Universidad
Carlos III de Madrid

Tema 5

Estructuras de datos simples

Programación
2015-2016

Tema 5. Estructuras de datos simples

- **Introducción.**
- Arrays.
- Records.

¿Qué es una estructura de datos?

- Es un grupo de datos que queremos guardar agrupado
  ==> en la misma variable.

- Las estructuras de datos más simples son los “arrays” y los “records”

- Datos del mismo tipo → **array**

- Datos de distinto tipo → **record**
  - Java no dispone de record, pero se pueden utilizar objetos.

Tema 5. Estructuras de datos simples

- Introducción.
- **Arrays.**
- Records.

Arrays

- Ejemplo: temperatura máxima en Colmenarejo durante una semana (7 variables) ¿y un año?
- **Definición**: conjunto de datos que ocupan posiciones sucesivas de memoria y a los que se accede mediante una única variable.
- Definición de arrays, 3 pasos:
  - Declarar la variable — `tipo [] nombre` ó `tipo nombre []`
  - Definir el nº de elementos — `nombre = new tipo [dimension]`
  - Reservar memoria

```
int [] a , b;
int [] a = new int [3], b, c= new int [2];
```

Arrays y memoria

- Un array es un **puntero:** Una variable que no contiene un valor, sino una referencia a otra posición de memoria.
- Cuando se declara..
  - Una variable → contenido indefinido.
  - Un array → **null** (valor especial, “no apunta a nada”)

```
int [] arr1 = new int [4];
```

- **Valor inicial**?

Arrays y memoria

- **Valor inicial**:
  - 0 a números
  - false a lógicos
  - la cadena vacía a char
  - null a String

Guardar datos en arrays

- Acceso a elementos del array:

  nombre [posición] = valor

Posición va de 0 a nº elementos-1.
(Java comprueba en ejecución)

- **Asignación inicial:**

```
int [] a; a = new int [] {1,2,3};
int [] a = new int [] {1,2,3,4}, b= {1,2};
int [a]; a = {1,2,3};
```

Guardar datos en arrays

- ¿Qué ocurre si imprimimos un Array?

Arrays y asignaciones

- **Igualando elementos** de dos arrays. Los elementos de arrays sí se comportan como variables normales
- **Copia de arrays**. Si igualamos dos arrays, no se copian sino que apuntan al mismo. Incluso aunque sean de distinto número de elementos
  - Para copiar hay que hacerlo elemento a elemento:
    - usar System.arraycopy (origen, pos, destino, pos, nº elementos),
    - usar un bucle

Arrays y asignaciones

- Dos arrays sólo son **iguales** con == si apuntan a la misma posición de memoria. Para ver si son iguales, hay que comparar elemento a elemento.

```
int [] array1={1,3,5}, array2={1,3,5};
array1 == array2 → false
```

Longitud de un array

- **Nombre.length** devuelve el número de elementos

- No se puede cambiar la longitud de un array.
  - Una vez definido el nº de elementos no se puede variar. Hay que crear uno nuevo y copiar los elementos.

- Array de constantes. Aunque declaremos un array como final, sus elementos se pueden cambiar (da igual que les hayamos dado valor al crearlo o que se lo demos luego). Lo que es constante es la posición de memoria a la que apunta.

Longitud de un array

- **Nombre.length** devuelve el número de elementos

- No se puede cambiar la longitud de un array.
  - Una vez definido el nº de elementos no se puede variar. Hay que crear uno nuevo y copiar los elementos.

- Array de constantes. Aunque declaremos un array como final, sus elementos se pueden cambiar (da igual que les hayamos dado valor al crearlo o que se lo demos luego). Lo que es constante es la posición de memoria a la que apunta.

Matrices

- Declaración y creación, similar a los de una sola dimensión
- El primer número son las filas y el segundo las columnas.
  - En memoria, cada fila es un array

```
tipo [][] nombre ó tipo nombre [] []
nombre = new tipo [d1][d2]
tipo [][] nombre = new tipo [d1][d2]
tipo [][]nombre = {{1,2},{3,4},{5,6}}
```

- Acceso a elementos de la matriz:

nombre [fila][columna] = valor;

Matrices

- Se puede declarar solamente el nº filas y dejar la segunda dimensión sin crear → se pueden crear arrays irregulares.

```
int [][] a;
a = new int [3][];
a[0] = new int [1];
a[1] = new int [3];
a[2] = new int [2];
```

```
a.length → 3 (filas)
a[0].length → 1
a[1].length → 3
a[2].length → 2
```

- Multi-dimensional arrays
  - Tantas dimensiones como se quiera.

Matrices y bucles anidados

- Ejercicio: Imprimir un array de dos dimensiones.

```
int [][] arr = new int [][]{{1,2,3},{4,5},{6,7,8,9}};
```

Matrices y bucles anidados

- Ejercicio: Imprimir un array de dos dimensiones.

```
int [][] arr = new int [][]{{1,2,3},{4,5},{6,7,8,9}};
```

```
for (<tipo del array> elemento : <nombre del array>)
```

```
for (int [] filas : arr){
    for (int elemento: filas){
        System.out.print(elemento+" ");
    }
    System.out.println();
}
```

Ejercicio

S6-Clase: Arrays

Tema 5. Estructuras de datos simples

- Introducción.
- Arrays.
- **Records.**

Recorsd → Objetos

- En Java no hay registros como tal, sino **objetos**.
- Un objeto es un nuevo tipo de datos, que nos permite guardar datos de **distintos tipos** en la **misma variable**.
- El nombre del tipo es el nombre de la clase (que debe coincidir con el nombre del fichero .java)
- Definimos cada uno de los datos que queremos guardar así:
  - public <tipo> <variable>;

```
public class Fecha {
  public int dia;
  public String mes;
  public int año;
  public boolean festivo;
}
```

Objetos

- Una **clase** puede contener:
  - un programa (o bloques de programa)
  - un nuevo tipo (la definición de una estructura de datos)

- Record → Sin método main

- Generalmente los programas constan de muchas clases que definen objetos y una que tiene el main

Declaración de variables

- En el main de otra clase distinta
  - No hace falta importarlo si están en el mismo paquete (en la misma carpeta en el disco)
- Se declara la variable de este tipo como la de cualquier otro
- Hace falta crearla para poder empezar a usarla (parecido a los arrays)
  - Si no los creamos están a null.
  - Son **punteros**.

Nomenclatura

- **Atributo\Campo**: cada una de las variables que conforma un nuevo tipo (la doc oficial usa **Field**)
- **Clase**: definición del nuevo tipo
- **Objeto**: variable del nuevo tipo
- Declarar y crear una variable de un tipo que es una clase == declarar y crear un objeto

Uso de variables: acceso a atributos

- Para acceder a un atributo
  - <nombre variable>.<nombre atributo>

```
Fecha f1;
f1 = new Fecha ();
f1.dia = 12;
f1.mes = "Noviembre";
f1.año= 2010;
```

- Ej.
  - Crear dos variables y darles distintos valores
  - Intentar acceder a un atributo de un objeto que no ha sido creado

Valores iniciales

- ¿Qué ocurre si intentamos utilizar un atributo sin darle valor?
  - Valores iniciales (tipos básicos / String / otros objetos)

- Se pueden asignar otros valores iniciales distintos cuando se declara la clase:

```
public class Fecha {
   public int dia = 1;
   public String mes = "Enero";
   public int año = 1900;
   public festivo = true;
}
```

Manipulando objetos

- **Imprimir**
- **Comparar**
- **Copiar**

- No se puede directamente. Deben hacerse atributo a atributo.
- Existen formas de modificar este comportamiento.

Resumen de variables

- Sitios en los que se declaran variables:
  - **Dentro del método main:** variables locales, sin valor inicial automático
  - **En una clase aparte para crear un nuevo tipo:** atributos, con valor inicial automático
  - Si hay main, NUNCA se deberían declarar variables fuera de él

Ejercicio

S7-Clase: Objetos