EnunciadoControl1C7Mayo2020.pdf

RexBlack

Programación Orientada a Objetos

1º Grado en Ingeniería del Software

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

DEPARTAMENTO
LENGUAJES Y CIENCIAS
DE LA COMPUTACIÓN

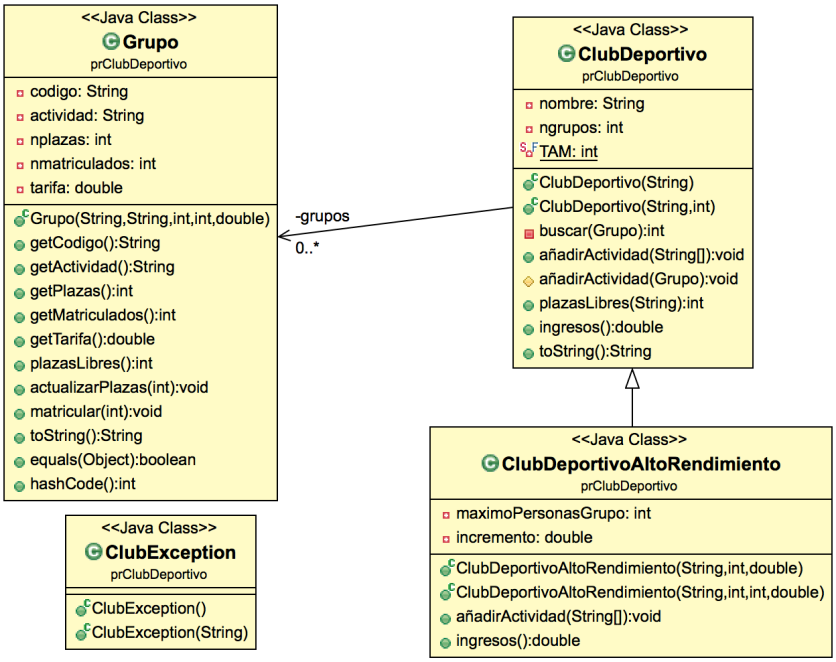
PROGRAMACIÓN ORIENTADA A OBJETOS

CONTROL 7 de MAYO de 2020

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- Al **inicio del contenido de cada fichero** realizado deberá aparecer un comentario con tus **apellidos y nombre, titulación y grupo.**
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, *no debes pararte en él indefinidamente*. Puedes abordar otros.
- **Está permitido:**
  - Consultar los apuntes (CV), la API (Internet), la guía rápida de la API (CV).
  - Añadir métodos privados a las clases.
- **No está permitido**:
  - Intercambiar documentación con otros compañeros.
  - Recibir ayuda de otras personas. Se debe realizar personal e individualmente la solución del ejercicio propuesto.
  - Añadir métodos no privados a las clases.
  - Añadir variables o constantes a las clases.
  - Modificar la visibilidad de las variables, constantes y métodos que aparecen en el diagrama UML.
  - Modificar el código suministrado.
- Una vez terminado el ejercicio, debéis subir (a la tarea creada en el campus virtual para ello) un fichero comprimido de la carpeta **src** que hayáis realizado y usáis vuestros apellidos y nombre para el nombre del mismo **(Apellido1Apellido2Nombre.rar** o **.zip)**.
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización.
- Para la corrección del ejercicio se utilizarán **programas de detección de copias/plagios**.
- Con posterioridad a la realización del ejercicio, el profesor podrá convocar a determinado/as alumno/as para realizar **entrevistas personales síncronas** con objeto de comprobar la autoría de las soluciones entregadas.

Se pide desarrollar una aplicación para gestionar los grupos de actividades deportivas ofertados por un club deportivo. Para ello será necesario construir las clases que se muestran en el siguiente diagrama.



Para realizar el examen se creará un proyecto llamado `prClubDeportivo`. Dentro del proyecto se creará un paquete con el mismo nombre del proyecto, en el que se implementarán las clases: `ClubException`, `Grupo`, `ClubDeportivo` y `ClubDeportivoAltoRendimiento`. Además de las clases del diagrama, en el examen se piden dos clases de prueba, `GrupoMain` y `ClubDeportivoMain`, que se crearán dentro del paquete por defecto. Por último, en el campus virtual se proporciona la clase `PruebaExamen`, para realizar una prueba completa de todas las clases. Esta clase se guardará en el paquete por defecto del proyecto.

NOTA: en todos los lugares del enunciado en los que se indica que se lanza una excepción, será del tipo `ClubException`. Se lanzará con un mensaje explicativo de la situación que ha provocado el error.

1. (0.25 puntos) Clase `ClubException`

Se trata de una excepción no comprobada que se usará en el resto de clases del programa para manejar las diferentes situaciones excepcionales que pudieran producirse.

2. (1.75 puntos) Clase `Grupo`

La clase `Grupo` mantiene información sobre un grupo programado por un club deportivo para realizar una determinada actividad. De cada grupo se almacena: un código, la actividad que se desarrolla, el número de plazas del grupo, el número de personas matriculadas en él y la tarifa de la actividad. Todos estos datos se le pasarán como argumentos al constructor de la clase. El constructor lanzará una excepción si el número de plazas, o el número de personas matriculadas o la tarifa son 0 o negativas. También se lanzará una excepción si el número de matriculados es mayor que el número de plazas.

La clase deberá proporcionar:

- Los métodos necesarios para consultar el valor de cada uno de los atributos de un grupo.
- El método `public int plazasLibres()`, que devuelve el número de plazas libres del grupo.
- El método `public void actualizarPlazas(int n)`, que cambia el número de plazas del grupo por el valor que se le pasa como parámetro. Si el valor del parámetro es menor o igual a 0, o es menor que el número de matriculados actualmente, entonces lanza una excepción.
- El método `public void matricular(int n)`, que suma al número de matriculados del grupo el valor que se le pasa como parámetro. Lanza una excepción si `n` es menor o igual que 0 o no hay suficientes plazas libres en el grupo.
- El método `public String toString()`, que devuelva una cadena de texto que representa la información de un grupo con el siguiente formato:

  `(código - actividad – tarifa` *`euros`* `– P:plazas – M:matriculados)`

  Por ejemplo: `(456B - Pilates - 50.0 euros - P:8 - M:0)`
- Un método que permita comparar dos grupos, teniendo en cuenta que dos grupos se consideran iguales si su código y el nombre de la actividad son los mismos (sin tener en cuenta las mayúsculas y las minúsculas).

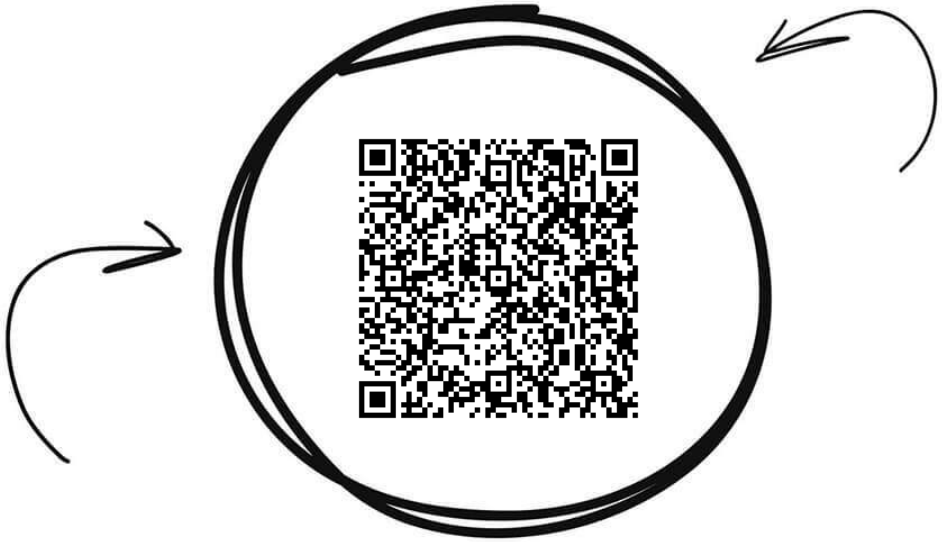
3. (0.5 puntos) Clase `GrupoMain`

La clase `GrupoMain` es una clase distinguida para probar la clase `Grupo`. En el método `main` se deben realizar las siguientes tareas:

1. Crear un grupo con los siguientes datos: código “456B”, actividad “Pilates”, 8 plazas, 5 matriculados y tarifa 50.0 euros, y mostrarlo por pantalla una vez creado.
2. Crear un grupo con los siguientes datos: código “123A”, actividad “Aerobic”, 15 plazas, 10 matriculados y tarifa 30.0 euros, y mostrarlo por pantalla una vez creado.
3. Comparar los dos grupos creados e indicar si son iguales o no.
4. Matricular a 10 alumnos en el grupo 123A y mostrarlo.


EL REINO DEL
PLANETA
DE LOS
SIMIOS
10 DE MAYO SOLO EN CINES
ENTRADAS YA A LA VENTA
©2024 20th Century Studios. Pendiente de calificación por edades.

Programación Orientada a Obj...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

La salida debería ser:

```
(456B - Pilates - 50.0 euros - P:8 - M:5)
(123A - Aerobic - 30.0 euros - P:15 - M:10)
Los grupos son diferentes.
ERROR: no hay plazas libres suficientes.
```

4. (4.75 puntos) Clase `ClubDeportivo`

La clase `ClubDeportivo` mantiene información sobre los grupos ofertados por un club deportivo para realizar distintas actividades. Contiene el nombre del club y un array con los grupos activos. Además, la clase tendrá una variable de instancia (`ngrupos`), que indica el número de grupos que hay en el array en cada momento.

La clase tendrá dos constructores:

- Uno con un único argumento, el nombre del club, que crea el array con un tamaño de 10 (constante `TAM`).
- Otro con dos parámetros, el nombre del club y un número entero, que indicará el tamaño inicial del array. Este constructor lanzará una excepción si el valor del tamaño que recibe es menor o igual que 0.

Además de los constructores, la clase proporcionará los siguientes métodos.

- `private int buscar(Grupo g)`, es un método privado auxiliar que devuelve la posición del grupo `g` dentro del array. Si no está, devuelve -1.
- `public void añadirActividad(String[] datos)`, que recibirá como parámetro un array de `String` con los datos de un grupo en este orden: código, actividad, plazas, matriculados y tarifa. Este método creará un grupo con los datos recibidos como parámetros y llamará al método `protected void añadirActividad(Grupo g)`. Cuando falten datos y/o cuando el formato de las cadenas que representan datos numéricos no sea correcto, se capturarán las excepciones correspondientes lanzadas por el sistema y se lanzará una excepción `ClubException` con el mensaje adecuado.
- `protected void añadirActividad(Grupo g)`, si el grupo que se recibe como parámetro existe en el array, se incrementará su número de plazas con el número de plazas indicado en `g`. En caso contrario, se añadirá el grupo al array. Si el array está lleno, se duplicará su tamaño. Puedes usar el método `int buscar(Grupo g)` antes implementado para el desarrollo de este método.
- `public int plazasLibres(String actividad)`, que recibe como parámetro el nombre de una actividad y devuelve la suma de plazas libres que hay en todos los grupos del club en los que se practica esa actividad.
- `public double ingresos()`, que devuelve los ingresos del club. Estos se calculan sumando el producto de la tarifa por el número de matriculados de cada grupo. Por ejemplo, para un club con estos datos:

  ```
  (456B - Pilates - 50.0 euros - P:12 - M:12)
  (123A - Pilates - 50.0 euros - P:15 - M:12)
  (789C - Danza - 30.0 euros - P:10 - M:5)
  ```

  los ingresos serían `12*50.0+12*50.0+5*30.0 = 1350.0`
- `public String toString()`, que devuelve una cadena de texto que representa la información de un club con el siguiente formato:

  ```
  nombre_del_club --> [ grupo1, grupo2,…, grupoN ]
  ```

  por ejemplo:

  ```
  UMA --> [ (456B – Pilates – 50.0 euros – P:12 – M:12), (123A – Pilates – 50.0
  euros – P:10 – M:8), (789C – Danza – 30.0 euros – P:10 – M:5) ]
  ```

  NOTA: se debe usar la clase `StringBuilder` o la clase `StringJoiner` para este apartado.

5. (0.75 puntos) Clase `ClubDeportivoMain`

Implemente la clase `ClubDeportivoMain` para probar los métodos de la clase `ClubDeportivo`. Se realizarán las siguientes tareas:

1. Crear un club de nombre "UMA" de tamaño 1.
2. Añadir estos dos grupos con los siguientes datos y mostrar el club.
   ```
   String [] grupo1 = {"123A","Pilates","10","10","50.0"};
   String [] grupo2 = {"789C","Danza","10","10","30.0"};
   ```
   La salida sería:
   ```
   UMA --> [ (123A - Pilates - 50.0 euros - P:10 - M:10), (789C - Danza - 30.0 euros - P:10 - M:10) ]
   ```
3. Añadir el grupo3 al club. Esto debería modificar el número de plazas del grupo añadido en el paso 3. Mostrar el club.
   ```
   String [] grupo3 = {"789C","Danza","20","10","30.0"};
   ```
   La salida sería:
   ```
   UMA --> [ (123A - Pilates - 50.0 euros - P:10 - M:0), (789C - Danza - 30.0 euros - P:20 - M:10) ]
   ```
4. Mostrar los ingresos del club.
   ```
   Salida: 800.0
   ```
5. Añadir este grupo con los siguientes datos y mostrar el club.
   ```
   String [] grupo4 = {"456B","Pilates","8Ax","10","50.0"};
   ```
   La salida debería ser: `"ERROR: formato de número incorrecto"`

6. (2 puntos) Clase `ClubDeportivoAltoRendimiento`

La clase `ClubDeportivoAltoRendimiento` representa un club en el que los inscritos demandan entrenamientos intensivos y personalizados, por lo que las actividades se realizan en grupos reducidos. Además, el club está equipado con instalaciones construidas usando las tecnologías más avanzadas en materia deportiva, lo que incrementa el precio de las actividades.

Esta clase hereda la funcionalidad de `ClubDeportivo`, pero tendrá dos nuevas variables: el número máximo de plazas permitidas para un grupo y el porcentaje de incremento del precio de las actividades.

- La clase tendrá dos constructores:
  - Uno con cuatro parámetros: el nombre del club, el tamaño del array, el número máximo de plazas permitidas en los grupos y el porcentaje de incremento sobre el precio de las actividades. Si el tamaño, el número de plazas o el porcentaje son negativos o 0 se lanza una excepción.
  - Uno con tres parámetros: el nombre del club, el número máximo de plazas permitidas en los grupos y el porcentaje de incremento sobre el precio de las actividades. Si el número de plazas o el porcentaje son negativos o 0 se lanza una excepción.
- La clase redefine el método `public void añadirActividad(String[] datos)`, para limitar el número de plazas de un grupo al permitido. Si el número de plazas que se recibe como parámetro es mayor que el permitido, se establece su valor al máximo permitido para el club. Cuando falten datos y/o cuando el formato de las cadenas que representan datos numéricos no sea correcto, se capturarán las excepciones correspondientes lanzadas por el sistema y se lanzará una excepción `ClubException` con el mensaje adecuado.
- Se redefine el método `public double ingresos()`, que aplica el porcentaje de incremento en el precio al total de ingresos del club.

Clase `PruebaExamen`

En el campus virtual podrás encontrar la clase `PruebaExamen`, que sirve para probar las clases implementadas. La salida debería ser la siguiente:

```
PRUEBA 1: crear club deportivo
------------------------------
UMA --> [ (456B - Pilates - 50.0 euros - P:8 - M:5), (123A - Danza - 50.0 euros
- P:10 - M:3), (789C - Pilates - 30.0 euros - P:10 - M:7) ]

PRUEBA 2: calcular ingresos
---------------------------
610.0

PRUEBA 3: crear club deportivo alto rendimiento (num. maximo plazas =5,
incremento de precio=10%)
-------------------------------------------------------------------------------
--
UMA --> [ (456B - Pilates - 50.0 euros - P:5 - M:5), (123A - Danza - 50.0 euros
- P:5 - M:3) ]

PRUEBA 4: calcular ingresos
---------------------------
440.0

PRUEBA 5: intentar aumentar las plazas del grupo de Pilates a 10
-----------------------------------------------------------------
UMA --> [ (456B - Pilates - 50.0 euros - P:5 - M:5), (123A - Danza - 50.0 euros
- P:5 - M:3) ]
```