice el método anterior en un programa que lea todas las líneas de un archivo y las grabe en otro archivo eliminando todas las apariciones de los strings ":-)" y ":-(". El programa debe obtener el nombre del archivo de entrada siguiendo el diálogo indicado en el siguiente ejemplo:

```
Nombre de archivo de entrada ? carta
El resultado quedará en el archivo carta.out
```

Solución

a)

Creamos el metodo con el encabezado indicado y las variables : "salida" en donde guardaremos el string resultante de la eliminación, "ini" índice hasta el que tenemos revisada la palabra y "idx" que utilizaremos para capturar el indice del substring a borrar dentro de la linea.

```
static public String borrar (String x, String y) {
      String salida = "";
      int ini = 0, idx = 0;
```

Ahora creamos un ciclo infinito que solo quebrará en el caso que no encontremos el string que debemos borrar dentro de la linea. Para obtener el indice a partir del cual debemos borrar utilizamos la función "indexOf(String a, int i)" de la clase string en que "a" es el string a buscar y "i" es el indice a partir del cual se debe buscar "a".
Si no encontramos el patron dentro de la linea quebramos el ciclo, en caso contrario agregamos a la variable "salida" el substring desde donde comenzamos a buscar hasta el indice de ocurrencia del patrón, y nos saltamos el largo del patrón para seguir la búsqueda.

```
      while (true) {
            idx = y.indexOf(x, ini);
            if (idx < 0) break;
            salida = salida + y.substring(ini, idx);
            ini = idx + x.lenght();
      }
```

Terminamos agregando el substring en que no se encontro el patrón y retornando la variable "salida".

```
      salida = salida + y.substring(ini);
      return salida;
}
```

b)

Leemos el nombre del archivo de entrada y de acuerdo a este creamos las variables para leerlo y escribir el archivo de salida.

```
Console c = new Console();
c.print("Nombre del archivo de entrada?");
String inFile = c.readLine();
c.println("El resultado quedará en " + inFile + ".out");
BufferedReader in = new BufferedReader(new FileReader(inFile));
PrintWriter out = new PrintWriter(new FileWriter(inFile + ".out"));
```

Utilizando un ciclo leemos linea a linea el archivo de entrada aplicando la funcion borrar para quitar los patrones indicados,una vez aplicada la función utilizamos el escritor para poner la linea modificada en el archivo de salida.

```
while(true) {
      String line = in.readLine();
      if (line == null) break;
      line = borrar(":-)", line);
      line = borrar(":-(", line);
      out.println(line);
}
```

Y lo que nunca debemos olvidar, cerrar el lector y escritor.

```
in.close();
out.close();
```

9. Calendario (Pregunta 3 Control 1, 2002)

La siguiente tabla muestra los métodos de una clase (Dia) que permite realizar operaciones con los días de una semana:

Encabezamiento	Ejemplo	Resultado
`Dia(int x) // 1 ≤ x ≤ 7`	`Dia a = new Dia(1);`	Objeto que representa el día lunes
`Dia mañana()`	`a.mañana()`	Objeto que representa el día de mañana
`int comparar(Dia x)`	`a.comparar(b)`	1 si los objetos son iguales (0 si distintos)
`String nombre()`	`a.nombre()`	"lunes" o "martes" o ... "domingo"

a) Utilice la clase anterior en un programa que muestre el calendario de un mes siguiendo el diálogo:

```
Ingrese cantidad de días del mes (1-31) : 31
Ingrese primer dia del mes (1-7) : 3
 mi   ju   vi   sa   DO   lu   ma   mi   ju   vi   sa   DO   ...  ju   vi
 1    2    3    4    5    6    7    8    9    10   11   12   ...  30   31
```

Nótese que:

- Los nombres de los días aparecen abreviados y separados por un espacio
- Los domingos se muestran en mayúsculas
- Los números de los días aparecen bajo la primera letra del nombre del día

b) Escriba los métodos mañana y nombre, suponiendo la siguiente declaración de la clase:

```
class Dia{
   private int d; // Representacion
   ...
}
```

Solución

a)

Recibimos la informacion desde el usuario

```
Console c = new Console();
c.print("Ingrese cantidad de dias del mes (1-31) : ");
int dias = c.readInt();
c.print("Primer dia del mes (1-7) : ");
int primer = c.readIne();
```

Ahora creamos dos variables de la clase "Dia" , una para recorrer los dias del mes y otra que utilizaremos como comparación para saber si el dia en que estamos es domingo.

```
Dia d = new Dia(primer);
Dia DOM = new Dia(7);
```

Ahora usamos un ciclo, por la cantidad de dias del mes, para imprimir los nombres de los dias. El ciclo tiene dos casos relevantes, si el dia es domingo debemos imprimir su nombre con mayusculas y si no solo obtenemos los dos primeros caracters de su nombre e imprimimos.
Notar que hemos usado la función nombre() de la clase dia y a su resultado aplicamos substring para obtener los primeros caracteres del dia.

```
for (int i=0; i<dias; i++) {
      if (d.comparar(DOM) == 0) {
            c.print((d.nombre()).substring(0, 2).toUpperCase());
      }
      else {
            c.print((d.nombre()).substring(0, 2));
      }
      c.print(" ");
```

La linea anterior la utilizamos para separar un nombre de otro y para avanzar de un dia al siguiente usamos la función mañana() de la clase "Dia".

```
      d = d.mañana();
}//for
```

Saltamos de linea y creamos un for en que imprimimos los numeros correspondientess a cada dia prestando cuidado con los numeros de una cifra a los que agregamos un espacio para la alineación con los nombres.

```
c.println();
for (int i=0; i<dias; i++) {
      if (i < 10) {
            c.print(" ");
      }
      c.print(i + " ");
}
c.println();
```

b)

Para la función mañana debemos retornar un objeto de la misma clase pero correspondiente al dia siguiente. Para esto hacemos uso del constructor de la clase para crear un dia nuevo y la variable "d" como referencia para el contructor.

```
public Dia mañana() {
      Dia m = new Dia(this.d+1);
      return m;
}
```

Asi hemos logrado crear un el mañana...

Para la funcion nombre, solo tenemos la información del numero del dia por lo que debemos hacer una asociación con el numero del dia y su nombre. Esto lo podemos hacer con una seguidilla de "if " o utilizando un "switch"

```
public String nombre() {
	switch (this.d) {
		case 1:
			return "lunes";
		case 2:
			return "martes";
		case 3:
			return "miércoles";
		case 4:
			return "jueves";
		case 5:
			return "viernes";
		case 6:
			return "sábado";
		case 7:
			return "domingo";
	}
	return null;
}
```

El ultimo retorno "return null" es necesario para la compilación en caso que ningun caso de switch fuera cierto.

10. Cartas Modelo

Honorato Mayorga tiene varias pololas, y quiere automatizar la correspondencia que mantiene con ellas. En particular, tiene una carta escrita en el archivo "carta.txt" y desea adaptarla a las características de sus amigas. En la carta ha dejado con @ el nombre de pila y con # el color de ojos de cada amiguita.

Este archivo es como sigue:

```
Querida @:

Dejame decirte que me gustas tanto como antes, o
quizas mas que antes. El color # de tus ojos me inspira
mucho, de hecho, # es mi color favorito. Asi, @, por favor
responde mis llamadas. Mandame tu respuesta en un sobre #.

Por siempre tuyo
Honorato.
```

Pese a que Ud. ha desaconsejado a Don Honorato, el le ha pedido el favor especial de escribir un programa que solicite el nombre y el color de ojos de una dama, tome la carta del archivo "carta.txt" y escriba la carta resultante en un archivo de nombre "salida.txt", reemplazando el nombre y color de ojos de la dama donde corresponda. Nota: una linea de la carta puede requerir mas de un reemplazo.

Solución

Solicitamos la información al usuario

```
class CartaModelo {
      static public void main (String[] args) {
            Console c = new Console();

            c.println (“Ingrese el nombre de la dama?”);
            String nombre = c.readLine();
            c.println (“Ingrese su color de ojos?”);
            String color = c.readLine();
```

Creamos el lector y escritor de archivo.

```
            BufferedReader carta = new BufferedReader (
                  new FileReader (“carta.txt”));
            PrintWriter salida = new PrintWriter (
                  new FileWriter (“salida.txr”));
```

Ahora usamos un ciclo que leera linae a linea el archivo. El trabajo dentro del ciclo se divide en dos para cada linea, primero buscaremos ocurrencias de “@” en la linea y las reemplazaremos por el nombre ingresado por el usario, y luego buscaremos “#” par areemplazarlos pro el color ingresado.

```
            String linea;
            while ( (linea = carta.readLine()) != null) {
```

Reemplazamos los "@". Buscamos en la linea hasta no encontrar mas, en cuyo caso la función idexOf retornara –1, y rearmamos la linea cortando el carácter "@" y reemplazandolo por el contenido de "nombre".

```
                  int indice = linea.indexOf("@");
                  while (indice >= 0) {
                        linea = linea.substring(0, indice)+
                              nombre +
                              linea.substring(indice + 1);
                        indice = linea.indexOf("@");
                  }
```

Y ahora hacemos lo mismo con los "#".

```
                  indice = linea.indexOf("#");
                  while (indica >= 0) {
                        linea = linea.substring(0, indice)+
                              color +
                              linea.substring(indice + 1);
                        indice = linea.indexOf("@");
                  }
```

E imprimimos en el archivo de salida y cerramos los archivos.

```
                  salida.println(linea);
            }//while
            salida.close();
            carta.close();
      }
}
```

11. Fechas

La clase Fecha contiene los métodos indicados en la siguiente tabla:

Ejemplo (a y b son Fechas)	Significado	Encabezamiento
a.resta(b)	a – b en años	int resta(Fecha x)
a.comp(b)	0 si a=b, N°<0 si a<b, N°>0 si a>b	int comp(Fecha x)
new Fecha("dd/mm/aaaa")	Objeto con fecha dd/mm/aaaa	Fecha(String x)

(a) Escriba un programa que use la clase para calcular los años transcurridos entre dos fechas, de acuerdo al diálogo indicado en el siguiente ejemplo:

```
Calcular diferencia en años entre dos fechas
Fecha 1 (dd/mm/aaaa) ? 22/01/2001
Fecha 2 (dd/mm/aaaa) ? 24/08/1989
Años transcurridos = 11
```

Nota: Las fechas pueden estar en cualquier orden.

(b) Escribir el método **comp**, suponiendo la siguiente declaración de la clase Fecha:

```
public class Fecha {
      public int d, m, a; //representación: dia, mes y año
      . . .
}
```

(c) Propuesto. Escribir los métodos resta y constructor

Solución

a)
Es claro que la solución a este problema es mucho mas sencilla utilizando la clase Fecha. Solo debemos crear objetos de esta clase a partir de las fechas ingresadas y utilizar la función resta entre ambas para obtener los años transcurridos.
Dado el constructor de la clase Fecha, podemos utilizar la interfaz con el usuario para exigir el ingreso bajo un cierto formato, en este caso "dd/mm/aaa", y facilitarnos la construcción de los objetos de tipo Fecha. Si confiamos en que el usuario seguira bien las instrucciones podemos crear directamente el objeto de la linea leida.

```
      c.print("Fecha 1 (dd/mm/aaaa) ? ");
      Fecha1 f1 = new Fecha(in.readLine());
      c.print("Fecha 2 (dd/mm/aaaa) ? ");
      Fecha1 f1 = new Fecha(in.readLine());
```

Para asegurarnos de dar un resultado coherente tomaremos la precaución de restar la fecha ms antigua a la mas reciente y asi retornar un numero de años positivo.

```
      if( f1.comp(f2) < 0 )
            c.println( "An~os=" + f1.resta(f2) );
      else
            c.println( "An~os=" + f2.resta(f1) );
```

b)
Dad la declaración de la clase Fecha, sus variables de instancia nos permiten hacer comparaciones muy sencillas. Es bueno, prever la facilidad o dificultad que daran las variables de instancia a la definición de los metodos y funciones de una clase y en base a esto decidir el tipo y el numero de variables a incluir.
Para comparar las fechas claramente debemos ir de mayor a menos. Primero fijarnos en la diferencia de años, en caso que no nos sirva para decidir pasar a los meses y luego a los dias. Entonces la definición de la función comp será:

```
public int comp(Fecha x)
{
      //comparar años
      if( a < x.a ) return -1;
      if( a > x.a ) return 1;
      //comparar meses
      if( m < x.m ) return -1;
      if( m > x.m ) return 1;
      //comparar dias
      return d - x.d;
}
```

12. Servicio de Inteligencia

El servicio de inteligencia de Putre, le ha solicitado a usted que programe un traductor del codigo Morse, para lo cual le proporciona la clase **Morse**, que le permite traducir una letra del ya mensionado codigo al alfabeto que usamos.

Encabezado	Descripción	Ejemplo
public Morse()	Constructor	Morse m = new Morse();
public char toAlfabeto(String clave)	Traduce de clave Morse a alfabeto	String b = m.toAlfabeto(".-.")

(a)Usted debe construir la clase Traductor, con su respectivo constructor, y los métodos **morseToAlfa**, que recibe un string en clave morse, con cada letra separada por un **"/"**, y retorna un string con la traduccion. Y el método **traducirArchivo**, que recibe el nombre del archivo a traducir, y el nombre del archivo donde debe guardar la traducción.

i.e.

Encabezado	Descripción	Ejemplo de uso
Public Traductor();	Constructor	Traductor t = new Traductor();
Public String morseToAlfa(String mensaje)	Traduce de Morse al alfabeto	String message = t.morseToAlfa(elmensaje);
public void traducirArchivo(String mensaje, String traducido)	Traduce un archivo Morse y guarda en otro archivo	t.traducirArchivo("mensaje.txt", "mensajetrad.txt"

Solución

Lo primero que debemos hacer al momento de disponernos a crear una nueva clase es pensar cuales serán las variables de isntancia necesarias para el funcionamiento de la misma. Generalmente hablamos de las variables que guardan las caracteristicas de un objeto de la clase y objetos de otras clases de las cuales se usan sus metodos.
En este caso nuestro traductor no necesitara variables para guardar caracteristicas, pero si necesitara utilizar metodos de la clase "Morse" por lo que necesitaremos un objeto de esa clase.

```
import java.io.*;

public class  Traductor {

      private Morse m;
```

Lo segundo que debemos programar es el constructor de la clase, sin el la clase no sirve de mucho. Dentro del constructor damos los valores iniciales a nuestras variables de instancia, en este caso nuestra variable m.

```
	public Traductor (){
		m = new Morse();
	}
```

Y ahora ponemos nuestra atencion en los metodos y funciones que queremos nuestra clase disponga.
Para la funcion "morseToAlfa" necesitaremos parsear el parametro mensaje utilizando las ocurrencias de "/" para separar cada letra del mensaje. Como la traduccion sera letra a letra utilizaremos una variable "traduccion" que acumulara el mensaje y dos variables indices que indicaran el inicio y el fin de una letra morse encontrada.
El procesamiento del mensaje lo haremos en un ciclo que busca el carácter "/" y recupera la letra encontrada en cada iteracion traduciendola y agregandola a nuestra variable "traduccion". Como el carácter "/" funciona como un separador de palabras la ultima letra del mensaje no podremos obtenerla por este metodo y tendremos que recuperarla como el resto del mensaje desde la ultima ocurrencia de "/".

```
	public String morseToAlfa(String mensaje) {
		String traduccion = "";
		int i = 0;
		int j = mensaje.indexOf("/");
		while (j != -1) {
			traduccion += m.toAlfabeto(
				mensaje.substring(i,j));
			i = j+1;
			j = mensaje.indexOf("/", i);
		}
		traduccion += m.toAlfabeto(mensaje.substring(i));
		return traduccion;
	}
```

Crear el metodo "traducirArchivo" resulta mucho mas sencillo una vez que hemos creado la funcion "morseToAlfa". Utilizando esta ultima funcion solo tendremos que leer cada linea del archivo de entrada, traducirla usando "morseToAlfa" y ponerla en el archivo de salida. Sin olvidar cerrar los archivos.

```
	public void traducirArchivo(String mensaje,
			String traducido)throws IOException {
		BufferedReader L = new BufferedReader(
			new FileReader(mensaje));
		PrintWriter E = new PrintWriter(
			new FileWriter(traducido));
		String linea;
		while ((linea = L.readLine()) != null) {
			E.println(morseToAlfa(linea));
		}
		L.close();
		E.close();
	}

}
```

13. Fracciones (Propuesto)

(a) Escribir la clase Fraccion que contenga los siguientes métodos:

Ejemplo	Significado
Fraccion(No,No)	Constructor que recibe valores enteros para el númerador y el denominador
a.sumar(b)	sumar Fraccion b a Fraccion a
a.multiplicar(b)	Multiplicar Fraccion a por Fraccion b
a.comparar(b)	0 si a=b, No<0 si a0 si a>b
a.copiar(b)	copiar Fraccion b en Fraccion a
a.simplificar()	simplificar Fraccion a
a.toString()	entregar String con Fraccion a expresada en la forma No/No

(b) Escribir un programa, que utilice la clase Fraccion para leer una cantidad indeterminada de fracciones y escribir su promedio y la fracción de mayor valor, ambos como fracciones simplificadas. El programa debe responder al diálogo indicado en el siguiente ejemplo:

```
Promedio y Mayor de lista de fracciones
Fin de datos se indica con denominador cero
Numerador ?
Denominador ?
Suma = No/No
...
Numerador ?
Denominador ? 0
Promedio = No/No
Mayor = No/No
```

14. Notas de Controles (Propuesto)

En CC10A, las notas de los controles se guardan en archivos CONTROLES_xx.TXT en donde xx indica el número de la sección de computación. Por ejemplo, las notas de la sección 06 estaría en el archivo CONTROLES_06.TXT.

El formato de estos archivos es bastante peculiar y es el siguiente:

<username>:<nombre completo>:<codigo alumno>:<c1>:<c2>:<c3>:<c4>:<c5>

Cada campo está separado por ":". Por ejemplo, el archivo puede ser:

```
aacuna:Angel Acuna Avila:254:3.7:2:7:5.3:6.4
aalvarez: Avelina Alvarez Barraza:123:7:6:5.3:1:1
...
```

Pero este formato no siempre posee todas las notas de los controles. Es decir, a medida que se van realizando los controles, los campos en cada línea del archivo van aumentando. Es así como a principio de año solo se observan 3 campos:

```
joconcha:Jorge Concha Concha:12
```

y durante el año este va creciendo. Para el control 1:

```
joconcha:Jorge Concha Concha:12:7
```

el control 2:

```
joconcha:Jorge Concha Concha:12:7:4.3
```

etc, hasta el control 5:

```
joconcha:Jorge Concha Concha:12:7:4.3:1:2.7:6.5
```

El procesamiento de estos archivos requiere un gran trabajo por parte de los profesores y es por eso que necesitan de algo que les automatice PARA CUALQUIER SECCION este trabajo.

Un Arquitecto de Software que conoce el concepto de Orientación al Objeto (pero no cacha cómo programar) definió que la forma de procesar esto era creando un "parser" que trabajara con cada línea por separado (sin preocuparse de donde provenía) y que utilizando las funcionalidades de este parser el usuario pudiera obtener el i-ésimo parámetro o saber cuántas notas trae el archivo. Con esta premisa definió una clase **Parser** con las siguientes características:

ELEMENTO	DESCRIPCION
String linea	**Variable de Instancia** que representa la línea con el cual el parser está trabajando.
Parser(String)	**Constructor** que permite entregarle al parser cuál será la línea con la cual debe trabajar.
String saca(int)	**Método** que recibe un entero que indica el número del parámetro que debe retornar (partiendo de 1). Si el parámetro no está, retorna *null*.
int cuenta()	**Método** que permite saber cuántos campos tiene la línea.

a)Escriba la clase **Parser** con todos sus métodos.

b)Escriba un programa principal que responda al siguiente diálogo:

```
Ingrese sección? 06
PROMEDIOS DE LA SECCION 06
Angel Acuna Avila - 3.7
Avelina Alvarez Barraza - 5.7
...
Total: 97 alumnos
```

Nota: No es necesario hacer la parte (a) para hacer la parte (b)