INTRODUCCIÓN A LA PROGRAMACIÓN

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN
Luz A. Sánchez Gálvez
sanchez.galvez@correo.buap.mx
luzsg@hotmail.com

Declaración de variables array

- **Sintaxis:**

  tipo [] NombreArreglo;

  - **tipo** es el tipo de los elementos del arreglo a la que la variable que se declara se referirá
  - **nombreArreglo** es el nombre de la variable del arreglo que se declara

  Semántica:

  - Declara una variable **nombreArreglo** que puede referenciar a un objeto arreglo con elementos del **tipo**.
  - **Ejemplo**:

    ```
    int [] a;    // a es una variable del tipo de referencia a un
                 // arreglo de enteros
    ```

Declaración de variables array

- **Nota**: Al declarar una variable del tipo de referencia a un arreglo aún no se ha construido el objeto array al cual la variable se refiere, y por lo tanto el arreglo no se puede utilizar todavía.

Creación de un objeto array

- Para utilizar un arreglo, primero se debe crear, especificando el número de elementos que debe tener.
- **Sintaxis**:
  new tipo [dimensión]

  - **tipo** es el nombre del tipo de los elementos del objeto arreglo que se quiere crear
  - **dimensión** es una expresión de tipo **int** que se evalúa como un valor positivo (o cero) y que especifica la dimensión del arreglo

Creación de un objeto array

- **Semántica**:
  Crear un objeto arreglo con la dimensión de elementos del tipo **tipo** y devuelve una referencia al objeto creado.
- Los elementos del arreglo son indexados y cada uno se inicializa con el valor por default para el tipo.
- **Ejemplo**:

```
int [] a;      // a es una variable de tipo arreglo de enteros
a = new int [5];   // se crea un objeto arreglo con 5
                   //elementos de tipo int asociado a la
                   //variable arreglo a
```

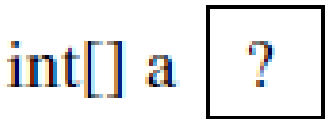
Creación de un objeto array

- Después de haber creado un objeto arreglo asociado a una variable, es posible acceder a los elementos individuales de la colección contenida en el arreglo. Estos elementos se inicializan con el valor por default para el tipo (por números enteros, se inicializan a 0).

Creación de un objeto array

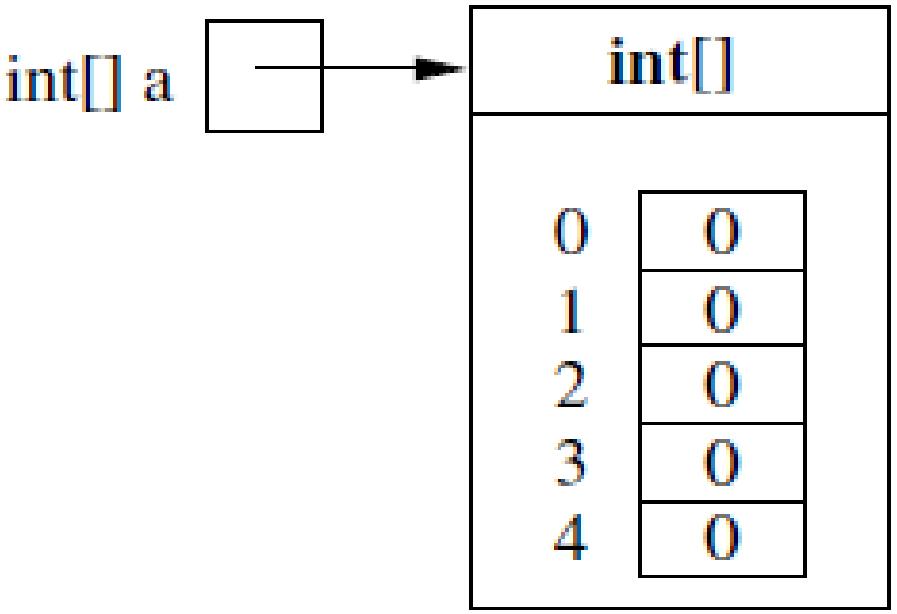
Después de la declaración

int[] a;



Después de la sentencia

a = new int[5];



Acceso a los elementos de un array

- Cada elemento individual de un objeto arreglo se puede acceder como si fuera una variable independiente. El acceso a los elementos individuales de un arreglo
- **Sintaxis:**
  nombreArreglo [indice]
  donde
- nombreArreglo es el nombre de la variable arreglo que contiene una referencia al arreglo en el que se desea acceder.
- índice es una expresión de tipo int con valor no negativo que especifica el índice del elemento que se desea acceder.

Acceso a los elementos de un array

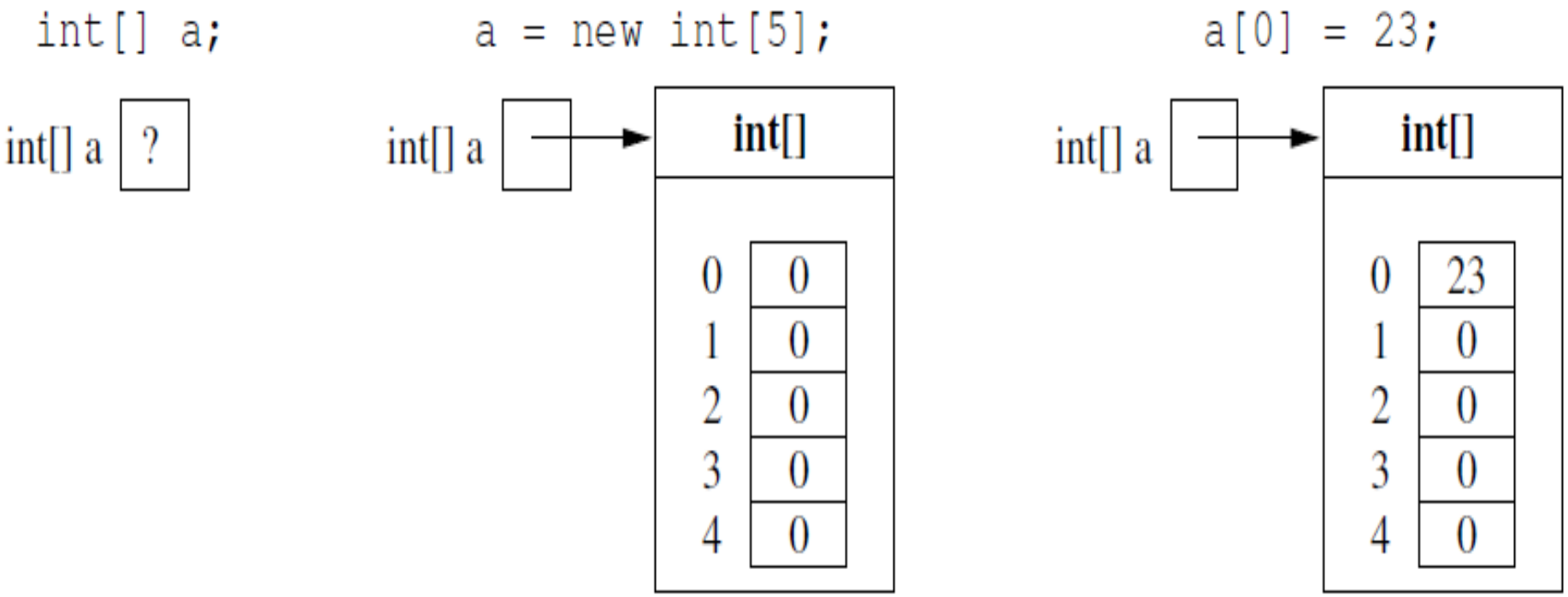
- **Semántica:**
- Accede al elemento del arreglo nombreArreglo en la posición especificada por índice permite leer o modificarlo.
- Si el arreglo nombreArreglo contiene N elementos, la evaluación de la expresión índice debe devolver un número entero en el intervalo [0, N - 1], y si este no es el caso, un error en tiempo de ejecución ocurre cuando se ejecuta el programa.

Acceso a los elementos de un array

- **Ejemplo**:

```
int [] a;              // a es una variable de tipo arreglo de enteros
a = new int [5];       // un objeto arreglo con 5 elementos de tipo int
                       // asociado a la variable arreglo a
```

- a[0] = 23; // asignamiento al primer elemento del arreglo



Número de elementos de un array: variable de instancia *length*

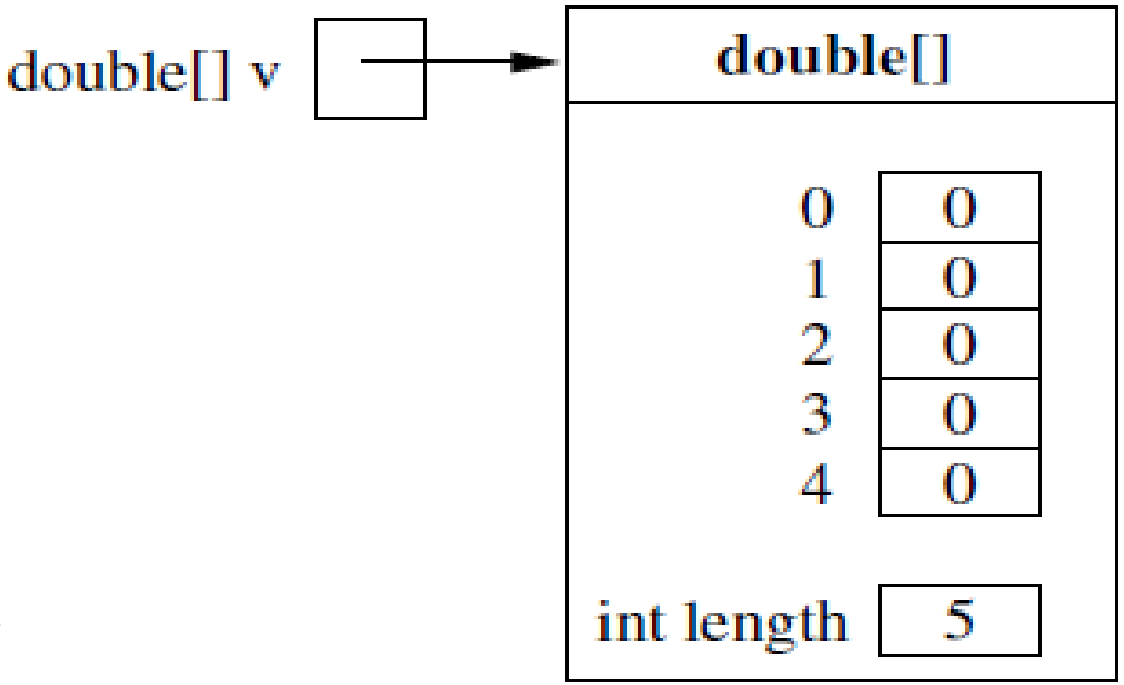
- Cada objeto del arreglo contiene, además de la colección de elementos, una variable de instancia pública *length* que almacena el número de elementos del arreglo.
- Mediante el acceso a la variable length es posible obtener el número de elementos.
- **Ejemplo:**

```
double[] v;
v = new double[5];
System.out.println(v.length);        // imprime 5
```

Número de elementos de un array: variable de instancia *length*

- **Ejemplo:**

```
double[] v;
v = new double[5];
System.out.println(v.length);          // imprime 5
```



Expresiones que denotan objetos array

- En Java, es posible escribir expresiones que denotan objetos arreglo, similarmente a las cadenas, por ejemplo, "ciao" denota un objeto String.
- Una expresión que denota un objeto arreglo es una lista de expresiones del tipo de los elementos del arreglo, de la forma: {*expresion1, expresion2, ..., expresionk*} y denota un objeto arreglo de k elementos.
- **Ejemplo:**

```
int[] v = { 4, 6, 3, 1 };
```

es equivalente a:

```
int[] v = new int[4];
v[0] = 4; v[1] = 6; v[2] = 3; v[3] = 1;
```

Expresiones que denotan objetos array

- **Nota**: Mientras que una cadena, tal como "ciao" se puede utilizar en cualquier parte del cuerpo de un método para denotar un objeto cadena, las expresiones que denotan un arreglo puede ser utilizadas sólo para inicializar un arreglo cuando es declarado.

Ejemplo de un fragmento de código erróneo:

```
int [] v;
v = {4, 6, 3, 1};        / / ¡ERROR!
```

Suma de elementos de un array de enteros

- Un método estático, sumaValoresArreglo() que tiene como parámetro un arreglo de enteros y devuelve la suma de los valores de los elementos del arreglo.

```
public static int sumaValoresArreglo(int[] v) {
        int suma = 0;
        for (int i = 0; i < v.length; i++)
            suma= suma+v[i];                    //suma += v[i];
        return suma;
    }
```

Suma de elementos de un array de enteros

```
public static void main(String[] args) {
    // se crea un arreglo de 5 elementos
   int[] x = new int[5];
   // se asigna al elemento x con índice i el valor 2*i
   for (int i = 0; i < x.length; i++)
      x[i] = 2*i;
    // imprime la suma de los 5 elementos del arreglo
   System.out.println(sumaValoresArreglo(x));
}
```

Búsqueda de un elemento en un array

- Un método estático, buscarArreglo con parámetros un arreglo de cadenas y una cadena; que devuelve verdadero si la cadena está presente en el arreglo, y falso en caso contrario.

```
public static boolean buscarArreglo(String[] v, String e) {
   for (int i = 0; i < v.length; i++)
      if (e.equals(v[i]))
         return true;
    return false;
}
```

Búsqueda de un elemento en un array

```
public static void main(String[] args) {
    // se crea un arreglo x de 3 cadenas
    String[] x = {"uno", "dos", "tres"};
    // se busca la cadena "dos" en el arreglo x
    if (buscarArreglo(x, "dos"))
       System.out.println("presente");
     else
       System.out.println("no presente");
 }
```

Búsqueda del elemento máximo en un arreglo de números

- Un método estático maxArreglo() con parámetro un arreglo de enteros y devuelve el valor máximo del mismo. Se asume que el arreglo contiene al menos un elemento.

```
public static long maxArreglo(long[] v) {
   long max = v[0];
   for (int i = 1; i < v.length; i++)
      if (v[i] > max) max = v[i];
      return max;
}
public static void main(String[] args){
   long[] x = { 42, 97, 31, -25 };          // se crea un arreglo x
   System.out.println(maxArreglo(x));    // imprime 97
}
```

Invirtiendo el contenido de un arreglo

- Un método estático invierteArreglo() con parámetros un arreglo de enteros que modifica invirtiendo el orden de sus elementos (el primero se convierte en el último de ellos, el segundo en el penúltimo, etc.):

```
public static void invierteArreglo(int[] v) {
   for (int i = 0; i < v.length/2; i++) {
      int temp;
      temp = v[i];
      v[i] = v[v.length-1-i];
      v[v.length-1-i] = temp;
   }
}
```

Invirtiendo el contenido de un arreglo

```
public static void main(String[] args) {
    int[] x = { 5, 3, 9, 5, 12};      //se crea un arreglo x de 5 enteros
    for (int i = 0; i < 5; i++)      //imprime 5  3  9  5  12
       System.out.println(x[i]);
       invierteArreglo(x);                  // se invierte el arreglo x
         for (int i = 0; i < 5; i++)         //imprime 12  5  9  3  5
           System.out.println(x[i]);
 }
```

- Cuando el arreglo **x** se pasa como un parámetro actual al método invierteArreglo(), el parámetro formal **v** se refiere al mismo objeto arreglo al cual se refiere **x**. Por lo tanto, todas las modificaciones hechas al arreglo dentro del método se realizan al arreglo que se refiere a **x**, y por lo tanto será visible para el método de llamada (en este caso, main ()).

Ejemplo: propietarios de departamentos

- Una clase con información sobre los propietarios de departamentos. Cada objeto de la clase debe contener una cadena que indique el nombre del propietario, y un array de 10 posiciones de tipo String, indexados por los números del 0 al 9, que contengan la dirección del departamento propiedad del propietario (o nulo, si la posición está vacía).

La clase debe exportar las siguientes funcionalidades:

- Un **constructor** cuyo parámetro sea el nombre del propietario, y crear un objeto con el propietario especificado en el que inicialmente todas las posiciones del arreglo estén vacías;
- Un método que devuelva el propietario de un departamento;
- Un método que devuelva la dirección contenida en una posición (o nulo, si la posición está vacía);
- Un método que asigne la dirección de un departamento a una posición;

La clase debe exportar las siguientes funcionalidades:

- Un método que devuelva el número de departamentos (es decir, posiciones no vacías);
- Un método que reorganice las direcciones, de tal manera que estén contenidas en las primeras posiciones consecutivas del arreglo;
- Un método toString (), que redefina al método toString () heredado de Object, y devuelva una cadena que contenga la información sobre el objeto.

```
public class PropietarioDepartamento {
    private String propietario;
    private String[] departamentos;

    public PropietarioDepartamento(String propietario) {
        this.propietario = propietario;
        departamentos = new String[10];
    }
    public String getPropietario() {
        return propietario;
    }
    public String getDepartamento(int posicion) {
        return departamentos[posicion];
    }
    public void setDepartamento(String direccion, int posicion) {
        departamentos[posicion] = direccion;
    }
    public int contarDepartamentos() {
        int num = 0;
        for (int i = 0; i < 10; i++)
            if (departamentos[i] != null)
                num ++;
            return num;
    }
```

```
public void reorganizarDepartamentos() {
    int vacio = -1;                    // índice de la primera posición vacía
    for (int i = 0; i < 10; i++) {
        if (departamentos[i] == null && vacio == -1)
            vacio = i;
            if (departamentos[i] != null && vacio != -1) {
                departamentos[vacio] = departamentos[i];
                departamentos[i] = null;
                vacio++;
                }
            }
    }
    public String toString() {
        String  res = "";
        res += "Propietario: " + propietario;
            for (int i = 0; i < 10; i++)
                if (departamentos[i] != null)
                    res += "\n Departamento " + i + ": " + departamentos[i];
                return res;
    }
}
```

```
public class PrincipalPropietarioDepartamento {

    public static void main (String[] args) {
        PropietarioDepartamento p = new PropietarioDepartamento("Ricardo Ibañez");
            p.setDepartamento("Avenida Vicente Guerrero 911", 0);
            p.setDepartamento("Avenida San Claudio , 5720", 1);
            p.setDepartamento("Avenida 14 sur, 3814", 2);
            p.setDepartamento("Río Conchos, 3856", 5);
            p.setDepartamento("Río Usumacinta, 2500", 8);
            System.out.println(p);
            System.out.println();
            System.out.println(p.getPropietario() + " tiene " +
            p.contarDepartamentos() + " departamentos");
            System.out.println("Departamento 2: " + p.getDepartamento(2));
            System.out.println();
            p.reorganizarDepartamentos();
            System.out.println(p);
    }
}
```