UTN SANTA FE

Diseño e Implementación

De Estructuras de Datos

Guía 08 – Práctica

Contenido

Guía 08

Realizar un fork del proyecto https://github.com/died-utn/guia08 y resolverlo en su propio repositorio implementando los métodos solicitados.

Problema 01: Recursos humanos – ENTREGA OBLIGATORIA 29/5

1. Un sistema de gestión de recursos humanos registra los empleados de una empresa y permite asignar tareas a empleados asignados a distintos proyectos.
2. De las tareas se registra el id, una descripción, la duración estimada, una bandera booleana que marca si ya fue facturada, las fechas de inicio y fin, el empleado asignado y la cantidad de horas estimadas.
3. De un empleado se conoce el tipo de empleado (puede ser contratado o efectivo), el valor hora de trabajo y la fecha de contratación, además de la lista de tareas que tiene asignadas.
4. Un empleado puede ser asignado a un proyecto según las siguientes condiciones
    a. Si es contratado, no puede tener más de 5 tareas asignadas pendientes de finalizar.
    b. Si es Efectivo, no puede tener asignadas, tareas pendientes de finalizar que sumen más de 15 horas de trabajo estimadas.
    c. En cualquiera de los dos casos si el empleado no puede ser asignado se retorna un falso.
    d. En caso de que se quiera asignar un empleado a una tarea que ya tiene otro empleado asignado, o a una tarea que ya fue finalizada, entonces se lanza una excepción indicando que la tarea que se quiere asignar es incorrecta que se seleccione otra tarea.
5. Lo que percibe un empleado por una tarea también depende del tipo de empleado

a. Ambos tipos de empleados cobran la duración estimada de la tarea multiplicado por el costo hora.
b. Además, cobran un extra si terminan la tarea antes de lo planificado. Para determinar esto se calcula la diferencia en días, entre la fecha de inicio y la fecha final de la tarea, y se multiplica por 4 horas teóricas de trabajo diaria. Entonces si una tarea estaba estimada en 10 horas, y fue finalizada en 2 días, se asume que el empleado tardó 8 horas en resolverla.
c. Los empleados efectivos aumentan el valor de su hora en un 20% en aquellas tareas que finalizaron antes.
d. Los empleados contratados aumentan el valor de su hora en un 30 % para estas tareas. En cambio, si un empleado contratado se demora más de 2 días, en finalizar su tarea respecto a lo que estaba estimado cobrará un 75% de su valor hora.
e.

6. La aplicación de Recursos humanos debe:
   a. Cargar los datos de los empleados de un archivo CSV. Deberá implementar los métodos que leen el archivo y los métodos que asignan empleados a la lista de empleados.
   b. Cargar los datos de las tareas de un archivo CSV y asignar las tareas al empleado en cuestión.
   c. Periódicamente 'se ejecuta el método facturar que permite calcular el monto total a facturar por todas las tareas realizadas y guarda un archivo csv con las tareas facturadas.

Tareas a resolver:

1. Agregar a las clase Empleado y Tarea los constructores y métodos getters y setters que necesite.
2. En la clase Empleado implementar
   a. `public Boolean asignarTarea(Tarea t)`
      i. **según lo indicado en el item 4**
   b. `public Double salario()`
      i. **según lo indicado en el item 5**
   c. `public void comenzar(Integer idTarea)`
      i. **Lógica a aplicar: buscar la tarea en la lista de tareas asignadas. Si la tarea no existe lanza una excepción personalizada que deberá crear con el mensaje apropiado. Si la tarea existe indica como fecha de inicio la fecha y hora actual**
   d. `public void finalizar(Integer idTarea)`
      i. **Lógica a aplicar: buscar la tarea en la lista de tareas asignadas. Si la tarea no existe lanza una excepción personalizada que deberá crear con el mensaje apropiado. Si la tarea existe indica como fecha de fin la fecha y hora actual**
   e. `public void comenzar(Integer idTarea,String fecha)`
      i. **Sobrecarga el método comenzar pero permite recibir una fecha en formato "DD-MM-YYYY HH:MM"**
   f. `public void finalizar(Integer idTarea,String fecha)`
      i. **Sobrecarga el método finalizar pero permite recibir una fecha en formato "DD-MM-YYYY HH:MM"**

3. En la clase Tarea implementar el método
    a. `public void asignarEmpleado(Empleado e)`
        i. Asigna el empleado a la tarea. Si la tarea ya tiene un empleado asignado y fue finalizada, es decir tiene fecha de finalizado disntinto de null, debe lanzar una excepción personalizada con un mensaje apropiado
4. En la clase AppRRHH implementar los siguientes métodos:
    a. `public void agregarEmpleadoContratado(Integer cuil,String nombre,Double costoHora)`
        i. Crea un empleado cuya lógica de asignación de tarea y calculo de costo de tarea corresponde con la de un contratado y lo agrega a la lista de empleados.
    b. `public void agregarEmpleadoEfectivo(Integer cuil,String nombre,Double costoHora`
        i. Crea un empleado cuya lógica de asignación de tarea y calculo de costo de tarea corresponde con la de un contratado y lo agrega a la lista de empleados.
    c. `public void asignarTarea(Integer cuil,Integer idTarea,String descripcion,Integer duracionEstimada) :`
        i. Busca un empleado por cuil, si lo encuentra crea la tarea y se la asigna.
    d. `public void empezarTarea(Integer cuil,Integer idTarea) {`
        i. Busca un empleado por cuil, si lo encuentra le indica que la termine. Gestionar las excecpiones posibles.
    e. `public void terminarTarea(Integer cuil,Integer idTarea)`
        i. Busca un empleado por cuil, si lo encuentra le indica que la termine. Gestionar las excecpiones posibles.
    f. `public void cargarEmpleadosContratadosCSV(String nombreArchivo)`
    g. `public void cargarEmpleadosEfectivosCSV(String nombreArchivo)`
        i. los dos métodos anteriores cargan los empleados de un archivo CSV donde se indica:
            1. CUIL
            2. NOMBRE
            3. COSTO HORA
    h. `public void cargarTareasCSV(String nombreArchivo)`
        i. Este método carga las tareas de una lista de tareas donde se indica
            1. ID
            2. DESCRIPCION
            3. DURACION ESTIMADA
            4. CUIL EMPLEADO ASIGNAR
    i. `private void guardarTareasTerminadasCSV()`
        i. Este método guarda todas las tareas que fueron terminadas y no fueron facturadas en un archivo CSV indicando la información de la tarea, el CUIL del empleado que la finalizo y el nombre del mismo.

Se considerará apropiado implementar los métodos de testing en el paquete de pruebas.

Problema 02 - Liga de futbol

1. Se necesita modelar información de una **liga** de futbol que tiene hasta 10 equipos. Para ello, se registran **Equipos**. De cada Equipo se conoce el nombre, y la lista de **jugadores**. Cada equipo puede tener inscriptos hasta 21 Jugadores. La liga, registra en una lista de **Fechas**, cada uno de los **Partidos**, donde se enfrentan el equipo local y el equipo visitante. De cada partido se registra información del Dia y hora del partido, y quienes fueron los jugadores 11 jugadores titulares y 7 suplentes que firmaron la planilla ese día. De cada partido también se registran **Eventos**, indicando el tipo de evento, que jugador está asociado a dicho evento y en que minuto del partido fue el evento.
2. Los tipos de evento que se registran son

a) Infracción: jugador que la cometió jugador que la recibió.
b) Tarjeta Amarilla o Roja: jugador que la recibió y una descripción con el motivo
c) Tiro de esquina: Jugador que lo ejecutó y costado en que se produjo (derecho o izquierdo)
d) Tiro Libre: Jugador que lo ejecutó y una descripción con el resultado de la acción.
e) Gol: jugador que lo marco, jugador que asistió, y en caso de ser un gol en el arco propio, se marca con un flag en true

3. Crear la clase Jugador donde se registre información del jugador, (nombre, dni, fecha de nacimiento, posición, altura, peso, si patea con el pie izquierdo o derecho)
4. Crear la clase Equipo que permita registrar:
   a) Información del equipo
   b) Agregar hasta 21 jugadores, siguiendo la siguiente regla:
      i. Hasta 3 arqueros.
      ii. Hasta 9 defensores
      iii. Hasta 9 volantes
      iv. Hasta 6 delanteros.
5. Crear la clase Liga a la que se le pueda:

a) Agregar, quitar, consultar, y modificar Equipos.
b) Agregar una lista con las fechas a disputar y los encuentros que se disputan en cada fecha.
c) Registrar en cada fecha los partidos.
d) Registrar en cada partido los eventos que se han producido.

Métodos a implementar

a) Imprimir la tabla de posiciones y guardarla en un archivo con la fecha de impresión.
b) Imprimir la tabla de goleadores.
c) Imprimir la lista de jugadores por el siguiente orden
   a. Partidos jugados
   b. Goles realizados
   c. Amonestaciones recibidas.

   d. Altura
   e. Peso

d) Imprimir la lista de equipos ordenados por
   a. Con mayor cantidad de goles a favor
   b. Con mayor diferencia de gol
   c. Con mayor cantidad de amonestados y expulsados
   d. Con mayor cantidad de infracciones.

e) Buscar en los eventos, si existe:
   a. Un evento donde el Jugador A, le dio una asistencia al Jugador B
   b. Un evento donde el jugador A le cometió una falta al jugador B
   c. Un evento donde el jugador A recibió una tarjeta en un lapso de tiempo determinado dentro del partido (por ejemplo entre los 70 y 90 minutos)