**3.14 ANEXO C – CUESTIONES**

**1**. ¿Que es mejor, almacenar la fecha de nacimiento o la edad? Razona tu respuesta.

**Es mejor calcularla en un método, ya que el usuario introduce la fecha de nacimiento podemos extraer la edad restándola con la fecha actual del sistema.**

**2**. ¿Como se implementan las operaciones de una clase?

**JAVA**

**public void setAlias (String alias);**

**UML**

**+setAlias (alias: String);**

**3**. Puede una clase tener varios constructores? ¿Para qué?

La clase puede tener varios constructores y se denomina:

***Constructores sobrecargados***

***Por ejemplo, en un constructor podemos tener lo valores inicializados por defecto, y en otro tener los valores que se pasan. Teniendo en cuenta de que no se le puede pasar el mismo número de parámetros a todos por igual.***

**4.** Es correcta la expresión:(**(b>4) \* (c<2)**? Razona tu respuesta

**La expresión es incorrecta: no se pueden multiplicar tipos booleanos.**

5. Escribe la signatura de 3 métodos que formen parte de la interface

de la clase String.

Consulta la documentación en la API de Java en Oracle.

<https://docs.oracle.com/javase/7/docs/api/java/lang/String.html>

**String substring(int beginIndex, int endIndex)**

**String trim()**

**boolean equalsIgnoreCase(String anotherString)**

**6.** Puede un accesor no recibir ningún parámetro? Pon un ejemplo y razona tu Respuesta

***El accesor se encarga de devolver(return) alguna característica del objeto sin pasarle nada como parámetro.***

**public String getAlias () {**

**return alias**

**}**

**7**. Puede un mutador no recibir ningún parámetro? Pon un ejemplo y razona tu Respuesta.

***El mutador set se encarga de permitirnos cambiar el alias introducido como parámetro, al no recibir ningún parámetro no podemos cambiar el nombre del alias.***

**public class Alumno {**

**private String alias;**

**public void setAlias (String alias) {**

**this.alias = alias;**

**}**

**}**

**8**. Qué pasaría si no cumpliéramos la ENCASULACION y OCULTACION de datos

usando nivel de acceso **public** para los atributos de una clase?

***OCULTACIÓN***

**Al establecer publico los atributos estamos permitiendo que de cualquier clase se puedan modificar, nunca bajo ningún concepto se puede permitir que pueda llamar al objeto desde otra clase y modificarlo, por ejemplo, si el atributo alias fuera público:**

**public static class Instituto {**

**alumno.alias = “pepito”**

**}**

**El** *ENCASULAMIENTO* **es el encapsulado de todos los miembros en la clase.**

**Porque que puede incumplir**

**9**.. ¿Cuál es el ámbito de una variable de clase? ¿Y su tiempo de vida? Razona la Respuesta

***El ámbito de una variable define******su alcance de uso****, o lo que es lo mismo, en que secciones de código una variable estará disponible. Fuera de este ámbito, una variable no podrá ser accedida (no existe):*

* Local
* Global
* Estático

**El atributo está para toda la clase y todos los métodos pueden usar dicha variable.**

**Su tiempo de vida es hasta que termina el programa.**

**10.**. ¿Para que serviría definir una variable local estática (static)?

Haz una prueba y razona tu respuesta.

Idea: comprueba si la variable tiene un tiempo de vida distinto al que esperas…

**Las variables locales se utilizan en los métodos y si fuera estática se podría utilizar desde la clase pero su tiempo de vida es hasta que el programa muere.**

**class alumno {**

**private static int contador = 0;**

**}**

**11**.. ¿Puede un método de clase ser privado? Haz un ejemplo y razona tu respuesta.

**Un método de clase puede ser privado para evitar que el usuario que use el programa no ejecute dicho método desde otra clase a partir de un objeto a menos que use la clase para llamar al método**

private void setPeso(int peso){

this.peso = peso;

}

public static void main(String[] args){

}

**12.**. ¿Como se pueden controlar las restricciones sobre atributos?

**Los atributos se controlan en los mutadores.**

**13**. Piensa e implementa un mutador que permita modificar un atributo de tipo String

(supongamos que a nivel de diseño se ha establecido una restricción NOT NULL y otra

NO VACIO sobre el atributo)

**public class Prueba {**

**private String nombre;**

**public void setNombre(String nombre) {**

**assert nombre!=null && !nombre.isEmpty() : “Error: Se tiene que introducir un nombre”**

**}**

**14**.. ¿Es posible tener dos métodos en una clase cuyas signaturas sean iguales excepto el tipo de retorno? ¿Es un caso de sobrecarga?

**No es posible tener dos métodos que se llamen igual y tengan el mismo número de parámetros e idénticos el valor de retorno no implica nada en la sobrecarga**

**15**. ¿Por qué el uso de aserciones no es la mejor solución para el control de restricciones en mutadores?

**Porque cuando existe dicha restricción, se cierre el programa y de un mensaje de error y sepamos exactamente donde está el error.**

**Es una forma de depurar el programa de forma muy estricta, lo correcto sería utilizar ”if” para que el programa no se cierre si no muestre un error por pantalla.**