Herencia

Existen clases generales y unas clases más especificas, las cuales comparten características, a las clases especializadas se las conoce como clase hijo las cuales heredan de la clase general (clase padre) en la cual se encuentran las características comunes.

Una clase hijo puede obtener todos los métodos y/o atributos (aún si nos son accesibles) de la clase padre como si fueran suyas, lo cual no ocurre con la clase padre, esta clase no puede obtener ningún método y/o atributo clase de hijo.

Una clase hijo hereda de una clase padre usando la palabra clave **extends**:

class Hijo **extends** Padre {

//métodos y atributos

}

Sólo puede existir una superclase Padre y esta puede tener varias subclases Hijos

Estas pueden ser a su vez padres (“cadena de herencia” , se recomienda que sea menor a 5) ; a esta jerarquía forma un árbol.

Palabras clave

El acceso a los métodos y/o atributos se declara con los siguientes modificadores:

- ✓ **Private:** solo pueden acceder al método o atributo si se encuentran dentro de la misma clase (encapsular), esto hace que se necesiten adicionar métodos de acceso a la clase padre para que acceda a los métodos de esta clase indirectamente.
- ✓ **Protected:** permite que una subclase incluso de un paquete diferente acceda a su superclase.
- ✓ **Public:** permite el acceso a todas las clases, subclases y superclases.
- ✓ **Default:** permite el acceso a las subclases dentro de un mismo paquete, pero si no pertenecen a este se comportan como private.

Constructores

A diferencia de los métodos y atributos, los constructores no se heredan, y no pueden ser redefinidos, para ser usados deben llamarse desde cada clase hijo o padre perteneciente a la jerarquía, para llamar a los constructores se usa la palabra clave **super** :

super (lista de argumentos**);** // Esta lista se usa para seleccionar un //constructor en particular, si se deja vacía llama al constructor de la superclase.

Cuando se crea un objeto se realizan llamas a constructores e inicializaciones, lo cual es un proceso automático en el que solo se puede controlar qué método de la clase padre es invocado utilizando la palabra clave **super**

Polimorfismo

El polimorfismo consiste en obtener un mismo método con diferentes funciones.

Las clases hijos añaden atributos propios, pero a su vez heredan atributos y métodos de la clase padre; si un método es declarado en la clase padre y vuelve a ser declarado en las clases hijos este sobreescribe a la clase padre, para que este método se muestre en las clases hijos es necesario que sea llamado mediante la sentencia super.

Creación de hojas de cálculo en Excel

Para la creación de hojas de cálculo debemos seguir estos pasos:

```
Workbook wb = new HSSFWorkbook(); // Creación de libros

Sheet hoja1 = wb.createSheet(); // Creación de hojas
FileOutputStream out = null;


Row row1 = hoja1.createRow(1); // Creación de columnas

Cell cell1_0 = row1.createCell(0); // Creación de celdas

cell1_0.setCellValue("CARACTERISTICA"); // Asignar valores a la celda

Sheet sheet = wb.getSheetAt(0); // Modifica el tamaño de la celda
sheet.autoSizeColumn(0);
```

```
  try{   // Se utiliza para introducir un código que puede provocar errores
        out = new FileOutputStream("Aeropuerto.xls"); // Crea el archivo excel
        wb.write(out);
    }catch(IOException e){ // Código a ejecutarse si ocurriese el error.
        System.out.println(e.toString());
    }finally{
        try {
            out.close();
        }catch(IOException e){
            System.out.println(e.toString());
        }
    }
}
```

*Las sentencias catch deben asignarse en orden de jerarquía de menor a mayor, porque estas se ejecutan en orden, es decir que si tenemos un error menor y al escribir las sentencias catch pusimos error más general, entonces ya no ejecutará la sentencia que le correspondía sino la primera que encontró (general).