Actividad resuelta 8.1

Diseñar la clase `Hora`, que representa un instante de tiempo compuesto por la hora (de 0 a 23) y los minutos. Dispone de los métodos:

- `Hora(hora, minuto)`, que construye un objeto con los datos pasados como parámetros.
- `void inc()`, que incrementa la hora en un minuto.
- `boolean setMinutos(valor)`, que asigna un valor, si es válido, a los minutos. Devuelve `true` o `false` según haya sido posible modificar los minutos o no.
- `boolean setHora(valor)`, que asigna un valor, si está comprendido entre 0 y 23, a la hora. Devuelve `true` o `false` según haya sido posible cambiar la hora o no.
- `String toString()`, que devuelve un `String` con la representación de la hora.

Actividad resuelta 8.2

A partir de la clase `Hora` implementar la clase `HoraExacta`, que incluye en la hora los segundos. Además de los métodos heredados de `Hora`, dispondrá de:

- `HoraExacta(hora, minuto, segundo)`, que construye un objeto con los datos pasados como parámetros.
- `setSegundo(valor)`, que asigna, si está comprendido entre 0 y 59, el valor indicado a los segundos.
- `inc()`, que incrementa la hora en un segundo.

Solución

Actividad resuelta 8.3

Añadir a la clase `HoraExacta` un método que compare si dos horas (la invocante y otra pasada como parámetro de entrada al método) son iguales o distintas.

Actividad resuelta 8.4

Crear la clase abstracta `Instrumento`, que almacena en una tabla las notas musicales de una melodía (dentro de una misma octava). El método `add()` añade nuevas notas musicales. La clase también dispone del método abstracto `interpretar()` que, en cada subclase que herede de `Instrumento`, mostrará por consola las notas musicales según las interprete. Utilizar enumerados para definir las notas musicales.

Actividad resuelta 8.5

Crear la clase `Piano` heredando de la clase abstracta `Instrumento`.

Solución

8.1. Sobre una subclase es correcto afirmar que:

a) Tiene menos atributos que su superclase.
b) Tiene menos miembros que su superclase.
c) Hereda los miembros no privados de su superclase.
d) Hereda todos los miembros de su superclase.

8.2. En relación con las clases abstractas es correcto señalar que:

a) Implementan todos sus métodos.
b) No implementan ningún método.
c) No tienen atributos.
d) Tienen algún método abstracto.

8.3. ¿En qué consiste la sustitución u overriding?

a) En sustituir un método heredado por otro implementado en la propia clase.
b) En sustituir un atributo por otro del mismo nombre.
c) En sustituir una clase por una subclase.
d) En sustituir un valor de una variable por otro.

8.4. Sobre la clase `Object` es cierto indicar que:

a) Es abstracta.
b) Hereda de todas las demás.
c) Tiene todos sus métodos abstractos.
d) Es superclase de todas las demás clases.

8.5. ¿Cuál de las siguientes afirmaciones sobre el método `equals()` es correcta?

a) Hay que implementarlo, ya que es abstracto.
b) Sirve para comparar solo objetos de la clase `Object`.
c) Se hereda de `Object`, pero debemos reimplementarlo al definirlo en una clase.
d) No hay que implementarlo, ya que se hereda de Object.

8.6. ¿Cuál de las siguientes afirmaciones sobre el método `toString()` es correcta?

a) Sirve para mostrar la información que nos interesa de un objeto.
b) Convierte automáticamente un objeto en una cadena.
c) Encadena varios objetos.
d) Es un método abstracto de `Object` que tenemos que implementar.

8.7. ¿Cuál de las siguientes afirmaciones sobre el método `getClass()` es correcta?

a) Convierte los objetos en clases.
b) Obtiene la clase a la que pertenece un objeto.
c) Obtiene la superclase de una clase.
d) Obtiene una clase a partir de su nombre.

8.8. Una clase puede heredar:

a) De una clase.
b) De dos clases.
c) De todas las clases que queramos.
d) Solo de la clase `Object`.

8.9. La selección dinámica de métodos:

a) Se produce cuando una variable cambia de valor durante la ejecución de un programa.

b) Es el cambio de tipo de una variable en tiempo de ejecución.

c) Es la asignación de un mismo objeto a más de una variable en tiempo de ejecución.

d) Es la ejecución de distintas implementaciones de un mismo método, asignando objetos de distintas clases a una misma variable, en tiempo de ejecución.

8.10. ¿Cuál de las siguientes afirmaciones sobre el método `super()` es correcta?

a) Sirve para llamar al constructor de la superclase.

b) Sirve para invocar un método escrito más arriba en el código.

c) Sirve para llamar a cualquier método de la superclase.

d) Sirve para hacer referencia a un atributo de la superclase.

8.11. Crea la clase `Campana` que hereda de `Instrumento` (definida en la Actividad resuelta 8.4).

8.12. Las empresas de transporte, para evitar daños en los paquetes, embalan todas sus mercancías en cajas con el tamaño adecuado. Una caja se crea expresamente con un ancho, un alto y un fondo y, una vez creada, se mantiene inmutable. Cada caja lleva pegada una etiqueta, de un máximo de 30 caracteres, con información útil como el nombre del destinatario, dirección, etc. Implementa la clase `Caja` con los siguientes métodos:

- `Caja(int ancho, int alto, int fondo, Unidad unidad)`: que construye una caja con las dimensiones especificadas, que pueden encontrarse en «cm» (centímetros) o «m» (metros).
- `double getVolumen()`: que devuelve el volumen de la caja en metros cúbicos.
- `void setEtiqueta(String etiqueta)`: que modifica el valor de la etiqueta de la caja.
- `String toString()`: que devuelve una cadena con la representación de la caja.

8.13. La empresa de mensajería BiciExpress, que reparte en bicicleta, para disminuir el peso que transportan sus empleados solo utiliza cajas de cartón. El volumen de estas se calcula como el 80 % del volumen real, con el fin de evitar que se deformen y se rompan. Otra característica de las cajas de cartón es que sus medidas siempre están en centímetros. Por último, la empresa, para controlar costes, necesita saber cuál es la superficie total de cartón utilizado para construir todas las cajas.

Escribe la clase `CajaCarton` heredando de la clase `Caja`.

8.14. Reimplementa la clase `Lista` de la Actividad resuelta 7.11, sustituyendo el método `mostrar()` por el método `toString()`.

8.15. Escribe en la clase `Lista` un método `equals()` para compararlas. Dos listas se considerarán iguales si tienen los mismos elementos (incluidas las repeticiones) en el mismo orden.

8.16. Diseña la clase `Pila` heredando de `Lista` (ver Actividad resuelta 7.13).

8.17. Escribe la clase `Cola` heredando de `Lista` (ver Actividad final 7.18).

8.18. Diseña la clase `ColaDoble`, que hereda de `Cola` para enteros, añadiendo los siguientes métodos:

- `void encolarPrincipio()`, que encola un elemento al principio de la cola.
- `Integer desencolarFinal()`, que desencola un elemento del final de la cola.

8.19. Un conjunto es un objeto similar a las listas, capaz de guardar valores de un tipo determinado, con la diferencia de que sus elementos no pueden estar repetidos. Escribe la clase `Conjunto` para enteros heredando de `Lista` y reimplementando los métodos de inserción para evitar las repeticiones.

8.20. Implementa el método `equals()` en la clase `Conjunto`. Dos conjuntos se consideran iguales si tienen los mismos elementos, no importa en qué orden.

8.21. Implementa los siguientes métodos:

- `static boolean esNumero(Object ob)`, que nos dice si su parámetro de entrada es de tipo numérico (`Integer, Double, Long, Float. . .`).
- `boolean sumar(Object a, Object b)`, que muestra por consola la concatenación de los parámetros de entrada, si ambos son cadenas, o muestra su suma convertida al tipo `Double`, si ambos son de tipo numérico. En cualquier otro caso, muestra el mensaje «No sumables».

8.22. La clase `Object` dispone del método `finalize()`, que se ejecuta justo antes de que el recolector de basura destruya un objeto. Escribe un programa que, mediante la creación masiva de objetos no referenciados y el overriding del método `finalize()`, compruebe el funcionamiento del recolector de basura.

8.23. Implementa la clase abstracta `Poligono`, con los atributos `base` y `altura`, de tipo double y el método abstracto `double area()`.

8.24. Heredando de `Poligono`, implementa las clases no abstractas `Triangulo` y `Rectangulo`.