Universidad Autónoma de Tlaxcala

Introducción al lenguaje JAVA

M.C. José Juan Hernández Mora

Segunda Sesión

1. Arreglos
2. Matrices
3. Clases en Java
4. Clases de Usuario en Java
5. Objetos definidos por el usuario
6. La sentencia ***static***
7. Clases de la librería de Java
8. La clase ***Math***
9. La clase ***String***

Arreglos

Los ***arrays*** de ***Java*** (vectores, matrices, hiper-matrices de más de dos dimensiones) se tratan como objetos de una clase predefinida.

Los ***arrays*** son ***objetos***, pero con algunas características propias.

Los ***arrays*** pueden ser asignados a objetos de la clase ***Object*** y los métodos de ***Object*** pueden ser utilizados con ***arrays***.

Arreglos

- Los ***arrays*** se crean con el operador ***new*** seguido del tipo y número de elementos.
- Se puede acceder al número de elementos de un array con la variable miembro implícita
- ***length*** (por ejemplo, ***vect.length***).
- Se accede a los elementos de un ***array*** con los ***corchetes []*** y un ***índice*** que varía de 0 a
- ***length-1***.
- Se pueden crear ***arrays*** de objetos de cualquier tipo. En principio un ***array*** de objetos es un ***array de referencias*** que hay que completar llamando al operador ***new***.

Arreglos

- Los elementos de un ***array*** se inicializan al valor por defecto del tipo correspondiente (cero para valores numéricos, la cadena vacía para ***Strings***, ***false*** para ***boolean***, ***null*** para referencias).
- Como todos los objetos, los ***arrays*** se pasan como argumentos a los métodos ***por referencia***.

Inicialización de ***arrays***

- Los ***arrays*** se pueden inicializar con valores entre llaves {...} separados por comas.
- También los ***arrays de objetos*** se pueden inicializar con varias llamadas a ***new*** dentro de unas llaves {...}.
- Si se igualan dos referencias a un array no se copia el array, sino que se tiene un array con dos nombres, apuntando al mismo y único objeto.

Ejemplo

- Lisarr.java

Inicialización de ***arrays***

- Creación de una ***referencia*** a un array. Son posibles dos formas:
  - double[] x; // preferible
  - double x[];
- Creación del ***array*** con el operador ***new***:
  - x = new double[100];
- Se pueden unir en una sola:
  - double[] x = new double[100];

Matrices

- Los arrays bidimensionales de ***Java*** se crean de un modo muy similar al de C++ (con reserva dinámica de memoria).
- En ***Java*** una ***matriz*** es un ***vector*** de ***vectores fila***, o más en concreto un vector de referencias a los vectores fila.
- Con este esquema, cada fila podría tener un número de elementos diferente.

Matrices

- Una matriz se puede crear directamente en la forma,
  - `int [][] mat = new int[3][4];`
- O Crear la ***referencia*** indicando con un doble corchete que es una ***referencia a matriz***,
  - `int[][] mat;`
- Crear el vector de referencias a las filas,
  - `mat = new int[nfilas][];`

Matrices

- En el caso de una matriz ***b***, ***b.length*** es el número de filas y
- ***b[0].length*** es el número de columnas (de la fila 0).

Ejemplo

- Mat1.java

Clases y métodos

Clases

Una clase es una agrupación de ***datos*** (variables o campos) y de ***funciones*** (métodos) que operan sobre esos datos.

La definición de una clase se realiza en la siguiente forma:

[public] **class** Classname {
// definición de variables y métodos
...
}

Objetos

- Un ***objeto*** (en inglés, ***instance***) es un ejemplar concreto de una clase.
- Las ***clases*** son como tipos de variables, mientras que los ***objetos*** son como variables concretas de un tipo determinado.

Clases

- Así, si definimos una clase, ésta podrá instanciarse tantas veces como se quiera.
- Pero todos los objetos creados tendrán la misma descripción y el mismo comportamiento.
- Es decir, la descripción de un objeto viene dada por sus atributos

- Por ejemplo, se define la clase ventana como:

```
class ventana {
   string titulo;
   int coordx;
   int coordy;
   int altura;
   int anchura;
}
```

¿ Cómo crear un objeto ?

- Mediante la palabra clave new, la cual reserva el espacio de memoria necesario para el objeto, el cual, el espacio, se calcula a partir de la descripción de la clase, y se llama al constructor del objeto.
- El constructor es un método cuyo nombre es idéntico al de la clase y que efectúa las operaciones que el programador le indica inmediatamente tras la creación del objeto.

Objetos

- Por ejemplo, se define los objetos ven1 y ven2 de la clase ventana y se llama al constructor:

```
ventana ven1 = new ventana ("Word",20,20,100,100);
ventana ven2 = new ventana ("Excel",22,05,19,64);
```

La sintaxis de este constructor será la siguiente:

```
ventana (string nombre, int x, int y, int b, int h, int l)
{
   titulo = nombre;
   coordx = x;
   coordy = y;
   altura= h;
   anchura = l;
}
```


las clases describen los atributos de los objetos, y proporcionan también los métodos

- Un método es una función que se ejecuta sobre un objeto
- Los atributos del objeto son implícitamente parámetros del método
- Al ejecutarse un método lo hace como si estuviese dentro del objeto

CONSTRUCTORES

- El constructor se llama en el momento de la creación de un objeto. La utilización de *new( )* implica la creación física del objeto y la llamada a uno de sus constructores. Si hubiera más de un constructor, éstos se difieren por los parámetros que se pasan en el *new( )*.

Constructores

Los constructores no tienen tipo de retorno.

Si por error definimos un constructor que tenga un tipo de retorno, el compilador lo toma como un método normal.

En tal caso se tendrá la impresión de que el constructor no es llamado en el momento de la creación del objeto.

Ejemplo

- pila.java

Superclases

- Para especificar explícitamente la superclase de una clase, se debe poner la palabra clave **extends** más el nombre de la superclase entre el nombre de la clase que se ha creado y el corchete abierto que abre el cuerpo de la clase, así:

```
class NombredeClase extends NombredeSuperClase
 {
 . . .
  }
```

Subclases

- Crear una subclase puede ser tan sencillo como incluir la cláusula **extends** en su declaración de clase.

Clases Public, Abstract, y Final

- Se puede utilizar uno de estos tres modificadores en una declaración de clase para declarar que esa clase es pública, abstracta o final.
- Los modificadores van delante de la palabra clave **class** y son opcionales.

public

- El modificador **public** declara que la clase puede ser utilizada por objetos que estén fuera del paquete actual. Por defecto, una clase sólo puede ser utiliza por otras clases del mismo paquete en el que están declaradas.

abstract

- El modificador **abstract** declara que la clase es una clase abstracta.
- Una clase abstracta podría contener métodos abstractos (métodos sin implementación).
- Una clase abstracta está diseñada para ser una superclase
- Una clase abstracta es una clase que sólo puede tener subclases--no puede ser ejemplarizada.

final

- Utilizando el modificador **final** se puede declarar que una clase es final, que no puede tener subclases. Existen (al menos) dos razones por las que se podría querer hacer esto: razones de seguridad y razones de diseño.

Interfaces Implementados por la Clase

- Un interface declara un conjunto de métodos y constantes sin especificar su implementación para ningún método.
- Para declarar que una clase implementa uno o más interfaces, se debe utilizar la palabra clave **implements** seguida por una lista de los interfaces implementados por la clase delimitada por comas.

- La clase NumeroImaginario puede declarar que implementa el interface Aritmetico de esta forma:

```
class NumeroImaginario extends Number
  implements Aritmetico
{
. . .
}
```


En suma, una declaración de clase se parecería a esto:

[**modificadores**] class **NombredeClase**
[extends **NombredeSuperclase**]
[implements **NombredeInterface**]
{
. . .
}

Variables Miembro

- Una clase en Java puede contener variables y métodos.
- Las variables pueden ser tipos primitivos como int, char, etc.
- Los métodos son funciones.

VARIABLES Y MÉTODOS ESTÁTICOS

- Como ya se ha dicho, cada objeto posee sus propios atributos y es posible que todos los objetos de una misma clase tengan atributos en común: son los atributos de la clase, introducidos por la palabra clave static.

Static

- Estos atributos son legibles y modificables por todos los objetos de una misma clase.
- La modificación de un atributo static es tenida en cuenta inmediatamente por los demás objetos, porque lo comparten.

VARIABLES Y MÉTODOS ESTÁTICOS

- Análogamente, puede también haber métodos que no actúen sobre objetos concretos a través del operador punto.
- A estos métodos se les llama ***métodos de clase*** o ***static***.
- Los métodos de clase pueden recibir objetos de su clase como argumentos explícitos, pero no tienen argumento implícito ni pueden utilizar la referencia ***this***.

Ejemplo

- Cfuns2.java

Math

- La clase ***java.lang.Math*** deriva de ***Object***. La clase ***Math*** proporciona métodos ***static*** para realizar las operaciones matemáticas más habituales.

- Proporciona además las constantes ***E*** y ***PI***

Métodos de la clase Math

- abs()
  - Valor absoluto
- sin(double)
  - Calcula el seno
- acos()
  - Arcocoseno
- tan(double)
  - Calcula la tangente

Métodos de la clase Math

- asin()
  - Arcoseno
- exp()
  - Calcula la función exponencial
- atan()
  - Arcotangente entre -PI/2 y PI/2
- log()
  - Calcula el logaritmo natural (base e)

Métodos de la clase Math

- atan2(,)
  - Arcotangente entre -PI y PI
- max(,)
  - Máximo de dos argumentos
- ceil()
  - Entero más cercano en dirección a infinito
- min(,)
  - Mínimo de dos argumentos

Métodos de la clase Math

- floor()
  - Entero más cercano en dirección a -infinito
- random()
  - Número aleatorio entre 0.0 y 1.0
- round()
  - Entero más cercano al argumento
- power(,)
  - Devuelve el primer argumento elevado al segundo

Métodos de la clase Math

- rint(double)
  - Devuelve el entero más próximo
- sqrt()
  - Devuelve la raíz cuadrada
- toDegrees(double)
  - Pasa de radianes a grados (Java 2)

Métodos de la clase Math

- cos(double)
  - Calcula el coseno
- toRadians()
  - Pasa de grados a radianes (Java 2)

Ejemplo

- Trigo.java

Cadenas

- Las clases ***String*** esta orientada a manejar cadenas de caracteres.
- Los objetos de la clase ***String*** se pueden crear a partir de cadenas constantes o *literals*, definidas entre dobles comillas, como por ejemplo: "Hola".
- ***Java*** crea siempre un objeto ***String*** al encontrar una cadena entre comillas.

Cadenas

- A continuación se describen dos formas de crear objetos de la clase ***String***:

```
String str1 = "Hola";
// el sistema más eficaz de crear Strings
String str2 = new String("Hola");
// también se pueden crear con un constructor
```


Métodos de la clase **String**

- String(...)
  - Constructores para crear Strings a partir de arrays de bytes o de caracteres
- charAt(int)
  - Devuelve el carácter en la posición especificada
- length()
  - Devuelve el número de caracteres de la cadena

Métodos de la clase **String**

- toLowerCase()
  - Convierte en minúsculas
- toUpperCase()
  - Convierte en mayúsculas
- valueOf()
  - Devuelve la representación como String de sus argumento. Admite Object, arrays de caracteres y los tipos primitivos

Nota

- Un punto importante a tener en cuenta es que hay métodos, tales como ***System.out.println()***, que exigen que su argumento sea un objeto de la clase ***String***. Si no lo es, habrá que utilizar algún método que lo convierta en ***String***.

Ejemplo

Cads.java

Programas

- Clase cola
- Contar palabras y vocales de un nombre