Nombre: ..... Fecha: / 01 / 2009

HOJA DE EJERCICIOS 6
REPASO TEMAS 1 A 5

1. Los modificadores que nos permiten determinar la visibilidad de un objeto (método o atributo) de una clase son (ordénalos de menor visibilidad a mayor visibilidad):

En Java:

En C++:

De las anteriores, la opción por defecto es:

En Java:

En C++:

2. Escribe la especificación e inicialización de los siguientes dos atributos (no incluyas las cabeceras de la clase):

a) Un atributo de clase, constante, privado, de tipo "double", de nombre "PI", y valor 3.1416;
b) Un atributo (no constante) de clase "contadorObjetos", privado, de tipo "int" y valor inicial "0".

En Java:

```
public class Ejemplo {



}
```

En C++:

```
*.h                                                    *.cpp
class Ejemplo{



};
```

3. Representa el diagrama UML de la siguiente clase "Usuario".

a) Atributos (privados, salvo que se diga lo contrario):
- nombre: string
- fechaNac: Date
- id: int, con modificador de acceso protegido
- contadorUsuarios: int, variable de clase

b) Métodos (públicos, salvo que se diga lo contrario):
- Usuario() de tipo protegido
- Usuario (string, Date, int)
- getNombre(): string
- setNombre(string): void
- getID(): int

- getContadorUsuarios(): int, que es un método estático (o de clase) y protegido
- getFechaNac(): Date
- toString(): string
- getSolicitud(): Solicitud, que es abstracto

4. Observa los siguientes comandos en Java y C++ (supón definida una clase “Color”, que contiene un constructor sin parámetros).

Java:

```
Color [] array_c = new Color [35];
```

C++:

```
Color array_c [35];
```

¿Son correctas (compilarían)?

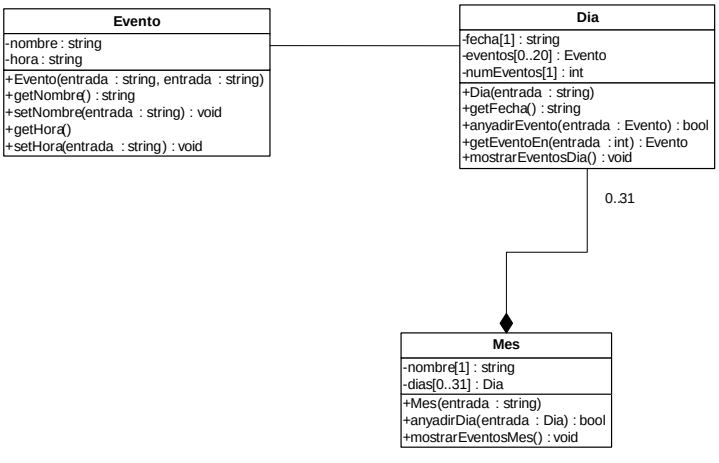
¿Son equivalentes?

¿Qué objeto ocupa cada posición de “array_c”?

En Java:

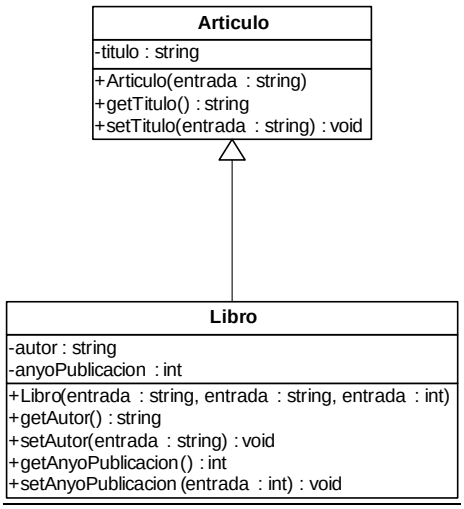
En C++:

5. Enumera las relaciones y los tipos de las mismas que observas en el siguiente diagrama UML:



Comenta cómo se ha resuelto la implementación de cada una de las relaciones, a la vista del gráfico UML:

6. Observa el siguiente gráfico UML:



Imagina que tenemos implementada la clase “Articulo” en Java y en C++. Escribe la definición (y la declaración, en C++) de la clase “Libro”:

En Java:

En C++:

*.h *.cpp

7. Supongamos una clase “Vehiculo” que cuenta con un método “getImpMatric(): double”, que devuelve el valor del atributo (privado) “impMatric: double”.

Estamos definiendo una clase en Java y C++, “Coche”, en la cual el método “getImpMatric(): double” debe ser redefinido con el siguiente comportamiento:

- Si el atributo (protegido) “CV: double” de “Vehiculo” es mayor o igual que 90, “getImpMatric(): double” devolverá el valor del atributo “impMatric: double” multiplicado por “1.25”;

- En otro caso (si el valor de “CV: double” es menor que 90), “getImpMatric(): double” devolverá el valor del atributo “impMatric: double” multiplicado por “0.90”.

Escribe la definición de dicho método en Java:

Escribe la definición de dicho método en C++:

8. ¿Qué relación debe existir entre las clase “A” y “B” para que los siguientes comandos sean correctos?

A aux = new B();

A [] array_A = new A [45];
array_A [3] = new B();

9. ¿Cuáles son los dos requisitos previos necesarios para que tenga sentido decir que un método se comporta de modo polimorfo?

¿Qué tipo de enlazado nos encontramos por defecto en C++?

10. Observa el siguiente caso práctico:

Consideramos una clase “Articulo” y una clase “Libro” que hereda de ella. El método “getPrecio(): double” está definido en “Articulo”, y redefinido en “Libro”.

¿Cuál de estas versiones se invocaría en el siguiente fragmento de código?

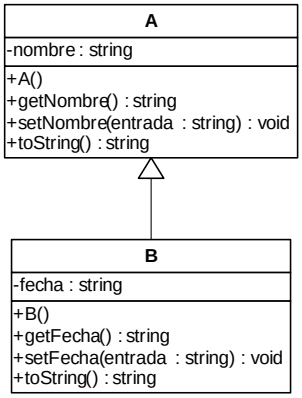
C++:

```
void mostrarPrecio(Articulo a1){
        cout << “El precio del articulo es” << a1.getPrecio() << endl;
}
```

Java:

```
public static void mostrarPrecio(Articulo a1){
        System.out.println("El precio del articulo es" + a1.getPrecio());
}
```

11. Observa el siguiente diagrama de clases:



Suponemos ahora el siguiente fragmento de código (en Java):

```
public static void main (String [] args){

        A  [] array_a = new A[45];
        ...
}
```

¿Cuál sería el resultado de ejecutar ahora el siguiente comando a continuación?

```
B aux = new B();
aux.getFecha();
```

¿Cuál sería el resultado de ejecutar el siguiente comando a continuación?

```
array_a [0] = aux;
array_a [0].getFecha();
```

¿Cuál de las definiciones de "toString(): string" sería utilizada ahora?

```
array_a [0].toString();
```

12. ¿Cuáles son los dos requisitos imprescindibles para que un método se comporte de modo polimorfo en C++?

13. ¿Cuál es la definición (y principal característica) de "clase abstracta"?

¿Una clase abstracta puede contener constructores?

¿Cómo se puede distinguir una clase abstracta en Java?

¿Y en C++?

14. ¿Qué tipo de elementos se pueden declarar en una “interface”?

¿Una “interface” puede contener constructores?

¿De una “interface” se pueden declarar objetos?

¿Qué estructura utilizarías en C++ para representar una “interface” de Java?

15. Observa la siguiente cabecera en Java, y expresa su significado en palabras:

```
public abstract class A extends B implements D, E{
        ...
}
```

16. Supongamos que tenemos una “interface” “A” en Java que contiene métodos “escribir(): String”, “ordenar(): void” e “insertar(double): void”.

Escribe (sólo las cabeceras de la clase y de los métodos) una clase abstracta “B” que implemente la “interface” “A”, y defina los métodos “insertar(double): void” y “escribir(): String”.

17. ¿Cuál es la clase de la que heredan todas las clases que representan excepciones en Java? (Por supuesto, no nos estamos refiriendo a “Object”)

¿Cuál es la clase de la que heredan todas las clases que representan excepciones que no es obligatorio gestionar por parte del usuario?

Enumera tres métodos que se puedan usar sobre cualquier excepción. Explica brevemente el resultado de aplicar los mismos.

1.

2.

3.

18. ¿Cuál sería el resultado de ejecutar el siguiente fragmento de código sobre el fichero “entrada.txt” que se muestra a continuación ?

```
public static void main (String [] args){
        File fichero = new File (“entrada.txt”);
        double aux;
        try{
                aux = escanea_fichero();
                System.out.println (“El valor leido es ” + aux);
        }
        catch (FileNotFoundExcepction f){
                System.out.println (“El fichero no existe”);
        }
}

public static double escanea_fichero (File fichero) throws FileNotFoundException{
        Scanner lectura_fichero = new Scanner (fichero);
        Double un_double = new Double (0.0);
        while (lectura_fichero.hasNext()){
                try {
                        un_double = new Double (lectura_fichero.nextLine());
                }
                catch (NumberFormatException nfe){
                        System.out.println ("El valor leido no era un double");
                }
        }
        return un_double;
}
```

Fichero “entrada.txt”:

Este fichero contiene números que se van a leer, además de esta primera línea de texto:
8.345
0.89067
2.3456

19. ¿Cuál sería el resultado de ejecutar el siguiente fragmento de código sobre el fichero “entrada.txt” que se muestra a continuación ?

```
public static void main (String [] args){
        File fichero = new File (“entrada.txt”);
        double aux;
        try{
```

```
                aux = escanea_fichero();
                System.out.println (“El valor leido es ” + aux);
        }
        catch (FileNotFoundExcepction f){
                System.out.println (“El fichero no existe”);
        }
}

public static double escanea_fichero (File fichero) throws FileNotFoundException{
        Scanner lectura_fichero = new Scanner (fichero);
        Double un_double = new Double (0.0);
        while (lectura_fichero.hasNext()){
                try {
                        un_double = new Double (lectura_fichero.nextLine());
                }
                catch (NumberFormatException nfe){
                        System.out.println ("El valor leido no era un double");
                        throw nfe;
                }
        }
        return un_double;
}
```

Fichero “entrada.txt”:

Este fichero contiene números que se van a leer, además de esta primera línea de texto:
8.345
0.89067
2.3456

20. Define en Java un método “mostrar_elementos (Object []): void” que se comporte del siguiente modo.

- Si el “array” contiene algún elemento (puedes usar el atributo “length” del mismo para conocer su longitud) debe mostrar todos los elementos de mismo por pantalla, por medio de “toString(): String”;

- Si la longitud del “array” es igual a 0, crea y lanza una excepción de tipo “Exception” que contenga el mensaje “El array estaba vacío”.

(No olvides escribir también la cabecera del método)