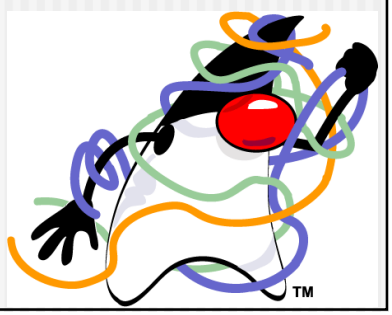
Unidad 4
Programación modular: los métodos

Programación
1º D.A.M.

1

Contenido

1. Programación modular
2. Métodos en Java
3. Tipos de métodos
4. Paso de parámetros
5. Ámbito de variables
6. El método main
7. Recursividad
8. Sobrecarga



2

1. Programación modular

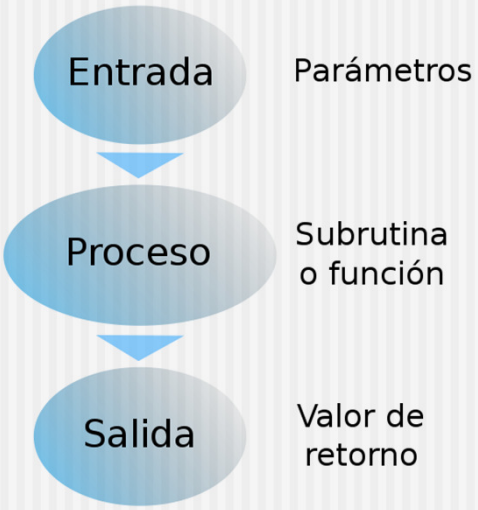
3

1. Programación modular

- Paradigma de programación
  - División de problema complejo en varios sencillos
    - Análisis descendente (Top-Down)
    - Divide y vencerás
  - Evolución del "estructurado" para problemas complejos
- Módulo
  - Cada subprograma que resuelve un problema sencillo
  - Tarea bien definida
  - Unos necesitan de otros
- Programa
  - Integración de módulos o subprogramas

4

1. Programación modular



5

1. Programación modular

- Ventajas
  - Mejora la legibilidad y manejabilidad
  - Favorece la localización de fallos
  - Mejora el mantenimiento
  - Permite la reutilización
  - Mayor abstracción en la solución

6

2. Métodos en Java

1. Sintaxis
2. Cabecera
3. Cuerpo
4. Sentencia `return`
5. Invocación

7

2. Métodos en Java

- Descripción
  - Bloque de código Java con un nombre
  - Recibe parámetros (opcional)
  - Tiene sentencias para realizar tareas (opcional)
  - Devuelve un valor (opcional)
- Módulos funcionales de una clase
  - Tarea bien definida dentro de dicha clase
  - Marcan el comportamiento de sus objetos
    - Tareas que pueden realizar

8

2.1. Métodos en Java. Sintaxis

```
Cabecera{
    Cuerpo
}
```

- Cabecera
  - Nombre del método
  - Tipo de retorno
  - Tipo y nombre de sus parámetros
- Cuerpo
  - Código propiamente dicho (instrucciones)

9

2.2. Métodos en Java. Cabecera

```
[modificadores] <tipo_devuelto>
  nombre([lista_parámetros])
```

- Modificadores
  - Configuración de aspectos adicionales
- Tipo_devuelto
  - Tipo de datos del valor devuelto
- Nombre
  - Identificador del método para invocarlo
- Lista_parámetros
  - Lista de parámetros pasados al método

10

2.2. Métodos en Java. Cabecera

- Modificadores
  - `public`,`protected`,`private`,"vacío" (friendly)
    - Visibilidad: desde dónde será visible el método
  - `static` (de clase), "nada" (de instancia)
    - Alcance: qué se requerirá para acceder al método
  - `native`
    - Método escrito en lenguaje distinto (usa JNI)
  - `abstract`
    - El método carece de implementación (cuerpo)
  - `final`
    - El método no puede redefinirse en una subclase
  - `synchronized`
    - Gestión de bloqueos para evitar concurrencia

11

2.2. Métodos en Java. Cabecera

- Tipo devuelto
  - Tipo de datos del elemento devuelto por el método
    - Tipo de datos básico
      - El método devuelve un `int`,`float`,`char`,`boolean`, …
    - Nombre de clase
      - El método devuelve un objeto de dicha clase
    - `void`
      - El método no devuelve nada

12

2.2. Métodos en Java. Cabecera

- Nombre
  - Identificador del método
  - Usado para invocarlo
  - Características
    - Ha de ser significativo
    - Mismas reglas que para variables
    - No puede ser una palabra reservada
      - `main` reservado para el método principal
    - Recomendado comienzo en minúsculas

13

2.2. Métodos en Java. Cabecera

- Lista_parámetros
  - Opcional
    - Puede no haber parámetros en el método
      - `()`
  - Lista de elementos `tipoI parámetroI`
    - `TipoI`: tipo de datos del parámetro I-ésimo
    - `ParámetroI`: nombre del parámetro I-ésimo
  - Si hay varios, separados por comas

14

2.2. Métodos en Java. Cabecera

- Ejemplos
  - `int suma(int valor1, int valor2)`
    - Devuelve la suma de dos enteros que se pasan
  - `public float longitud(int radio)`
    - Devuelve la longitud de una circunferencia de radio "radio"
  - `void muestra(String mensaje, int veces)`
    - Muestra un mensaje un número determinado de veces
  - `Fecha dimeFecha()`
    - Obtiene la fecha actual del sistema
  - `static void ordena(int v1, int v2, int v3)`
    - Muestra ordenados tres valores que recibe
  - `void muestraMenu()`
    - Visualización de un menú en pantalla

15

2.3. Métodos en Java. Cuerpo

- Instrucciones para realizar su tarea
- Bloque de código delimitado por llaves
- Fin de su ejecución
  - Si devuelve valores
    - Sentencia `return`
  - Si no devuelve valores
    - Llave de cierre
    - Sentencia `return` sin parámetros

16

2.4. Métodos en Java. Sentencia `return`

- Valor devuelto por el método

```
return <expresion>;
```

  - Tipo de dato de expresión según cabecera
  - Retorno del control a quien invocó la función
  - Puede no haber `return` (tipos `void`)
    - Devolución del control al encontrar `}`
  - Puede haber varios `return`
    - No recomendado

- Cantidad de valores devueltos
  - Por defecto sólo uno
  - Para devolver más de un valor
    - Objeto con varios campos
    - Paso de valores por referencia en argumentos

17

2.4. Métodos en Java. Sentencia `return`

- Ejemplos

```
int cubo (int base)
{
     return base * base * base;
}
void mensaje ()
{
     System.out.println("Mi mensaje");
}
```

18

2.5. Métodos en Java. Invocación

- Invocar, acceder o llamar al método
  - Paso del control al método
  - Retorno del control con `return` o `}`
  - Partes
    - Nombre del método
    - Lista de argumentos
- Ejemplos:

```
resultado = cubo(4);
imprime_mensaje();
pot = potencia(base, exponente);
```

19

3. Tipos de métodos

20

3. Tipos de métodos

- Según su alcance, pueden ser
  - Métodos de instancia
    - Atribuidos a un objeto (instancia)
    - Invocados a partir del objeto antes creado
  - Métodos estáticos
    - Atribuidos a una clase
    - Invocados a partir del nombre de clase
      - No requieren la creación previa de un objeto
    - Usados para métodos genéricos (utilidades)

21

3. Tipos de métodos

- Ejemplo de método de instancia
  - Creación del método en una clase

```
class Coche{
   String color;
   …
   void pinta(String nuevoColor){
      color = nuevoColor;
    }
}
```

  - Invocación del método con un objeto

```
Coche miCoche = new Coche();
miCoche.pinta("azul");
```

22

3. Tipos de métodos

- Ejemplo de método estático
  - Creación del método en una clase

```
class Mates{
    static int potencia (int base, int exp){
        int pot = 1;
        for(int i=0; i<exp; i++) pot *= base;
        return pot;
    }
}
```

  - Invocación del método a partir de la clase

```
int resultado = Mates.potencia(4, 5);
```

23

4. Paso de parámetros

24

4. Paso de parámetros

- Paso por valor
  - El método recibe copia de los argumentos
  - Cambio de valor de parámetro en método
    - Sólo afecta al método
    - Sin efecto fuera del método
  - Los tipos básicos se pasan por valor

25

4. Paso de parámetros

- Paso por referencia
  - El método recibe referencia como parámetro
    - Dirección de memoria
  - Cambio de valor de parámetro en el método
    - Afecta al método
    - Cambio también fuera del método
  - Los objetos se pasan por referencia

26

5. Ámbito de variables

27

5. Ámbito de variables

- Ámbito de la variable
  - Zona de código en que puede ser accedida
  - Bloque de código en que se ha declarado
- Tiempo de vida de la variable
  - Tiempo entre su declaración y su destrucción
  - Suele coincidir con el ámbito

28

5. Ámbito de variables

- Elementos (variables) locales
  - Declarados dentro de un método
  - Accesibles sólo desde dicho método
    - Desde su declaración
    - Hasta la terminación del bloque en el que se declara
- Elementos globales (atributos)
  - Declarados en la clase, fuera de todo método
  - Accesibles desde cualquier método de la clase
    - Desde cualquier punto del método

29

6. El método main

30

6. El método main

- Cabecera
  - `public static void main(String[] args)`
    - `public` : método visible desde el exterior
    - `static` : método con alcance de clase
    - `void` : no devuelve nada
    - `main` : nombre del método
    - `String[] args` : lista de argumentos del programa
- Punto de entrada del programa Java
  - Toma el control al ejecutarse el programa
  - El programa termina con dicho método

31

7. Recursividad

32

7. Recursividad

- Java admite métodos recursivos
  - Se invocan a sí mismos en el cuerpo
- Definición de la solución
  - Parte definida recursivamente
    - En términos de sí misma
    - Invocación del propio método
  - Parte no recursiva
    - Imprescindible condición de finalización
- Ejemplo
  - Cálculo del factorial

33

7. Recursividad

```
int factorial(int numero){
   int fact;

   if(numero == 0)
      fact = 1;
   else
      fact = numero * factorial(numero - 1);

   return fact;
}
```

34

8. Sobrecarga

35

8. Sobrecarga

- Análogo a la sobrecarga de operadores
  - + : suma de enteros, suma de reales, concatenación
- Métodos con el mismo nombre
- Normalmente funcionalidad análoga
- Diferenciados por sus parámetros
  - Número distinto de parámetros
  - Tipos de datos diferentes
- Se invoca uno u otro según
  - Número de parámetros pasados
  - Tipo de datos de los parámetros pasados

36

8. Sobrecarga

- `void muestra()`
  - Mostrar un mensaje por defecto
- `void muestra(int veces)`
  - Mostrar un mensaje por defecto un número de veces
- `void muestra(String mensaje)`
  - Mostrar un mensaje indicado
- `void muestra(String mensaje, int veces)`
  - Mostrar un mensaje indicado un número de veces

37

Unidad 4
Programación modular: los métodos

Programación
1º D.A.M.

38