```
//-------------------------------------------------------------------------
//------------------------- Herencia de interfaz/implementación ----------------
//-------------------------------------------------------------------------
// Forma.java
public abstract class Forma {

    int x; // de instancia
    int y; // de instancia

    static int ESTATICO=0; //de clase

    Forma(int a, int b) { this.x=a; this.y=b; } // de instancia

    public int getX() { return this.x; } // de instancia, enlace dinámico
    public int getY() { return this.y; } // de instancia, enlace dinámico

    public static int getESTATICO() { return  ESTATICO; } // de clase, enlace  estático

    public abstract void pintar(); // de instancia, enlace dinámico

    // public abstract static void repintar() ; // enlace estático y dinámico ?!?!?

    public abstract String toString();

}

//Circulo.java
public class Circulo extends Forma {

    private int radio;

    public Circulo(int a, int b, int r) {
        // Constructor refinado
        super(a, b);
        this.radio=r;
    }

    @Override
    public void pintar() {
        System.out.println("CIRCULO: ("+x+", "+y+", "+radio+")");
    }

    public String toString() { return "("+x+","+y+","+radio+")"; }

    public static int getESTATICO() { return  ESTATICO+1; }


}

// Triangulo.java
public class Triangulo extends Forma {

    int lado1, lado2, lado3;

    public Triangulo(int a, int b, int l1, int l2, int l3) {
        super(a, b);
        lado1=l1;
        lado2=l2;
        lado3=l3;
    }

    //@Override
    public void pintar() {

        System.out.println("TRIANGULO: ("+x+", "+y+", <"+lado1+","+lado2+","+lado3+">)");
    }

    public String toString() { return "<"+lado1+","+lado2+","+lado3+">"; }


}
```

```
// Isosceles.java
public class Isosceles extends Triangulo {

    public Isosceles(int a, int b, int l1, int l2) {
        super(a, b, l1, l2, l2);
    }

    public final int getX() { return -1; }

}

// Main.java
import java.util.ArrayList;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        List<Forma> formas = new ArrayList<>();

        formas.add(new Circulo(2,3,4));
        formas.add(new Circulo(3,4,5));
        formas.add(new Triangulo(6,7,8,9,10));
        formas.add(new Isosceles(16,17,18,19));

        pintor(formas);

    }

    private static void pintor(List<Forma> formas) {
        for (Forma f : formas) {
            f.pintar();
        }

        for (Forma f : formas) {
            System.out.print(f.getESTATICO());
            System.out.print(" - ");
            System.out.println(f.getX());
        }

        formas.get(0).ESTATICO = 10;

        for (Forma f : formas) {
            System.out.print(((Circulo)f).getESTATICO());
            System.out.print(" - ");
            System.out.println(f.getX());
        }
    }

}


//-----------------------------------------------------------------------------
//-------------------------------- CallingSuper.java ------------------------
//-----------------------------------------------------------------------------

class A {
    private void f(int x) { x=1; }
    protected void g(int y) { y=2; }
    public void h(int z) { z=3;}
}

class B extends A {
    // super.f() not visible here.
    //private void f(int x) { super.f(x); }
    public  void f(int x) { super.g(x); }
    protected void g(int y) { super.g(y); }
    public void h(int z) { super.h(z); }
}

public class CallingSuper {
```

```
    /**
     * Tests calling a method with identical signature in the base class
     * from the same method in the subclass
     * @param args nothing
     */
    public static void main(String[] args) {
        B obj = new B();
        obj.f(3);
        obj.g(4);
        obj.h(5);
    }

}

//-------------------------------------------------------------------------
//------------ EJERCICIO WEB Pila de enteros (excepciones) ------------
//-------------------------------------------------------------------------
public class PilaEnteros {
    private int maximo; // Tamaño máximo de la pila
    private int[] pila; // la pila
    private int cima; //número de elementos actualmente en la pila-1

    public PilaEnteros(int max) {
        if (max>=0) maximo = max; else maximo = 0;
        pila = new int[maximo];
        cima = -1;
    }

    void apilar (int elem)  {
        System.out.println("Llamo método apilar");
            if (cima  < maximo-1) {
                pila[++cima]= elem;
                System.out.println("Apilo");
            } // sino no apila nada ni notifica a nadie
    }

    public static final void main(String args[]){
        PilaEnteros pi = new PilaEnteros(4);
        for (int i=0;i<20;i++)
            pi.apilar(i);
    }

}

//-------------------------------------------------------------------------
//------------ HERENCIA DE IMPLEMENTACION: Cuenta/CuentaJoven  ------------
//-------------------------------------------------------------------------
public class Cuenta{
      private String titular;
      private double saldo;
      protected double interes;

      private static int numCuentas;

      public Cuenta()
      { titular=""; saldo=0.0; interes=0.0; numCuentas++; }

      public Cuenta(String t, double s, double i)
       { titular=t; saldo=s; interes=i; numCuentas++; }

      public Cuenta(Cuenta tc)
      { titular=tc.titular; saldo=tc.saldo; interes=tc.interes; numCuentas++; }

      protected void finalize() throws Throwable { numCuentas--; }

      void abonarInteresMensual()
      { setSaldo(getSaldo()*(1+getInteres()/100/12)); }

      public String toString ()
      {
        return "NumCuentas=" + Cuenta.numCuentas + "\n"
          + "Titular=" + titular + "\n"
```

```
            + "Saldo=" + saldo + "\n"
            + "Interes=" + interes + "\n";
      }

    public String getTitular() { return titular; }
    public void setTitular(String titular) { this.titular = titular; }
    public double getSaldo() { return saldo; }
    public void setSaldo(double saldo) { this.saldo = saldo; }
    public double getInteres() { return interes; }
    public void setInteres(double interes) { this.interes = interes; }

    public static void visualiza(Cuenta c) { System.out.println(c); }
}

// CuentaJoven.java

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class CuentaJoven extends Cuenta {

    private int edad;


    public CuentaJoven(String unNombre,int unaEdad,
            double unSaldo, double unInteres) {
        super(unNombre,unSaldo,unInteres);
        edad=unaEdad;
    }

    public CuentaJoven(CuentaJoven tcj)
    // llamada explícita a constructor de copia de Cuenta.
    {
        super(tcj);
        edad=tcj.edad;
    }

    @Override
    void abonarInteresMensual() {
        //no interés si el saldo es inferior al límite
        if (getSaldo()>=10000) super.abonarInteresMensual();
    }

    int getEdad() {return edad;}
    void setEdad(int unaEdad) {edad=unaEdad;}

    @Override
    public String toString(){
        String s = super.toString();
        return s + "Edad:"+edad;
    }

    public static void visualiza(Cuenta c) { Cuenta.visualiza(c); }

    public final static void main(String args[]) {
        CuentaJoven cj = new CuentaJoven("PEPE", 20, 0, 0);
        Cuenta c = cj;

        List<Cuenta> l = new LinkedList<>();
        l.add(c);

        //c = new Cuenta("JUAN",0,0);

        c.abonarInteresMensual();

        cj = (CuentaJoven)c;
        cj.setEdad(20);

        visualiza(c);
        visualiza(cj);
    }
```

```
}//fin clase CuentaJoven

// Banco.java
public class Banco {

    private List<Cuenta> cuentas;
    public Banco() { cuentas = new ArrayList<Cuenta>(); }

    private void addCuenta(Cuenta c) {
        if (c!= null) cuentas.add(c);
    }

    public void nuevaCuenta(String tit, double s, int edad) {
        if (edad<21) addCuenta(new CuentaJoven(tit,edad,s,0.02));
        else addCuenta(new Cuenta(tit,s,0.01));
    }

    public void eliminaCuenta(String tit) {
        for (Cuenta c : cuentas) {
            if (c.getTitular().equals(tit)) {
                cuentas.remove(c);
                break;
            }
        }
    }

    public void abonarInteresMensual() {
        for (Cuenta c: cuentas) { c.abonarInteresMensual(); }
    }
}

//-------------------------------------------------------------------------------
//---------------- HERENCIA DE INTERFAZ: Pila/ArrayPila/PilaDeEnteros -----------
//-------------------------------------------------------------------------------
// Pila.java
public interface Pila {
    void apila(Object o);
    Object desapila();
    Object tope();
    int size();
}

// ArrayPila.java
public class ArrayPila implements Pila {

    public static final int MAX_PILA=10;
    private Object[] pila;
    private int tope;

    public ArrayPila() {
        pila = new Object[MAX_PILA];
        tope=-1;
    }

    @Override
    public void apila(Object o) {
            if (tope<MAX_PILA) pila[++tope] = o;
            else throw new IndexOutOfBoundsException("Pila llena");
    }

    @Override
    public Object desapila() {
        if (tope!= -1) return pila[tope--];
        else return null;
    }

    @Override
    public int size() { return tope+1; }

    @Override
    public Object tope() {
```

```
        // Sin copia defensiva es fácil:
        if (tope!=-1) return pila[tope];
        else return null;

        /* Copia defensiva de un objeto que no sabemos de qué clase es
         * (bueno, sí, es un Object). Si la clase (en tiempo de ejecución) del objeto no
         * tiene constructor de copia, obtendremos una 'NoSuchMethodException'
        Object ret=null;
        if (tope!=-1) {
            Class<?> clase = pila[tope].getClass();
            Class<?>[] paramTypes = new Class[] { clase };
            Constructor<?> ctor;
            try {
                ctor = clase.getConstructor( paramTypes );
                Object[] arguments = new Object[] { pila[tope] };
                ret = ctor.newInstance(arguments);
            } catch (NoSuchMethodException | SecurityException | InstantiationException
                    | IllegalAccessException | IllegalArgumentException
                    | InvocationTargetException e) {
                e.printStackTrace();
            }
        }
        return ret;
        */
    }
}

// PilaDeEnteros.java
public class PilaDeEnteros extends ArrayPila {

    public PilaDeEnteros() { super(); }

    // No @Override !!
    public void apila(Integer o) { super.apila(o); }

    // @Override
    public void apila(Object o) {
        if (o instanceof Integer) { super.apila(o); }
        else throw new IllegalArgumentException("Solo se pueden apilar enteros");
    }

    @Override
    public Integer desapila() { return (Integer) super.desapila(); }

    @Override
    public Integer tope() {
        // Copia defensiva de un objeto que SÍ sabemos de qué clase es: Integer
        return new Integer( (Integer)super.tope() );

        // ¡¡en realidad java.lang.Integer no tiene constructor de copia!! ¿¿¿???
    }

    public static final void main(String args[]) {
        PilaDeEnteros pint = new PilaDeEnteros();

        for (int i=1; i<10; i++)
            pint.apila(i);

        while (pint.size() > 0) {
            System.out.println("Tope: "+pint.tope());
            System.out.println("Desapilo: "+ pint.desapila());
        }
    }
}
```