Segundo Parcial
24/06/2020

Tener en cuenta que:

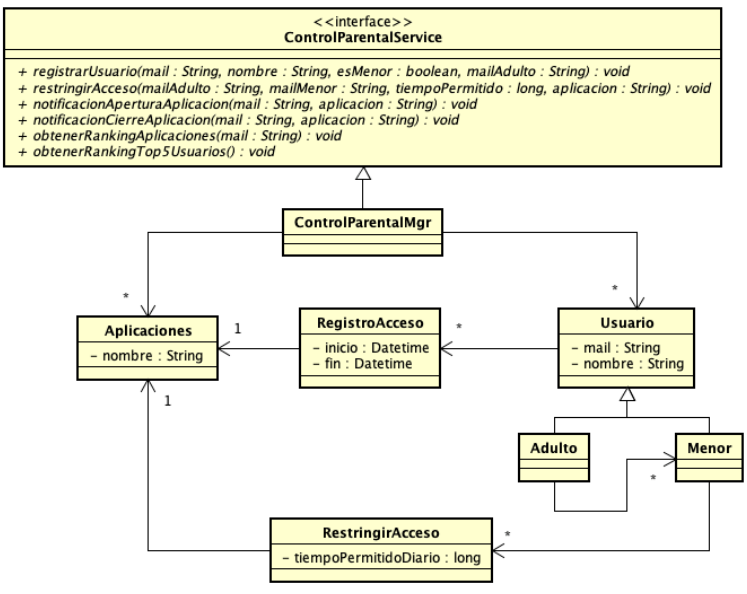
- La duración del parcial es de 2:30 horas.
- Se debe realizar en la computadora, para este fin se deberá realizar un workspace que tendrán que entregar al final el parcial.
- Se entrega un proyecto "Programación 2" que tiene clases de utilidades.
- Culminado las horas del parcial se deberá entregar en el Moodle.
- Tenga en cuenta que NO se tomaran en cuenta para la corrección clases que no compilen.
- La prueba es con material.
- Solo se contestarán dudas sobre la letra, y podrán ser realizadas durante la primer hora del parcial.

Ejercicio Práctico

Se desea desarrollar un sistema que permita controlar el tiempo que una persona esta frente a la pantalla (celular o pc), brindando como resultado reportes que permitan apoyar a los usuarios a conocer su comportamiento y ajustarlo en caso de ser necesario. A su vez si el usuario es un adulto y tiene menores a cargo, el sistema debe permitir registrar a los menores asociados al adulto y configurar cual son la cantidad de horas máxima que el menor puede acceder a cada aplicación por día.

Para lograr esto, cada sistema operativo de un dispositivo brinda mecanismos para notificar cuando una aplicación es abierta y cuando es cerrada por un usuario, validando en el momento de que es abierta si el usuario tiene permisos para accederla. En caso de no tener permisos la aplicación se cierra automáticamente por el sistema operativo. Esto sucedería si el usuario es menor y supero la cantidad de horas diarias o si no tiene acceso a la misma.

Teniendo esto en mente se presenta el siguiente diagrama que modela la realidad planteada:



Se debe implementar la interfaz ControlParentalService y la clase manejador ControlParentalMgr con las siguientes operaciones:

- **void registrarUsuario(String mail, String nombre, boolean esMenor, String mailAdulto)**
  - Esta operación registra un usuario dentro del sistema. El usuario puede ser un adulto o un menor indicado por el argumento "esMenor", en caso de ser adulto el campo "mailAdulto" debe venir vacío, en caso de ser menor el campo mailAdulto es obligatorio e indica el adulto que tiene a cargo el menor.
- **void restringirAcceso(String mailAdulto, String mailMenor, String aplicacion, long tiempoPermitido)**
  - Está operación establece una regla de acceso de un adulto a un menor a cargo para una aplicación. El campo tiempoPermitido puede tomar un valor mayor igual a cero, en caso de ser cero indica que el menor tiene bloqueado el acceso, en caso de ser mayor a cero indica el **tiempo en minutos** diario máximo que se le permite usar la aplicación.
  - En caso de que la aplicación con nombre "aplicacion" no exista se debe crear la aplicación en esta operación.
- **boolean notificacionAperturaAplicacion(String mailUsuario, String aplicacion)**
  - Esta operación es llamada por el sistema operativo para indicar que un usuario intenta abrir una aplicación. En caso de que el usuario no disponga de acceso a esta aplicación se debe retornar false, en caso de que tenga acceso se debe retornar true.
  - Se debe asumir que el horario de comienzo de uso de la aplicación coincide con el instante que es llamado la operación.
- **void notificacionCierreAplicacion(String mailUsuario, String aplicación)**
  - Esta operación es llamada por el sistema operativo para indicar que un usuario ha cerrado una aplicación.
  - Se debe asumir que el horario de cierre de la aplicación coincide con el instante que es llamada esta operación.
- **Lista obtenerRankingAplicaciones(String mailUsuario)**
  - Se debe devolver una lista de todas las aplicaciones usadas por el usuario, con el total de horas de cada una ordenada de mayor a menor.
- **Lista obtenerRankingTop5Usuarios()**
  - Se debe devolver una lista con los 5 usuarios que mayor cantidad de horas enfrente a una pantalla estuvieron, ordenado de mayor a menor.

Restricciones:

- Un usuario debe ser de ser localizado por su clave en O(log n) caso promedio.
- Una aplicación de ser localizada por su clave en O(1) caso promedio.
- Como algoritmo de ordenamiento de las últimas dos operaciones se debe implementar HeapSort.

Simplificación de la realidad:

- Dado que la verificación si un usuario puede abrir una aplicación solo se realiza al abrir la aplicación, puede suceder que al cerrar la aplicación el usuario haya superado el limite de tiempo permitido. Esto es válido, es importante que la próxima vez intente abrir la aplicación no se lo permita.
- Puede asumir que tanto la notificación de apertura de aplicación como de cierre de esta suceden en un mismo día. Dicho de otra manera, no puede suceder que la aplicación se abra en un día y se cierre en el siguiente.

- En caso de recibir la notificación que se abre una aplicación que ya se encuentra abierta debe ignorarse, como también si recibe una notificación de cierre de una aplicación que ya se encuentre cerrada.

Consideraciones:

- En cada operación se debe agregar las excepciones que considere necesarias.
- Para la implementación de todo el problema puede asumir que no hay restricciones de memoria.
- Las clases de la solución deben estar dentro del paquete sparcial.
- Se debe seleccionar el TAD más adecuado dado el tipo de problema planteado. Una incorrecta selección no fundamentada, será considerada como incorrecta.
- Se debe realizar test unitarios JUnit para todas las operaciones.
- Para el manejo de fechas y horas debe usar la clase LocalDateTime información útil:
    - JavaDoc:
        - https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html
    - Ejemplos de uso:

```
// Obtener la fecha y hora de este instante
LocalDateTime now = LocalDateTime.now();

// Obtener la fecha y hora con datos especificos
LocalDateTime certainDate = LocalDateTime.of(2020, 6, 24, 16,
0, 0);

// Obtener la diferencia en minutos de dos fechas
long minutes = Duration.between(certainDate, now).toMinutes();

// Obtener solamente la fecha de un datetime
LocalDate fecha = now.toLocalDate();
```