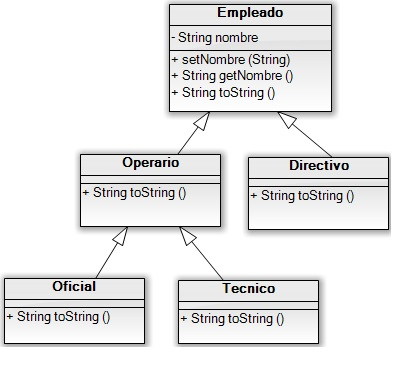
**HERENCIA Y POLIMOSFISMO**

1. Implementa la siguiente jerarquía de clases Java representada por el siguiente diagrama UML:

Un empleado, sea de la clase que sea, solo contiene el atributo nombre, de tipo string y acceso privado, y un constructor que recibe el nombre del empleado.

Dependiendo del tipo del empleado, el método toString() debe mostrar:

Empleado Juan Si Juan es un empleado raso

Empleado Pepe – Directivo Si Pepe es directivo

Empleado Pepe – Operario Si Pepe es operario

Etc.

1. Nos piden realizar una agenda telefónica de contactos.

Un contacto está definido por un nombre y un teléfono (No es necesario de validar). Un contacto es igual a otro cuando sus nombres son iguales.

Una agenda de contactos está formada por un conjunto de contactos.

Se podrá crear de dos formas, indicándoles nosotros el tamaño o con un tamaño por defecto (10)

Los métodos de la agenda serán los siguientes:

* **aniadirContacto(Contacto c)**: Añade un contacto a la agenda, sino se pueden meter más a la agenda se indicara por pantalla. No se pueden meter contactos que existan, es decir, no podemos duplicar nombres, aunque tengan distinto teléfono.
* **existeContacto(Conctacto c)**: indica si el contacto pasado existe o no.
* **listarContactos()**: Lista toda la agenda
* **buscaContacto(String nombre)**: busca un contacto por su nombre y muestra su teléfono.
* **eliminarContacto(Contacto c)**: elimina el contacto de la agenda, indica si se ha eliminado o no por pantalla
* **agendaLlena()**: indica si la agenda está llena.
* **huecosLibres()**: indica cuantos contactos más podemos meter.

Crea un menú con opciones por consola para probar todas estas funcionalidades.

1. Diseña una clase abstracta denominada Figura, cuyas características sean las siguientes:

**Atributos**  **Significado**

Base Base de la figura (double)

Altura Altura de la figura (double)

**Métodos** **Significado**

setBase(double) Asigna un valor a la base de la figura.

setAltura(double) Asigna un valor a la altura de la figura.

double getBase() Devuelve el valor de la base del objeto sobre el que se aplica

double getAltura() Devuelve el valor de la altura del objeto sobre el que se aplica.

double area() Método que calcula el área de la figura

Figura contiene dos subclases denominadas **Rectangulo** y **Triangulo**. Programad estas dos clases que carecerán de atributos y métodos propios y que sólo sobrescribirán el método area (heredado de Figura). Estas dos clases también deben tener, al menos, un constructor que permita crear objetos Triángulo y Rectángulo respectivamente.

Escribid una clase de prueba que debe crear dos triángulos y dos rectángulos y ejecutar todos los métodos programados mostrando el área de cada una de los objetos y las dimensiones (base y altura) de cada una de ellos.

1. Diseña una clase denominada **Vehículo**. La información a guardar de un vehículo es: número de ruedas y autonomía. Además del constructor y los métodos set y get para cada uno de los atributos de la clase, tendremos un método denominado **verDatos** que muestre en pantalla los datos de un vehículo. La clase Vehículo tiene dos subclases: **Coche** y **Camión**. Un coche además de la información de un vehículo almacena el número de pasajeros. Un camión tendrá además el límite de carga. Cada una de las subclases definirán también un método denominado verDatos que permita ver en pantalla todos los atributos de coches y camiones respectivamente. Cread un programa que pruebe el funcionamiento de las clases creadas.