UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA – ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

71901072 – PROGRAMACIÓN ORIENTADA A OBJETOS (GRADO EN INGENIERÍA INFORMÁTICA / TECNOLOGÍAS DE LA INFORMACIÓN)

SEPTIEMBRE 2022 – ORIGINAL – NO ESTÁ PERMITIDO EL USO DE MATERIAL ADICIONAL

PARTE TEÓRICA - TEST [2,5 PUNTOS]:

El test consta de 14 preguntas y 2 preguntas adicionales de reserva. Solo una de las respuestas es válida. Las respuestas correctas se puntuarán con +1.0, mientras que las respondidas de manera incorrecta se puntuarán con -0.25. Las no contestadas no tendrán influencia ni positiva ni negativa en la nota.

Las preguntas de reserva sólo tendrán utilidad en el caso de que alguna de las 14 preguntas iniciales del test sea anulada por cualquier circunstancia. Caso de ocurrir este hecho, si se produjera la anulación de alguna de las 14 preguntas iniciales, la primera pregunta de reserva sustituiría a la pregunta anulada. Caso de que una segunda pregunta de las 14 iniciales fuese anulada, entonces la segunda pregunta de reserva sustituiría a esta segunda pregunta anulada. En aquellos hipotéticos casos en los que se produjese la anulación de una tercera o sucesivas preguntas de las 14 iniciales, entonces sólo en ese caso, las preguntas tercera y sucesivas anuladas se considerarían como correctas (al no existir más preguntas de reserva que las sustituyan).

Pregunta 1: ¿Cuál de las siguientes afirmaciones respecto a las instrucciones condicionales es correcta?

a. Una instrucción condicional lleva a cabo una acción entre un número impar de posibles acciones basándose en el resultado de una prueba.
b. Las instrucciones condicionales se conocen también como "instrucciones case".
c. Las expresiones booleanas son utilizadas a menudo como control de la elección entre las rutas de ejecución especificadas en una instrucción condicional.
d. El bloque "else" es siempre obligatorio cuando se utiliza una instrucción condicional.

Pregunta 2: Indica cuál de las siguientes afirmaciones es **falsa**:

a. Una variable local se define dentro del cuerpo de un constructor o método y sólo se puede inicializar y utilizar dentro del cuerpo de ese constructor o método.
b. Nunca se puede acceder a un campo definido como "private" desde ningún punto fuera de la clase en la que se ha definido.
c. Los campos mantienen el estado actual de un objeto y se definen fuera de los constructores o métodos.
d. El ámbito de un campo puede coincidir o no con la clase, en función de la accesibilidad que se defina al declararlo.

Pregunta 3: ¿Cómo se denomina al hecho de que una clase pueda contener más de un constructor o más de un método con el mismo nombre, siempre que cada uno tenga un conjunto diferente de tipos de parámetros?

a. Sobrecarga.
b. Polimorfismo.
c. Declaración múltiple.
d. Modularización.

Pregunta 4: ¿Cuál de las siguientes instrucciones permite declarar un campo que consiste en una colección de objetos String?

a. private ArrayList<Object:String> coleccion;
b. private ArrayList<String> coleccion;
c. private Array[] String coleccion;
d. private ArrayList[] String coleccion;

Pregunta 5: Indica cuál de las siguientes afirmaciones es correcta:

a. El método hasNext() es un método predefinido dentro de la clase ArrayList.
b. La única manera de iterar sobre una colección ArrayList es mediante un objeto Iterator.
c. El método remove de Iterator permite eliminar un objeto de la colección mientras se está iterando sobre ella.
d. La naturaleza del método remove de Iterator fuerza a que se deba utilizar siempre dentro de un bucle "for-each".

Pregunta 6: ¿Qué se mostrará por pantalla tras ejecutar el siguiente fragmento de código?

```
String input = "entrada";
input.toUpperCase();
System.out.println(input);
```

a. ENTRADA
b. entrada
c. Entrada
d. No se mostrará nada ya que se devolverá un error de ejecución al ser String un tipo inmutable.

Pregunta 7: Indica cuál de las siguientes afirmaciones es correcta:

a. Un mapa es una colección de parejas clave/valor de objetos, que permite almacenar un tipo flexible de entradas.
b. Las entradas en un mapa se buscan utilizando siempre un índice entero, de forma similar a ArrayList.
c. La búsqueda inversa en un mapa requiere un tiempo similar o menor a la búsqueda de un valor a partir de su clave.
d. La clase Map es una especialización de la clase HashMap.

Pregunta 8: Considera el siguiente código que trabaja con matrices bidimensionales. ¿Qué se mostrará por pantalla tras compilar y ejecutar dicho código?

```
public class clasePrueba{
    public static void main(String[] args){
        int[][] matrix = new int[3][4];
        for(int i = 0; i < matrix.length; i++){
            for(int j = 0; j < matrix[i].length; j++){
                matrix[i][j] = i*j;
            }
        }

        for(int k = 0; k < matrix[0].length; k++){
            System.out.print(metodo(matrix,k) + " ");
        }
    }
    public static int metodo(int[][] mat,int ind){
        int result = 0;
        for(int r = 0; r < mat.length; r++){
            result += mat[r][ind];
        }
        return result;
    }
}
```

a. 3 6 9 12
b. 0 6 9
c. 0 3 6 9
d. Se producirá un error de ejecución.

Pregunta 9: Indica cuál de las siguientes afirmaciones es correcta:

a. Un diseño con acoplamiento fuerte es recomendable, ya que implica que las clases son fundamentalmente independientes.
b. Un diseño con una alta cohesión es recomendable, ya que implica que cada unidad de código es responsable de una tarea bien definida.
c. En un sistema fuertemente acoplado podremos modificar una clase sin necesidad de modificar las demás, y la aplicación seguirá funcionando correctamente.
d. El concepto de cohesión es relevante a nivel de método, pero no al de clase ni al de paquete.

Pregunta 10: Indica cuál de las siguientes afirmaciones es **falsa**:

a. La directriz referida a la encapsulación sugiere que se debe hacer visible para el exterior la información acerca de lo que hace una clase y de cómo lo hace.
b. El diseño dirigido por responsabilidad es uno de los factores que influyen en el grado de acoplamiento.
c. El acoplamiento implícito describe una situación en la que una clase depende de otra de una forma no inmediatamente obvia.
d. Es importante disponer de un conjunto de pruebas antes de refactorizar un programa.

Pregunta 11: ¿Cómo se denomina al proceso de ejecutar pruebas que ya se pasaron anteriormente con éxito, cada vez que se realiza un cambio en el software?

a. Pruebas unitarias.
b. Ejecución de aserciones.
c. Pruebas de depuración.
d. Pruebas de regresión.

Pregunta 12: Disponemos del siguiente código:

```
1    class Prueba{
2         public static void main(String args[]){
3              A a = new A();
4              B b = new B();
5              C c = new C();
6              D d = new D();
7
8              d = c;
9              d = b;
10             a = c;
11
12             a = d;
13             a = b;
14             b = a;
15        }
16   }
```

Sabiendo que las 3 primeras asignaciones (líneas 8, 9 y 10) son legales (compilan correctamente), y que las tres últimas (líneas 12, 13 y 14) producen errores de compilación, ¿cuál será la estructura de herencia entre las cuatro clases propuestas?

a. La clase raíz es A; D hereda de A; B hereda de D; C hereda de D.
b. La clase raíz es A; C hereda de A; D hereda de C; B hereda de D.
c. La clase raíz es D; A hereda de D; B hereda de D; C hereda de A.
d. La clase raíz es D; C hereda de D; B hereda de D; A hereda de C.

Pregunta 13: ¿Qué implica la declaración de un método como "protected"?

a. Se puede acceder al método desde cualquier clase.
b. Únicamente se puede acceder al método desde la clase en la que está declarado.
c. Se puede acceder al método desde la misma clase en la que está declarado y desde cualquier clase que herede de ella, pero sólo si hereda directamente.
d. Se puede acceder al método desde la misma clase en la que está declarado y desde cualquier clase que herede directa o indirectamente de ella.

Pregunta 14: Indica cuál de las siguientes afirmaciones es **falsa**:

a. Un método abstracto se marca con la palabra clave "abstract" y se compone de una cabecera y un cuerpo del método.
b. Únicamente las clases abstractas pueden disponer de métodos abstractos.
c. Para transformar una subclase abstracta en concreta, se han de proporcionar implementaciones para todos los métodos abstractos heredados.
d. Intentar crear una instancia de una clase abstracta genera un error de compilación.

Pregunta R1: ¿Qué aparecerá en pantalla al ejecutar el siguiente código?

```
import java.util.Random;

public class Aleatorio
{
    public static void main(String args[]){
        Random rg = new Random();
        int index = rg.nextInt(5);
        System.out.println(index);
    }
}
```

a. Cualquiera de los siguientes números: 0, 1, 2, 3 o 4.
b. Cualquiera de los siguientes números: 0, 1, 2, 3, 4 o 5.
c. Cualquiera de los siguientes números: 1, 2, 3 o 4 o 5.
d. No se mostrará nada porque existe un error de compilación en el código.

Pregunta R2: Supongamos que tenemos una variable booleana denominada "decisor", cuyo valor en un momento dado es igual a "true", ¿cuál será la salida de la siguiente instrucción?

```
System.out.println(decisor ? "Primero" else "Segundo");
```

a. Primero
b. Segundo
c. Se produce un error de compilación.
d. Se produce un error de ejecución.

PARTE PRÁCTICA [6,5 PUNTOS]:

La Práctica del presente curso consiste en implementar un sistema integrado de gestión de un estudio de arquitectura. Los estudios de arquitectura, descrito de una forma genérica, se dedican a dos tipos de tareas. Por un lado, desarrollan proyectos arquitectónicos para la construcción o rehabilitación de edificios. Por otro lado, llevan a cabo una tarea de certificación.

Por tanto, el estudio ha de guardar información acerca de los proyectos y/o certificados que ha solicitado cada cliente, en un historial que es único para cada uno de los clientes.

En la aplicación que se desea desarrollar, es necesario almacenar dos tipos distintos de persona, para diferenciar si se trata de un cliente del estudio, o de un empleado del mismo. Dentro de los empleados, encontramos cuatro subtipos: el administrador del sistema, los arquitectos que realizan los proyectos y certificados, los aparejadores que llevan el día a día de una construcción, y los contables que llevan la economía del estudio.

Dentro de los proyectos de arquitectura que se pueden realizar encontramos tres tipos: los proyectos residenciales que engloban edificios y viviendas unifamiliares; los proyectos no residenciales como naves industriales, museos, o cualquier otra construcción no residencial; y los proyectos de rehabilitación que consiste en reformar una construcción ya finalizada previamente.

Por último, los certificados que desarrolla un estudio de arquitectura pueden ser de cuatro tipos: los certificados de habitabilidad, que además caducan cada 15 años y hay que renovarlos; la inspección técnica de edificios que se aplica a edificios comunitarios y es obligatoria a partir de los 45 años de su construcción; los certificados de eficiencia energética que se emiten con una categoría que puede ser desde la “A” (la más eficiente) a la “G” (la menos eficiente); y finalmente los informes periciales.

En general, las funciones que tiene el sistema de gestión son las siguientes:

- Gestión de usuarios: altas, bajas, modificaciones de las personas que figuran en el sistema (empleados -- administradores, arquitectos, aparejadores y contables -- y clientes). La primera vez que acude un cliente al estudio hay que darle de alta en el sistema.
- Desarrollo de un proyecto arquitectónico: los clientes acuden al estudio para solicitar un proyecto arquitectónico. Este proyecto puede ser de los tipos indicados anteriormente, y será necesario almacenar la fecha de solicitud, de entrega al cliente, duración prevista de la obra y presupuesto de ejecución. Una vez contratada la construcción también será necesario almacenar las fechas de inicio de construcción y duración prevista, y al finalizar la obra la fecha de fin de la obra. Un cliente puede solicitar todos los proyectos que desee y cada solicitud tiene que registrar los datos del cliente y del arquitecto que los desarrollaron. Cada proyecto arquitectónico tiene que registrar cierta información asociada al edificio. En el caso de proyectos residenciales tendrá que constar la dirección, superficie del terreno, superficie del edificio, plantas, habitaciones y baños. En el caso de proyectos no

residenciales, será necesario registrar la dirección, superficie del terreno, superficie del edificio y finalidad de la obra. Para los proyectos de rehabilitación habrá que registrar la dirección y la superficie a reformar. Además, cada proyecto tiene un coste que será determinado por un contable, el cual también tendrá que estar asociado al proyecto y quedar constancia en el sistema. Finalmente, un aparejador tendrá que llevar a cabo el control de la obra y por tanto habrá que asignar un aparejador a cada proyecto.

- Desarrollo de un certificado: los clientes acuden al estudio para solicitar un certificado. Este certificado puede ser de los tipos indicados anteriormente, y será necesario almacenar la fecha de solicitud y de entrega al cliente. Un cliente puede solicitar todos los certificados que desee y cada solicitud tiene que registrar los datos del cliente y del arquitecto que los desarrollaron. Además, cada proyecto tiene un coste que será determinado por un contable, el cual también tendrá que estar asociado al certificado y quedar constancia en el sistema. El sistema tendrá que almacenar un histórico para cada vivienda o edificio sobre los que se ha solicitado un certificado y en el que se puedan consultar todos los certificados asociados a esa vivienda o edificio. En el caso de los certificados de habitabilidad tendrá que haber una visita de un aparejador del estudio al edificio y un arquitecto desarrollará el certificado. Las fechas de la visita del aparejador y de la emisión del certificado tendrán que quedar registradas en el sistema, ya que a los 15 años caducan y el estudio mediante el administrador tendrá que ponerse en contacto con el cliente para informar de este hecho y preguntar por su posible renovación. La inspección técnica de edificios se aplica a edificios comunitarios y es obligatoria a partir de los 45 años de su construcción, por tanto, en el momento en que una vivienda para la que se ha desarrollado un proyecto cumple 45 años desde el fin de obra, el administrador se pondrá en contacto con el cliente de dicho proyecto para informar de este hecho y preguntar por su posible inspección por parte del estudio. Los certificados de eficiencia energética se emiten con una categoría que puede ser desde la “A” (la más eficiente) a la “G” (la menos eficiente) y además de esta categoría tendrá que registrar la fecha de emisión del certificado y del aparejador que haya realizado la visita. Por último, los informes periciales son desarrollados por un arquitecto del estudio y tendrán una fecha de emisión.
- Comunicación con los clientes: la función del administrador del estudio será la de ponerse en contacto con los clientes que o bien han solicitado un proyecto o un certificado. Para ello podrá obtener en todo momento el listado de dichos clientes. Además, podrá generar un listado de clientes cuya vivienda tenga más de 45 años para informar de la necesidad de obtener un certificado de inspección técnica de edificios y otro listado con los certificados de habitabilidad que han cumplido 15 años con el objetivo de informar a dichos clientes de la necesidad de renovarlos.
- Ejecución de obras: Una vez el cliente ha obtenido el proyecto y decide que el propio estudio desarrolle la obra, el administrador gestionará el calendario teniendo en cuenta que el estudio no puede tener más de tres obras en marcha al mismo tiempo. Para ello el administrador consultará la duración prevista de la obra que consta en el proyecto y asignará la fecha de inicio de la obra. Cuando la obra haya finalizado, también actualizará la fecha de fin de obra.

Se pide realizar las siguientes tareas:

a) **[1,0 puntos]** Diseñar utilizando un paradigma orientado a objetos, los elementos necesarios para la aplicación explicada de la práctica durante el curso. Es necesario identificar la estructura y las relaciones de herencia (mediante el uso de un diagrama de clases) y de uso de las clases necesarias para almacenar y gestionar esta información. Debe hacerse uso de los mecanismos de herencia siempre que sea posible. Se valorará un buen diseño que favorezca la reutilización de código y facilite su mantenimiento.

b) **[2,0 puntos]** Implementar un método (o métodos) que permita generar un listado de todos los certificados emitidos por el estudio de arquitectura, agrupados por el arquitecto responsable de la emisión de dicho certificado. Se deberá mostrar el tipo de certificado, así como la fecha en que se emitió.

c) **[2,0 puntos]** Implementar un método (o métodos) que permita calcular el coste total de todos los proyectos realizados por el estudio de arquitectura en un intervalo de tiempo determinado. Para ello, se deberá introducir la fecha de inicio y de fin de dicho intervalo, y se considerarán únicamente los proyectos cuyo fin de obra haya ocurrido dentro del intervalo especificado.

d) **[1,5 puntos]** Implementar un método (o métodos) que permita gestionar las siguientes estadísticas de trabajo de los arquitectos, aparejadores y contables relacionados con los proyectos del estudio de arquitectura:
   - Los 5 arquitectos del estudio cuyos proyectos (finalizados o sin finalizar) suman un mayor coste total.
   - Los 5 aparejadores que han participado en un mayor número de proyectos del estudio.
   - Los contables que han definido los 5 certificados de mayor coste almacenados en el sistema.