PRIMER SEMESTRE 2002

GRUPO # 22

Alumnos:

Aguilar Elba
Barrios Miguel
Camacho Yaquelin
Ponce Rodríguez Jhonny

Asistencia a Clases

FECHA: Jueves 2 de Mayo Del 2002

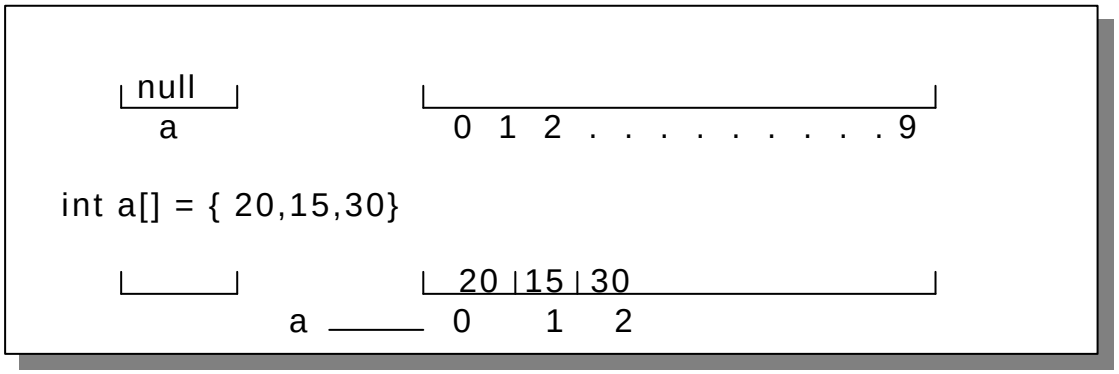
1. APRENDIZAJE

En esta clase se estudiaron los arreglos en Java y sobre como se manejan en la memoria del computador.
Ejem:

```
Q( )
{
   int a[]
   a = new int [10]
   a[0] =20;
   a[1]=15;
   a[2]=30;
   int n=3
   mostrar (a,n);
}

static void mostrar ( int a[], int n)
{
for ( int i=0, i<n; i++)
   system.out.println (a[i] + "/t");
system.out.println ();
}
```

En la memoria se representa de la siguiente forma:



Los pasos de parámetros a métodos de variables de tipo de datos básicos o predefinidos se pasan por valor y los objetos se pasan por referencia (Java interpreta un arreglo como un objeto)

2. AUTO APRENDIZAJE

Se deben escribir programas a modo de procesar casos excepcionales que puedan ocurrir durante la ejecución del programa. Esto da lugar a que el programa se ejecute con menor probabilidad de errores.

Referencia Bibliográfica.- Como Programare en Java – Deitel&Deitel

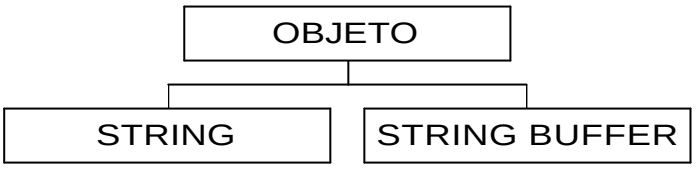
Pag. 77

Asistencia a Clases

FECHA: Viernes 3 de Mayo Del 2002

1 APRENDIZAJE

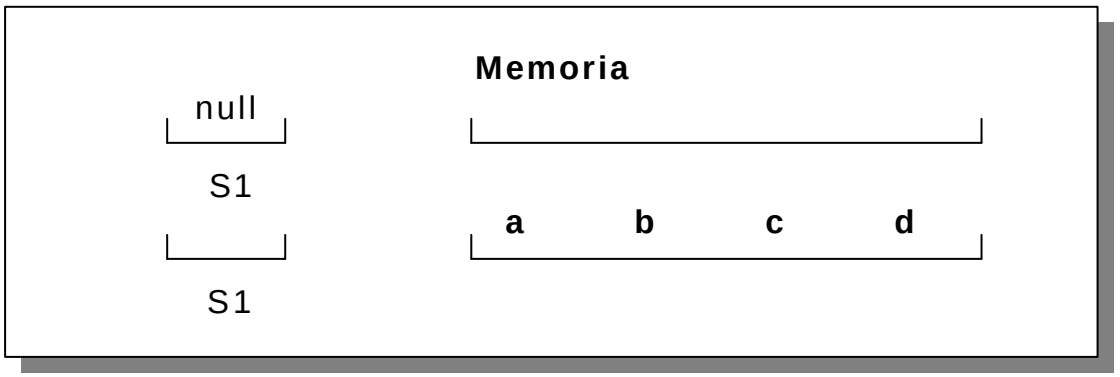
Cadenas:
En esta clase se estudió como son las cadenas en Java, para empezar diremos que las cadenas son objetos de Tipo String.



La diferencia entre un tipo y otro es que un objeto de tipo String no cambia de tamaño, de espacio de memoria ni de valor. Mientras que un String Buffer puede cambiar de tamaño y valor.
Analizaremos la clase String que internamente maneja un arreglo de char, es por eso que maneja índices de posiciones de caracteres, ejem:

```
Q( )
{
   string S1
   s1 = new string (“abcd ”);
   string S=”xyz”
   string S3=S2
   if (S1==S3)
        system.out.println(“Cad=”)
}
```

Esto se representa en memoria como:



También estudiamos los arreglos de String, su declaración:

```
String A[]                  // arreglo de cadenas
String A[] = {“abcde”,”xyz”,”pq”};
System.out.println(A.lenght);
```

Estos arreglos se los usa frecuentemente en Java apareciendo como argumento del método principal main:

```
Public static  void main ( string s1[])
{
-
}
```

También en esta clase trabajamos en grupos de tres personas con los siguientes compañeros:

Carlos Guerra	Grupo # 16
Miguel Barrios	Grupo # 15

2 AUTO APRENDIZAJE

La inicializacion de variables cuando se declaran en los metodos ayudan al programador a evitar mensajes del compilador previniendo de datos no inicializados. Las variables de ejemplar del tipo de datos primitivo int se inicalizan automáticamente en 0 y las variables de ejemplar del tipo de datos primitivo boolean se inicalizan automáticamente en false.

Referencia Bibliográfica.- Como Programare en Java – Deitel&Deitel

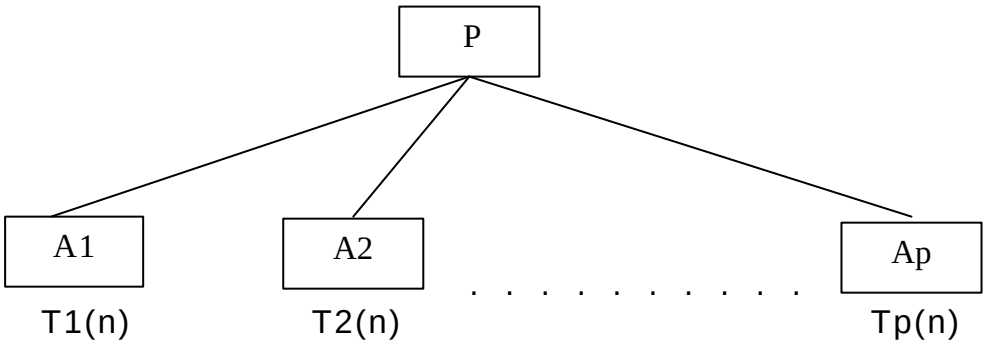
Pag. 90

Asistencia a Clases

FECHA: Martes 7 de Mayo del 2002

1. APRENDIZAJE

Análisis de Algoritmos:
La solución de un problema se la puede efectuar de diversas maneras, pero algunas son mejores si medimos su tiempo de ejecución.



Los algoritmos se miden por: la cantidad de memoria que ocupan, por la cantidad de veces que se ejecutan las instrucciones, por la claridad de lectura del código, etc.
Nosotros mediremos los algoritmos por la cantidad de veces que se ejecutan las instrucciones, que denotaremos por:
T(n): tiempo de ejecución de un algoritmo que depende de n.
T(n,m) : tiempo de ejecución de un algoritmo que depende de n,m
T(n,m,...): tiempo de ejecución de un algoritmo que depende de n, m,...

T(n) en las iteraciones: Sea

```
For (i=1; i<=n; i++)
{
      instr1...
      -
      -
      instrN
}
```

Suponemos que cada instrucción de i se ejecuta una sola vez en cada iteración.
¿Cuántas veces se ejecuta cada instrucción al salir de la iteración?
Resp: n veces
Justificación:

$$T(n)=n$$

Cascada de iteraciones Anidadas

```
for  ( i =1; i<=n; i++)
   for (j=1; j<=n; j++)
   {
   -
   -
   }
```

Tiempo de Ejecución: $T(n) = n^2$

```
For ( i=1; i<=n; i++)
   For (j=1;j<=n; j++)
     For (k=1; k<=n; k++)
         {
         -
         -
         }
```

Tiempo de Ejecución: $T(n) = n^3$

```
For ( i1=1; i1<=n; i1++)
   For ( i2=1; i2<=n; i2++)
         .
               .
                     .
                              For ( im=1; im<=n; im++)
                              {
                              -
                              -
                              }
```

Tiempo de Ejecución: $T(n) = n^m$; donde $n>=1$

2. AUTO APRENDIZAJE

Los tipos de datos primitivos son los bloques de construcción de tipos mas complicados, los tipos primitivos en Java son portátiles entre todas las plataformas que reconocen a Java ejem: bolean, char, byte, short, etc.
Bibliografía.- *Como Programare en Java – Deitel&Deitel Pag. 94*

Asistencia a Clases

FECHA: Viernes 10 de Mayo del 2002

1. APRENDIZAJE

En esta clase estudiamos con mas detenimiento lo que son los algoritmos recursivos y sus respectivos tiempos de ejecución. Como por ejem:

```
P (i,m,n)
{
   if (I<=m ) return;
   for (j=1; j<=n; j++)
         {
         -
         -
         P(i+1,m,n)
         -
         -
         }
}
```

Tiempo de Ejecución: $T(n,m) = n^m$

Cascada Ascendente de Iteraciones Anidadas.

```
for (i=1; i<=n; i++ )
{
   -
   -
}
```

Tiempo de Ejecución: T(n) = n

```
for (i=1; i<=n; i++)
   for (j=1;j<=n; j++)
   {
   -
   -
   }
```

Tiempo de Ejecución: T(n) = n(n+1)/2

Generalizando....

```
For (i1=1;i1<=n;i1++)
   For (i2=i1;i2<=n;i2++)
      -
         -
               for  (im=im-1; im<=n; im++)
               {
               -
               -
               }
```

Tiempo de Ejecución: T(n) = $\frac{n(n+1)(n+2)...(n+m-1)}{m!}$

Con Recursividad:

```
P(i,m,n)
{
      if (i>m) return;
      for (j=i; j<=n; j++)
      {
      -
      -
      -
      P(i+1,m,n)
      -
      -
      }
}
```

2. AUTO APRENDIZAJE

Un programa altamente modularizado (que son mas fáciles de probar, depurar, mantener y evolucionar) puede hacer mas llamadas a métodos que consumen tiempo de ejecución y espacio en el procesador de la computadora.

Se debe evitar usar recursión en programas en los que es importante el rendimiento, las llamadas recursivas ocupan tiempo y consumen memoria adicional.

Referencia Bibliográfica.- Como Programare en Java – Deitel&Deitel

Pag.188,189

Asistencia a Clases

FECHA: Martes 14 de Mayo Del 2002

1. APRENDIZAJE

El tema de estudio para este día fueron los tiempos de ejecución para programas de búsqueda secuencial y binaria.
Ejem. Sea un arreglo de elementos enteros, encontrar T(n) para una búsqueda secuencial de un elemento x.

```
static int BusquedaSec ( int a[], int x )
{
   int I=0;
   while ( i< a.length )
   {
   if (a[i] == x) return i;
   i=i+1;
   }
   return(-1);
}
```

Tiempos de Ejecución:

Mejor Caso: T(n) = 1
Peor Caso: T(n) = n
Caso Promedio T(n) = n/2

Ejem: Encontrar los tiempos de ejecución para una búsqueda binaria.

```
static   int BusqBin ( int a[], int x )
```

```
{
int izq = 0, der = a.length-1;
int k=( izq + der )/2;
while ( izq <= der )
{
   if ( x==a[k] ) return k;
   if ( x < a[k] )
         der = k-1;
   else
         izq =k+1;
         k=(izq+der)+2;
   return(-1)
}
}
```

Tiempos de Ejecución:

Mejor Caso: T(n) = 1
Peor Caso: $T(n) = \log_2 n$
Caso Promedio $T(n) = \log_2 n /2$

Hacer un algoritmo para determinar si un numero es primo, realizar 3 algoritmos A1, A2, A3 tales que: T1(n)<t2(n)<T3(n).

Solución.-

```
A1: static boolean primo (int n)
   {
   int I=1, c=0;
   while  (i<=n)
   {
         if n% i == 0
               c= c+1;
         I=I+1;
   }
   return (c==1);
}
```

Tiempo de Ejecución: T1(n) = n

A2: satic bolean primo (int n)

```
{
int i=1, c=0, lim = n/2;
while (i<=lim)
{
      if (n%i==0)
            c=c+1;
      i=i+1;
}
return (c==1);
}
```

Tiempo de Ejecución: T2(n) = n/2

A3: static boolean primo (int n)

```
{
int i=1, c=0, lim= Math.sqrt(n);
while (i<=lim)
{
      if (n%i==0)
            c=c+1;
      i=i+1;
}
return (c==1);
}
```

Tiempo de Ejecución: T3(n) = $\sqrt{n}$

En esta clase también trabajamos en grupos de 3 alumnos, trabajé con los siguientes compañeros:

Walter Rivero	Grupo # 16
Miguel Barrios	Grupo # 15

2. AUTO APRENDIZAJE

Se deben pasar los arreglos por referencia por razones de rendimiento, si estos se pasaran por valor se pasaría una copia de cada elemento. Esto constituiría un desperdicio de tiempo y consumiría un espacio de almacenamiento considerable para guardar las copias de los arreglos

Cuando un arreglo entra en un ciclo, el subíndice del arreglo nunca debe llegar a ser menor a 0 y siempre debe ser menor que el numero total de elementos del arreglo. Para esto debemos asegurarnos que la condición para terminar el ciclo impida el acceso a elementos fuera de rango.

Referencia Bibliográfica.- *Como Programare en Java – Deitel&Deitel*

Pag. 230