UD03.EXAME Práctico

DAM1-Programación 2023-24
2023/12/12

- Puedes utilizar apuntes y materiales que consideres pero deberás realizar los programas individualmente. En caso contrario se retirará el examen.
- Realiza programas bien estructurados, legibles, con comentarios, líneas en blanco, identificadores adecuados, etc.
- Cuida la interacción con el usuario, presentando la información de forma clara y ordenada.
- Utiliza los casos de prueba de ejemplo para probar tu programa y definir la salida por pantalla.

- Crea un **paquete** de nombre **examenud03** donde agrupar los ficheros de código fuente. Si es necesario crea un nuevo proyecto/carpeta Java.
- Crea dentro del paquete ficheros **.java** por cada ejercicio con el nombre indicado.
- Incluye al inicio de cada programa un **comentario con tu nombre y apellidos**.
- **Comprime y entrega la carpeta del paquete en el Aula Virtual al terminar.**

- **Tiempo estimado:** 2 horas

1. Ascensor (4)

Ascensor.java

Desarrolla una clase Ascensor que gestione el desplazamiento vertical entre pisos de un edificio. La clase debe tener un atributo privado, `pisoActual`, que representa el piso en el que se encuentra actualmente el ascensor, y otros dos atributos, `PLANTA_MAS_BAJA` y `PLANTA_MAS_ALTA`, públicos y constantes una vez asignados y que representan el piso más bajo y el más alto respectivamente.

Implementa 3 constructores:

- Uno que permita indicar el piso actual, la planta más baja y la planta más alta. Si la planta más baja es mayor o igual que la planta más alta o si el piso actual no se encuentra entre ellas, extremos incluidos, este constructor lanzará una excepción del tipo `IllegalArgumentException`.
- Otro que permita indicar la planta más baja y la planta más alta, situando el piso actual del ascensor en la planta más baja. Si la planta más baja es mayor o igual que la planta más alta, este constructor también lanzará una excepción del tipo `IllegalArgumentException`.
- Un constructor predeterminado que sitúe el ascensor por defecto en la planta baja, o piso cero, en un edificio de 9 plantas de viviendas y 2 sótanos de aparcamientos.

Además, implementa los siguientes métodos para las operaciones básicas:

- El *getter* para obtener el piso actual.
- `subir()`: incrementa el piso actual en uno para simular el ascenso de un piso.
- `bajar()`: decrementa el piso actual en uno para simular el descenso de un piso.
- `subirN(int n)`: incrementa el piso actual en `n` unidades para simular el ascenso de varios pisos.
- `bajarN(int n)`: decrementa el piso actual en `n` unidades para simular el descenso de varios pisos.
- `irAlPiso(int destino)`: establece el piso actual en el valor proporcionado, simulando la operación de ir directamente a un piso específico.

Asegúrate de manejar adecuadamente los límites superior e inferior del edificio para evitar operaciones que superen los pisos disponibles. Si las operaciones de los métodos anteriores se pueden llevar a cabo devolverán true. En caso contrario devolverán false y el ascensor se mantendrá en el piso en el que estaba.

Además, implementa un método `mostrar()` que imprima el piso actual del ascensor.

Inspirado en Ascensor - ¡Acepta el reto!

2. Recorrido ascensor (3)

RecorridoAscensor.java

Utiliza la clase anterior para crear un programa principal que resuelva el siguiente problema:

En un edificio de 9 pisos más la planta baja (piso 0), se desea calcular el recorrido que a lo largo del día hace un ascensor, contado en número de pisos que se ha movido arriba y abajo.
Para ello se proporciona una entrada por teclado que contiene una secuencia de números enteros. El primer número marca el piso inicial en el que empieza el ascensor. A continuación se introducen parejas de enteros, cada una de ellas representando el uso del ascensor por parte de un vecino. El primer número de la pareja representa el piso desde el que llama al ascensor y el segundo número el piso de destino. La entrada finaliza con un -1.

Al finalizar la entrada el programa mostrará la longitud (en número de pisos) del recorrido completo del ascensor a lo largo del día.

Ejemplos de entradas y salidas:

Entrada	**Salida**
`0 1 5 2 0 3 9 -1`	`19`
`5 5 4 -1`	`1`
`1 2 3 4 5 -1`	`4`
`0 0 7 2 0 0 3 9 3 -1`	`29`

Nota: si no has implementado la clase Ascensor puedes intentar resolver el problema sin utilizar objetos.

3. Recorrido de dos ascensores (3)

RecorridoDosAscensores.java

En el edificio del ejercicio anterior se ha instalado un segundo ascensor para reducir las esperas de los vecinos y optimizar los recorridos. Ahora cuando un vecino llame ascensor acudirá aquel que esté más cerca (en número de pisos). Si los dos ascensores se encuentran a la misma distancia acudirá el primer ascensor.

Crea un nuevo programa, similar al anterior, que acepte una secuencia de números enteros. El primer número marca el piso inicial en el que empieza el primer ascensor. El segundo número marca el piso inicial en el que empieza el segundo ascensor. A continuación, como antes, se introducen parejas de enteros representando el piso desde el que se llama al ascensor y el piso de destino. La entrada finaliza con un -1.

Como salida, el programa mostrará el número de pisos recorridos por cada ascensor.

Ejemplos de entradas y salidas:

Entrada	**Salida**
`0 0 1 5 2 0 3 9 -1`	`13 4`
`5 5 5 4 -1`	`1 0`
`1 1 2 3 4 5 -1`	`4 0`
`0 0 0 7 2 0 0 3 9 3 -1`	`15 7`