

EXAMEN PDOO : TEORÍA 1 ~ [2019 - 2020]

TIPO DE EXAMEN : 1

Pregunta 1 - [7 puntos]: Dados los siguientes ficheros

```
//FICHERO HIJA.JAVA
package paqueteA;

class Padre{
    protected void protegido(){
        System.out.println("Protegido Padre");
    }
    void metodo(){
        System.out.println("Metodo Padre");
    }
    public void procesa(){
        System.out.println("Procesando en el padre...");
    }
    public void ejecutarTarea(){
        procesa();
        System.out.println("Fin de la tarea en el padre");
    }
}

public class Hija extends Padre{
    void test(Padre p){
        p.protegido();
        p.metodo();
    }
}
```

```
//FICHERO NIETA.JAVA
package subpaquete.paqueteA;
import paqueteA.Hija;

public class Nieta extends Hija{
    void test(Hija p){
        p.protegido();
        p.metodo();
    }
    void test2(Nieta p){
        p.protegido();
        p.metodo();
    }
    @Override
    public void procesa(){
        System.out.println("Procesando en el nieto...");
    }
    public void tareaNieto(){
        System.out.println("Tarea en el nieto");
    }
}
```

```

//FICHERO PRINCIPAL.JAVA
import java.util.ArrayList;
import paqueteA.Hija;
import subpaquete.paqueteA.Nieta;

interface MyInterface{
    public void ejecutarTarea();
}

public class Principal{
    public static void main(String[] args){
        Nieta n=new Nieta();
        n.ejecutarTarea();
        ((Hija) n).ejecutarTarea();

        Hija h=new Nieta();
        MyInterface interf=h;

        h.tareaNieta();

        ArrayList<Integer> array = (ArrayList<Integer>) (Object) h;
        ArrayList<Hija> array2 = new ArrayList<Nieta>();
    }
}

```

Los métodos privados: `ajuste_del_atributo_a`, `calculo_factor`, `ajuste_factor`, y `ajuste_del_atributo_b` ya se encuentran implementados y este ejercicio no requiere conocer dicha implementación.

1.- Crear otro constructor que tenga 3 parámetros (a, b, filtro) que se deben proporcionar siempre obligatoriamente. Los dos primeros son los mismos que los del constructor existente y el otro es un valor utilizado para una operación de filtrado del parámetro a. Así, este constructor funcionará igual que el proporcionado, pero como primer paso, se llama a un método llamado `filtra_a(a,filtro)` que utiliza el primer y tercer parámetro del nuevo constructor y que devuelve el valor de a filtrado. Ese valor filtrado es el que se usa como valor de a para el resto de operaciones requeridas en la construcción del objeto.

2.- Se debe además escribir la cabecera completa del método `filtra_a(a,filtro)`. El cuerpo se dejará vacío.

La solución aportada no puede depender de cambios en el código existente.

Pregunta 2 - [7 puntos]: Se proporciona el siguiente código:

```
class Examen1
  def initialize(a=44,b==33)
    if ((a!=44)&&(b!=33))
      @a=ajuste_del_atributo_a(a)
      factor=calculo_factor(a)
      if (factor<1)
        factor=ajuste_factor(factor)
      end
      @b=ajuste_del_atributo_b(a,b,factor)
    else
      @a=a+1
      @b=b+2
    end
  end

  def salida
    return 2*(@a+@b)
  end

  #.....
end
```

Los métodos privados: `ajuste_del_atributo_a`, `calculo_factor`, `ajuste_factor`, y `ajuste_del_atributo_b` ya se encuentran implementados y este ejercicio no requiere conocer dicha implementación.

Además, ésta es la implementación completa de otra clase:

```
class Examen1_Hija<Examen1
  def initialize(c)
    @c=ajuste_del_atributo_c(c)
  end

  #.....
end
```

1.- Indicar la salida de la siguiente sentencia:

```
puts Examen1_Hija.new(77).salida
```

y justificar la respuesta. Se puede hacer referencia a las salidas de los métodos para los que no se proporciona implementación utilizando valores simbólicos.

Ejemplo: asumo que *valor_c* es la salida del método *ajuste_del_atributo_c(c)*

La solución aportada no puede depender de cambios en el código existente.

Pregunta 3 - [3 puntos]: Se proporciona el siguiente código de dos clases:

```
class Examen1
  def salida(a)
    return 2*Math.sqrt(a)+Math.exp(a+2)
  end
  #.....
end
```

```
class Examen1_Hija<Examen1
  def salida(a)
    #...
  end
  #.....
end
```

1.- Proporciona una implementación para el método *salida* de la segunda clase sabiendo que el resultado debe ser menor en una unidad que el del método del mismo nombre de la primera clase.0

2.- Indicar si en la segunda clase se podría crear adicionalmente otro método llamado *salida* que aceptase dos parámetros. En caso negativo, indicar si podría hacerse en la primera clase. Justificar la respuesta

La solución aportada no puede depender de cambios en el código existente.

Pregunta 4 - [3 puntos]: Dado el siguiente código:

```
19: package Examen;
20:
21: import java.util.ArrayList;
22: import java.util.Arrays;
23: import java.util.Random;
24:
25: class Examen4 {
26:     /*private*/ ArrayList<Integer> contenedor=new ArrayList<>();
27:     private Random generator = new Random();
28:
29:     public void rellena(){
30:         for(int i=0; i<10; i++){
31:             contenedor.add(generator.nextInt(100));
32:         }
33:     }
34:
35:     public ArrayList<Integer> getContenedor() { return contenedor; }
36: }
37:
38: public class Examen3 {
39:     public static void main(String[] args){
40:         Examen4 ex4 = new Examen4();
41:         ex4.rellena();
42:         ex4.getContenedor().add(44);
43:     }
44: }
```

- 1.- Indicar razonadamente si la línea 26 produce un error de compilación.
- 2.- Indicar la respuesta a la pregunta anterior si en la línea 8 dejara de estar comentada la palabra reservada.