1 Queremos hacer una aplicación para profesores y alumnos. De ambos se quiere conocer el nombre, la edad, el sexo y la nacionalidad. De los profesores se quiere conocer además el sueldo, y del alumno si es o no repetidor. Realiza los siguientes apartados creando tres paquetes diferentes, uno para apartado (1.1, 1.2.1 y 1.2.2). Cada paquete debe contener las clases y la clase con el main.

1.1 Realiza en java las clases que consideres oportunas teniendo en cuenta las relaciones de herencia. Realiza un constructor por defecto para todas las clases sin usar super(). Realiza también los toString para todas las clases. Haz un programa para probarlo que haga lo siguiente:

- Crea un profesor y saca por consola su toString.
- Crea un alumno y saca por consola su toString.
- Crea una variable del padre y según un valor solicitado por consola, introduce en ella un objeto profesor o alumno. Muestra por consola el toString.

1.2 Hacer otra versión sustituyendo el constructor por defecto del padre por otro con parámetros. Soluciona los errores que surjan de dos maneras posibles:

1.2.1 Realizar una llamada explícita a un constructor de la superclase.

1.2.2 Utilizando el this(parámetros).

2 Rellena las siguientes tablas. Para ello, realiza ejemplos para probar cada caso.

Tabla de visibilidad

	private	**friendly**	**protected**	**public**
Misma clase				
Clase en el mismo paquete				
Clase en otro paquete				
Subclase en el mismo paquete				
Subclase en distinto paquete				

Tabla de herencia

	private	**friendly**	**protected**	**public**
Subclase en el mismo paquete				
Subclase en distinto paquete				

3 Queremos colocar figuras geométricas en un plano bidimensional y además calcular el área de dichas figuras. Para ello, crear la clase abstracta FiguraGeométrica que tendrá como método abstracto calcular_area. Dicha clase iniciará los valores de las coordenadas *x* e *y* del plano bidimensional para indicar la posición de la figura en dicho plano. Para el círculo será el centro de la circunferencia y para el resto, el vértice inferior izquierdo. Crear las clases Rectángulo, Círculo y Triángulo heredadas de la clase FiguraGeométrica.

4 Crea una clase *empleado* y una subclase *encargado*. Los encargados reciben un 10% más de sueldo base que un empleado normal aunque realicen el mismo trabajo. Implementa dichas clases con el método calcularSueldo() para ambas clases.

5 ¿Por qué no compila el siguiente código?

```
class Prueba{
  protected String nombre;
  protected int id;
  public String getIdent() {return nombre;}
  public int getIdent() {return id;}
}
```

6 Implementa la siguiente estructura de clases:
forma es una superclase que tiene cuatro subclases: círculo, cuadrado, triángulo y rombo. Ni forma ni las subclases contienen atributos. La clase forma es abstracta y contiene el método abstracto toString(). Las subclases, al no tener atributos, en el método toString informarán del tipo de forma que son (círculo, etc).
La clase forma tendrá implementado un método identidad que devuelva una cadena con la subclase a la que pertenece. Realizar el método identidad de tres maneras:

6.1 Con el método getClass(). Investigar en la API sobre este método.
6.2 Con instanceof
6.3 Sin usar getClass() ni instanceof

Hacer un programa que cree un array con cuatro objetos, uno de cada subclase y ejecutar de todos el método identidad.

7 Este código está utilizando la estructura de clases del ejercicio anterior. Modifica la sintaxis de las líneas que dan problema y elimina aquellas líneas que aunque sean sintácticamente correctas nunca pueden funcionar.

```
class Testforma {
public static void main(String[] args) {
            Forma f=new Circulo();
            f.identidad();
            Circulo c=new Circulo();
            ((Forma)c).identidad();
            ((Circulo)f).identidad();
            Forma f2=new Forma();
            f2.identidad();
            (Forma)f.identidad();
            f=c;
            c=f;
      }
}
```

8 Averigua los errores del siguiente código:

```
public class Test {
	public int dato=0;
	public static int datostatico=0;
	public void metodo() {this.datostatico++;}
	public static void metodostatico(){
		this.datostatico++;
		datostatico++;
	}
	public static void main(String [] args){
		dato++;
		datostatico++;
		metodostatico();
		metodo();
	}
}
```

9 ¿Qué mostrará el siguiente programa por pantalla?

```
public class Bebe {
	Bebe(int i){
		this("Soy un bebe consentido");
		System.out.println("Hola, tengo " + i + " meses");
	}
	Bebe(String s){
		System.out.println(s);
	}
	void berrea(){
		System.out.println("Buaaaaaaaaa");
	}
	public static void main(String[] args){
		new Bebe(8).berrea();
	}
}
```

10 Averigua sin ejecutar el código, qué mostrará el siguiente programa por pantalla. Una vez que tengas claro lo que el programa debería de mostrar por pantalla ejecuta el código y verifica que lo que has pensado se cumple.

```
public class Bebe {
	static void pedir(){
		System.out.println(str1 + " , " + str2 + " , " + str3);
	}
	static {
		str2 = "mama pipi";
		str3 = "mama agua";
	}
	Bebe() {System.out.println("Nacimiento del bebe"); }
	static String str2,str3,str1 = "papa tengo caca";
	public static void main(String[] args) {
		System.out.println("El bebe se ha despertado y va a pedir cosas");
		System.out.println("El bebe dice: " + Bebe.str1);
		Bebe.pedir();
	}
	static Bebe bebe1 = new Bebe();
	static Bebe bebe2 = new Bebe();
```

```
    static Bebe bebe3 = new Bebe();
}
```

11 Tenemos la siguiente clase:

```
public abstract class Sorteo {
      protected int posibilidades;
      public abstract int lanzar();
}
```

Se pide:

- Crear la clase D*ado*, la cual desciende de la clase S*orteo*. La clase D*ado,* en la llamada al método lanzar devolverá un número aleatorio del 1 al 6.
- Crear la clase M*oneda,* la cual desciende de la clase S*orteo*. Esta clase en la llamada al método *lanzar* devolverá las palabras cara o cruz.

12 Averigua por qué el compilador da un mensaje de error en el siguiente código:

```
class Testfinal {
public static void main(String[] args) {
      final String s1=new String("Hola");
      String s2=new String(" Mundo");
      s1=s1+s2;
      }
}
```

13 Tenemos la siguiente clase:

```
public abstract class Vehiculo{
      private int peso;
      public final void setPeso(int p){peso=p;}
      public abstract int getVelocidadActual();
}
```

- ¿podrá tener descendencia esta clase?
- ¿se pueden sobrescribir todos sus métodos?

14 Tenemos una jardinería donde se venden plantas de jardín y productos de alfarería. Ambas disponen de atributos precio y descripción pero no tienen relación de herencia. Además, las plantas disponen de un atributo propio para saber si está regada, y los productos de alfarería tienen otro atributo para indicar si el producto es frágil. Ambas implementan la interfaz Mercancía:

```
interface Mercancia{
    public double damePrecio();
    public String dameDescripcion();
}
```

Haz un programa para probarlo que contenga el siguiente método estático:
public static void dameDatos(Mercancia producto)
Dicho método deberá mostrar el precio y la descripción del producto.

15 Implementa los siguientes interfaces que heredan de la interface Mercancía del ejercicio anterior:

```
interface MercanciaViva extends Mercancia{
      public boolean necesitaComida();
      public boolean necesitaRiego();
}
```

```
interface MercanciaFragil extends Mercancia{
        public String dameEmbalaje();
        public double damePeso();
}
```

Las plantas implementan la interface MercancíaViva y los productos de alfarería implementan MercancíaFragil. Haz un programa para probarlo que contenga el siguiente método estático: public static void dameDatos(Mercancia producto). Dicho método deberá utilizar los métodos de Mercancia, MercanciaViva y MercanciaFragil.

16 Realiza una clase Pez con los atributos nombre, especie y zona donde vive. Realiza una subclase llamada PezAguaDulce con un atributo booleano para saber si es un pez de acuario. Realiza para ambas clases el toString y el equals. Haz un programa para probarlos.

17 Para la clase pez anterior, crea un atributo privado entero numpeces común a todos los objetos pez el cual cuente el número de peces creados. Crea un programa que compruebe que esta variable se incrementa cada vez que se crea un objeto pez.

18 Realiza una clase Huevo que esté compuesta por dos clases internas, una clara y otra yema. Realiza un programa para probarlo.

19 ¿Qué resultado da el siguiente código? Analiza qué tipo de clase interna se está utilizando y haz una reflexión del resultado.

```
public class VerClaseInterna{
	public static void main(String args[]){
		Contenedor c1 = new Contenedor(34);
		Contenedor.Contenido i1 = c1.new Contenido(23);
		System.out.println(c1.muestraContenedor(i1));
		c1.numero=50;
		System.out.println(i1.muestraContenido());
		i1.numero2=25;
		System.out.println(c1.muestraContenedor(i1));
		i1.numero2=65;
		System.out.println(i1.muestraContenido());
	}
}
public class Contenedor{
	public int numero=0;
	public Contenedor(int numero){
		this.numero=numero;
	}
	public String muestraContenedor(Contenido refCont){
		return "N. contenedor= "+numero+" N. contenido= "+refCont.numero2;
	}
	public class Contenido{
		public int numero2;
		public Contenido(int numero2){
			this.numero2=numero2;
		}
		public String muestraContenido(){
			return "N.contenedor= "+numero+" N.contenido= "+numero2;
		}
	}
}
```

20 ¿Qué resultado da el siguiente código? Analiza qué tipo de clase interna se está utilizando y haz una reflexión del resultado.

```
public class ClaseLocal {
	public int numero=0;
	public ClaseLocal(int numero){
		this.numero=numero;
	}
	public String muestraContenido(){
		class Mostrador{
			public String muestraDato(){
				return "Número = "+numero;
			}
		}
		Mostrador m = new Mostrador();
		return m.muestraDato();
	}
}
public class VerClaseLocal{
	public static void main(String args[]){
		ClaseLocal c1 = new ClaseLocal(346);
		System.out.println(c1.muestraContenido());
	}
}
```

21 ¿Qué resultado da el siguiente código? Analiza qué tipo de clase interna se está utilizando y haz una reflexión del resultado.

```
public class VerClaseAnidada{
	public static void main(Strings args[]){
		PrimerContenedor.Contenido i1=new PrimerContenedor.Contenido(29);
		i1.numero2=25;
		System.out.println(i1.muestraContenido());
		PrimerContenedor c1 = new PrimerContenedor(34);
		System.out.println(c1.muestraContenedor(i1));
	}
}

public class PrimerContenedor{
	public int numero=0;
	static public int numero3=13;
	public PrimerContenedor(int numero){
		this.numero=numero;
	}
	public String muestraContenedor(Contenido refCont){
		return "Nº contenedor= "+numero+"Nº contenido= "+refCont.numero2;
	}
	static class Contenido{
		public int numero2;
		public Contenido(int numero2){
			this.numero2=numero2;
		}
		public String muestraContenido(){
			return "Nº contenedor= "+numero3+" Nº contenido= "+numero2;
		}
	}
}
```