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

HOJA DE EJERCICIOS 4
INTERFACES Y CLASES ABSTRACTAS EN JAVA Y C++

Esta hoja de ejercicios nos servirá para repasar las nociones más importantes del Tema 4.

1. ¿Cuál es la definición de clase abstracta?

2. ¿Una clase abstracta puede contener constructores?

3. ¿Una clase abstracta puede contener atributos?

4. Realiza el diagrama UML de una "interface" "*ShapeInt*" que contenga los métodos (públicos) "computeArea(): double", "computePerimeter(): double" y "readShapeData(): void".

5. Realiza el diagrama UML de una clase abstracta "*Drawable*" en Java que contenga atributos (privados) "pos: Point", "borderColor: Color", "interiorColor: Color", así como métodos (públicos, no abstractos) "setBorderColor(Color): void", "setInteriorColor(Color): void", "setPos(Point): void" y un método abstracto "drawMe (Graphics): void". Además, dicha clase redefine el método "toString(): String".

6. Recupera el diagrama UML del ejercicio 4 y añádele una clase (no abstracta) "Rectangle" que implemente "*ShapeInt*" y contenga atributos "width: double", "height: double", y defina los métodos de la "interface". La clase contendrá un único constructor (público) "Rectangle (double, double)". Además, redefinirá el método "toString(): String".

7. Escribe la definición en Java de la “interface” “*ShapeInt*” de acuerdo con la especificación que de la misma hemos hecho en el ejercicio 4.

8. Escribe la definición en Java de la clase “Rectangle” de acuerdo con la especificación de la misma que hemos hecho en el ejercicio 6 (incluidas las relaciones de la misma).

La especificación de sus métodos será la siguiente:

“Rectangle (double, double)”: inicia los atributos “width: double” y “height: double” con los parámetros recibidos;

“computeArea (): double” : calcula el área del rectángulo (base x altura);

“computePerimeter (): double” : calcula el perímetro del rectángulo (2 x base + 2 x altura);

“readShapeData (): void” : método que lee la base y altura del rectángulo, y las asigna a los atributos de la clase (puedes hacer uso de la clase “Scanner” para leer los mismos).

“toString(): String” : método que devuelve una cadena de texto que contenga los dos atributos de la clase

9. Define en C++ una clase completamente abstracta que represente la “interface” “*ShapeInt*” de acuerdo con la especificación que de la misma hemos hecho en el ejercicio 4.

10. Escribe la declaración y definición en C++ de la clase “Rectangle” de acuerdo con la especificación de la misma que hemos hecho en el ejercicio 6 (incluidas las relaciones de la misma).

El método “toString(): String” sustitúyelo por un método “toString(): void” que tenga el siguiente comportamiento:

“toString(): void” : método que muestra una cadena de texto que contenga los dos atributos de la clase.

Rectangle.h

Rectangle.cpp

11. Observa la siguiente cabecera en Java, dada la interface “ShapeInt” previa ¿Es correcta?

```
public class Circle extends ShapeInt {
        ...
}
```

12. Observa la siguiente cabecera en C++, dada la clase completamente abstracta “ShapeInt” ¿Es correcta?

```
class Circle: public ShapeInt {
        ...
}
```

13. Las dos cabeceras a continuación, ¿son equivalentes en Java y C++?

```
public class Circle implements ShapeInt {
        ...
}

class Circle: ShapeInt {
        ...
}
```

14. Escribe (sólo) la cabecera de una clase pública “DrawableRectangle” en Java que implemente las dos interfaces “*ShapeInt*” y “*Drawable*”.

15. Escribe la lista de métodos que debe contener la clase “DrawableRectangle” para no ser abstracta con respecto a la declaración de la misma que has hecho en el ejercicio 14 (incluye entre los mismos, al menos, un constructor).