EXAMEN PDOO : TEORÍA 1 ~ [2019 - 2020]

TIPO DE EXAMEN : 1

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

```java
//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();
    }
}
```

```java
//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");
    }
}
```

```java
//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.tareaNieto();

        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(generador.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 linea 26 produce un error de compilación.

2.- Indicar la respuesta a la pregunta anterior si en la linea 8 dejara de estar comentada la palabra reservada.

EXAMEN PDOO : TEORÍA 1 ~ [2019 - 2020] (SOLUCIONES)

TIPO DE EXAMEN: 1

A continuación, se muestran las soluciones al examen de `TEORÍA1` de `PDOO`:

Pregunta 1 - [7 puntos] - Apartado 1

*SOLUCIÓN: Se nos pide crear otro constructor para la clase `Examen1` que reciba 3 parámetros que son **(a,b,filtro)** Por lo tanto, la implementación sería la siguiente:*

```
def self.constructor(a,b,filtro)
     #PRIMER PASO: FILTRADO
     a_filtrado = filtra_a(a,filtro)
     #LLAMAMOS AL OTRO CONSTRUCTOR
     new(a_filtrado,b)
end
```

Pregunta 1 - [7 puntos] - Apartado 2

SOLUCIÓN: En este apartado sólo se nos pide definir el método, dejando la implementación vacía puesto que desconocemos el propósito de dicha función. La resolución sería la siguiente:

```
def self.filtra_a(a,filtro)
   #IMPLEMENTATION GOES HERE
end

#ADEMÁS, PODRÍAMOS PONER ESTE MÉTODO PRIVADO DE CLASE, PUESTO QUE SÓLO LO
UTILIZARÍAMOS PARA EL CONSTRUCTOR
method_class_private :filtra_a
```

PREGUNTA 2 - [7 puntos] - Apartado 1

SOLUCIÓN: Tenemos que ver la salida de la sentencia:

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

Se ha redifinido el initialize (con un argumento) de la clase Hija. Clase para la cual se define un atributo de instancia llamado "c".

Por tanto, lo que haría Examen1_Hija.new(77), sería inicializar el atributo de @c a un valor "valor_c" que es obtenido del método ajuste_del_atributo_c(77). Ahora, la clase Examen1_Hija hereda los métodos del padre, pero no sus atributos puesto que son de instancia, y en el constructor de la clase hija no se ha realizado ninguna llamada a super.

Es por lo tanto que esa sentencia produciría un error al ejecutarla.

PREGUNTA 3 - [3 puntos] - Apartado 1

*SOLUCIÓN: Utilizando la herencia de Ruby (sabiendo que `Examen1_Hija` hereda de `Examen1`, la implementación del método `salida(a)` de la clase hija se puede hacer muy sencilla utilizando la llamada a **super**. Por lo tanto:*

```
#METODO SALIDA DE EXAMEN1_HIJA
def salida(a)
    return super(a) - 1
end
```

PREGUNTA 3 - [3 puntos] - Apartado 2

SOLUCIÓN: No se puede hacer en ninguna de las dos clases , ya que en ruby no existe el concepto de sobrecarga (en java sí se podría).

PREGUNTA 4 - [3 puntos] - Apartado 1

En este apartado como en el otro se nos pregunta si hay errores en la sentencia:

```
ex4.getContenedor().add(44);
```

*SOLUCIÓN: Tenemos que tener en cuenta que `Examen4` como `Examen3` están en el mismo paquete `Examen`. En este caso en concreto el atributo **contenedor** tiene visibilidad de **PAQUETE**, por lo que no se produciría ningún error de ningún tipo.*

PREGUNTA 4 - [3 puntos] - Apartado 2

*SOLUCIÓN: La misma situación que la descrita anteriormente sólo que el atributo **contenedor** ahora tiene visibilidad **PRIVADA**. La tentación natural es pensar que se produciría un error puesto que este atributo ahroa pasa a ser privado y por tanto no se puede añadir el número. SIN EMBARGO, tanto en Java como en Ruby, **TODO SON REFERENCIAS**, y cuando se obtiene una referencia a un atributo (con el método `getContenedor()` en este caso), las restricciones impuestas ya no tienen efecto. En conclusión, se ejecutaría correctamente como en el caso anterior.*