Taller de Programación APREF 2020

https://tinyurl.com/Taller-APREF-2020

1.1. Genere una clase para representar espectadores. Un espectador se caracteriza por DNI, nombre y edad. Implemente un constructor que permita iniciar el espectador creado con un DNI, nombre y edad recibidos por parámetro. Incorpore getters/setters para los atributos.

Usando la clase desarrollada en el inciso anterior. Generar una clase para representar funciones de teatro. Una función de teatro se caracteriza por mantener: título de la obra, fecha, horario, una estructura que representa la sala (20 filas y 10 butacas por fila) que almacenará los espectadores asistentes, y una estructura que almacena para cada fila la cantidad de butacas ocupadas.

a) Defina métodos getters/setters para título, horario y fecha.

b) Implemente un constructor que permita iniciar la función de teatro creada con un título, fecha y horario recibidos por parámetro y sin espectadores en su sala.

c) Incorpore los siguientes métodos:

- validarFila: recibe un nro. “F” y devuelve un boolean que indica si “F” corresponde a una fila de la sala.
- hayButacaLibreEnFila: recibe un nro. de fila “F” válido y devuelve un boolean que indica si hay una butaca libre en la fila “F”.
- agregarEspectadorAFila: recibe un nro. de fila “F” válido y un espectador “E”, y agrega a “E” en la primer butaca libre de la fila “F”, debiendo retornar el nro. de butaca asignada.
- calcularButacasOcupadas: calcula y devuelve la cantidad total de butacas ocupadas para la función de teatro.
- calcularButacasLibres: calcula y devuelve la cantidad total de butacas libres para la función de teatro.
- calcularEdadPromEspectadores: calcula y devuelve la edad promedio de los espectadores asistentes.
- estaRegistradoEspectador: recibe un nro. de DNI “D” y devuelve un boolean que indica si existe un espectador con DNI igual a “D” registrado para la función de teatro.

Hecho

Sigue…

1.2- Escriba un programa que instancie una función de teatro para la obra titulada “Cazafantasmas” con horario “20:00” hs y fecha “19-12-2017”. Luego simule la venta de localidades de la siguiente manera. Leer DNIs de personas hasta que se ingresa el DNI 0 o hasta que no queden butacas libres para esta función de teatro. A cada persona se le solicita además nombre, edad y el nro. de fila que prefiere. En caso que el nro. de fila sea válido, exista una butaca libre en esa fila y la persona no esté registrada para la función de teatro, agregar a la persona como espectador en la fila solicitada e imprimir en consola el nro. de butaca asignada. Caso contrario, informar el error correspondiente. Al finalizar la venta, informar la cantidad de butacas ocupadas, la cantidad de butacas libres y la edad promedio de los espectadores.

Hecho

2.1- Implemente una clase para representar PCs. Una PC se caracteriza por su estado, que indica si está encendida o apagada, y su consumo por hora. Implemente un constructor que inicie la PC como apagada y con un consumo por hora recibido. La PC debe responder a los mensajes:

- encender: enciende la PC
- apagar: apaga la PC
- getEstado: devuelve el estado de la PC
- getConsumo: devuelve el consumo por hora de la PC

2.2- Implemente la clase Sala de PC. Una Sala de PC se caracteriza por mantener un vector con a lo sumo N PCs. Implemente un constructor que inicie la sala de PC con espacio para N PCs, con N recibido por parámetro; la sala inicialmente no tendrá PCs. La sala debe responder a los siguientes mensajes:

- agregarPC: recibe una PC y la agrega al final del vector.
- hayEspacio: devuelve un boolean que indica si hay espacio en la sala para al menos una PC.
- encenderPCs: enciende todas las PCs de la sala.
- ahorrarEnergia: recibe un valor C y apaga todas las PCs cuyo consumo por hora supere C.
- obtenerNrosEncendidas: devuelve un string con los números de PCs encendidas concatenados.
- calcularConsumoSala: calcula y devuelve el consumo de la sala, que es la suma de los consumos por hora de las PCs encendidas.

Sigue…

2.3- Implemente un programa que cree una sala de PC con espacio para N PCs (N se lee por teclado). Cree PCs y agréguelas a la sala, hasta que la sala se complete o leer un valor de consumo por hora igual a 0. Luego encienda las PCs de la sala y muestre en consola el consumo de la misma. Por último, aplique el ahorro de energía a la sala (lea el valor de consumo tope desde teclado), muestre el consumo de la sala y los números de PCs que quedaron encendidas.

3.1 Se desea implementar un sistema que simule dos juegos de cartas específicos:

A. El “Cambiazo” que se juega de la siguiente manera. Todos los jugadores eligen una de sus cartas para intercambiársela con el siguiente jugador. Por ejemplo, en un juego de cuatro jugadores con cinco cartas cada uno, el jugador 1 puede elegir cambiar su carta 3 con la del jugador 2 (ambos intercambian su carta 3), el jugador 2 luego puede intercambiar la carta 3, u otra con el jugador 3, éste intercambia con el jugador 4, finalmente el jugador 4 intercambia una carta con el jugador 1.

Todos los jugadores intercambian sus cartas solo una vez.

Gana el juego el jugador que tiene la menor cantidad de puntos (sumando todas sus cartas).

B. El “Descarte” que se juega de la siguiente manera. Cada jugador tira una de sus cartas y toma otra del mazo. El juego tiene dos rondas.

Gana el juego el jugador que tiene la menor cantidad de puntos (sumando todas sus cartas).

Sigue…

3.2 Implemente una clase para el juego “Cambiazo” y otra clase para el juego “Descarte”. Ambas clases deben implementar lo siguiente:

a. Un constructor que reciba la cantidad de jugadores y la cantidad de cartas que le tocan a cada jugador.

b. Un método agregarJugador que recibe el nombre de un jugador (un string) y lo almacena.

c. Un método repartir que recibe un mazo de cartas y reparte la cantidad de cartas que le corresponde a cada jugador.

NOTA: Por simplicidad simule el mazo de cartas como un vector de 40 enteros. Para inicializarlo puede usar el siguiente código:

```
for(int i = 0; i < 40; i++)
    mazo[i] = y % 10;
```

d. Un método jugar que simula el juego. Cada jugador siempre elige entre sus cartas, la de mayor puntaje para intercambiarla o tirarla (dependiendo del juego)

e. Un método ganador que devuelve el nombre del jugador que ganó el juego.

Sigue…

3.3 Implemente una función main que instancie un “Cambiazo” con cuatro jugadores y seis cartas para cada uno. Que instancie un “Descarte” con tres jugadores y cuatro cartas cada uno. Agregue al juego los nombres de los jugadores leídos por teclado.

3.4 Simule cada uno de los juegos enviando a las instancias de ambos juegos los mensajes repartir y jugar. Finalmente imprima el nombre del ganador de cada juego.

4.1 Queremos representar las personas que trabajan en una institución educativa: administrativos y docentes.

- Cualquier persona se caracteriza por: nombre, DNI, año de ingreso y sueldo básico.
- Los administrativos son personas que se caracterizan por el turno en el que trabaja (ejemplo: 1: mañana/ 2: tarde/ 3: noche).
- Los docentes son personas que se caracterizan por las horas semanales que debe trabajar.

4.2 Realice un primer modelo de la jerarquía de clases propuesta, con los atributos de cada clase y métodos para obtener/modificar el valor de los mismos. Programe cada clase y genere constructores para los administrativos y docentes, que reciban toda la información necesaria para iniciar los atributos del objeto a crear.

Sigue…

4.3 Incorpore la funcionalidad que permita calcular y devolver el sueldo final de cada persona (método calcularSueldoFinal). El mismo se calcula teniendo en cuenta lo siguiente:

- Para los administrativos es el sueldo básico y para aquellos que trabajan en el turno noche se adicionan 350$.
- ara los docentes que trabajan entre 0 y 9 horas el sueldo final es un sueldo básico; para los que trabajan entre 10 y 25 horas semanales, el sueldo final son dos sueldos básicos; para aquellos que trabajan entre 26 y 40 horas semanales, el sueldo final son cuatro sueldos básicos.

4.4 Genere un programa principal que instancie un administrativo y un docente con datos fijos. Pruebe el correcto funcionamiento del método calcularSueldoFinal.

5.1- Generar una clase para representar canciones, que se caracterizan por título, intérprete y duración en segundos. Implemente un constructor que inicie el objeto con título, intérprete y duración recibidos, y los métodos getters/setters y toString (ej. “Bohemian Rhapsody – Queen – 360 segs”).

b- Generar una clase Reproductor MP3. Un Reproductor MP3 almacena a lo sumo 100 canciones y mantiene el nro. de canción en reproducción (esto es, la posición dentro del almacenamiento de la canción que se está reproduciendo).

Implemente un constructor que inicie el reproductor MP3 sin canciones, y los siguientes métodos:

- quedaEspacio: retorna un boolean que indica si hay espacio para almacenar una canción más.
- cargarCancion: recibe una canción y la almacena en el reproductor MP3.
- iniciarReproduccion: inicia el nro. de canción en reproducción en 0.
- reproducir: imprime en consola el título, intérprete y duración de la canción en reproducción.
- avanzar: actualiza el nro. de canción en reproducción para pasar a la siguiente o volviendo a 0 en caso de haberse reproducido todas las canciones almacenadas.
- calcularDuracionTotal: calcula y devuelve la duración en segundos total entre todas las canciones almacenadas.

Sigue…

5.2- Realice un programa que instancie un reproductor MP3. Cárguele canciones, con datos leídos de teclado, hasta quedarse sin espacio o hasta leer el título de canción “ZZZ”. Luego lea la cantidad de canciones que quiera escuchar (N), inicie la reproducción y reproduzca N canciones, avanzando a la próxima luego de cada reproducción. Al finalizar muestre la duración total entre todas las canciones almacenadas en el reproductor.