Programación orientada a objetos con JAVA

Definición de arreglos

```
type variableName[];
o
type[] variableName;
```

Asignación de dimensión

```
variableName = new type[n]; // indica el tamaño de n, no crea objetos.
```

```
* variableName.length indica n (la dimensión del arreglo).
```

Ejemplo:

```
int notas[];
```

```
Dado dados[] = new Dado[6];
```

```
dados.length tiene el valor 6.
```

Definición de arreglos multidimencionales

```
type variableName[][]...;
```

```
variableName = new type[n][m]...;
```

```
* variableName[n].length indica m (la dimensión del arreglo n).
```

```
int matriz[][] = new int[10][5];
```

```
matriz[2].length tiene el valor 5.
```

```
String tridim[][][] = new int[10][5][3];
```

Herencia

```
class ClassName extends SuperClassName {
}
```

```
class ClassName {
}
```

es equivalente a

```
class ClassName extends Object {
}
```

Ejemplo:

```
class Alumno extends Persona {
   . . .
}
```

Clase abstracta

```
abstract class ClassName [extends SuperClassName] {
   . . .
}
```

Una clase abstracta no puede ser instanciada.

Ejemplo:

```
abstract class Persona {
   . . .
}

class Alumno extends Persona {
   . . .
}
```

Método abstracto

```
abstract type methodName([type paramName,type paramName,...]);
```

Ejemplo:

```
abstract int getCantidad();
```

Si una clase tiene por lo menos un método abstracto, la clase debe declararse abstracta.
El método abstracto debe definirse en la subclase o la subclase debe declararse abstracta.

Clase final

```
final class ClassName [extends SuperClassName] {
}
```

Una clase declarada final no puede ser superclase de otra clase.

Ejemplo:

```
final class Alumno extends Persona {
   . . .
}
```

Método final

```
final type methodName([type paramName,type paramName,...]) {
   . . .
   return type
}
```

Ejemplo:

```
final int getCantidad() {
   return cantidad;
}
```

Un método declarado final no puede ser sobreescrito por la subclase.

Variable final

```
final type VARIABLENAME = valor;
```

- el nombre de la variable se escribe todo en mayúsculas.

Ejemplo:

```
final int CANTALUMNOS = 20;
```

Uso de paquetes

```
import paquete.ClassName;
o
import paquete.*;

Ejemplo:

import figuras.Triangulo;
class Graficar {
   Triangulo t;
   . . .
}
```

Control de acceso a variables miembro

```
[tipo_acceso] type variableName;
```

Ejemplo:

```
private int cantidad;
```

Control de acceso a constructores

```
[tipo_acceso] ClassName([type paramName,type paramName,...]) {
    . . .
}
```

Ejemplo:

```
public Alumno (String n) {
   nombre = nombre;
}
```

Control de acceso a métodos

```
[tipo_acceso] type methodName([type paramName,type paramName,...]) {
   . . .
   return type
}
```

Ejemplo:

```
protected int getCantidad() {
   return cantidad;
}
```

Tipos de acceso:

private: solo se puede acceder / invocar desde la misma clase.
protected: las subclases los pueden acceder / invocar.
package: (no se escribe nada) solo pueden ser accedidos / invocados por clases que integran el mismo paquete.
public: pueden ser accedidos / invocados desde cualquier clase.