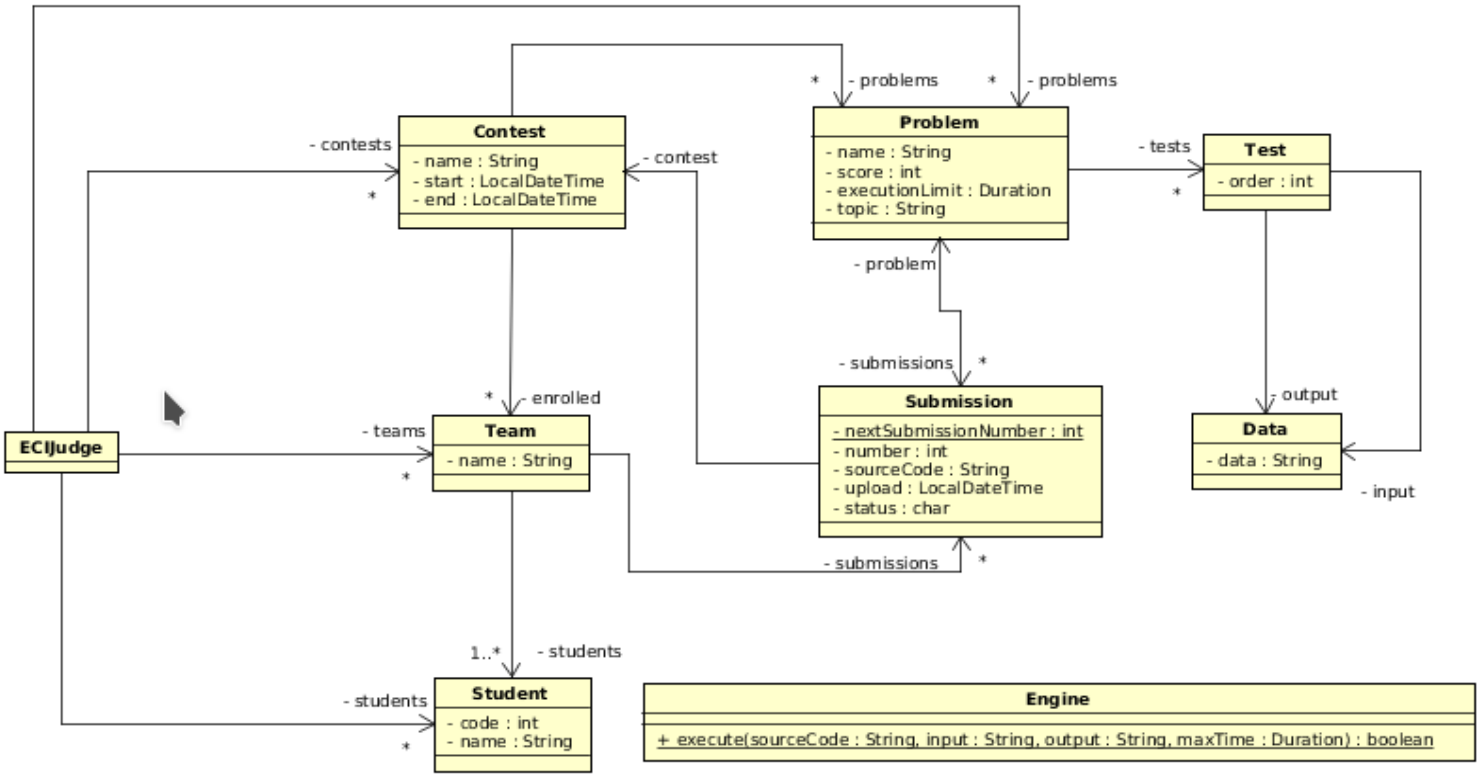
**ESCUELA COLOMBIANA DE INGENIERÍA**
**PROGRAMACIÓN ORIENTADA A OBJETOS**
**Parcial primer tercio. Nota esperada: _____**
**S06: 2020-01**

## ECIJudge

La decanatura de Ingeniería de Sistemas de la ECI quiere ofrecer a los estudiantes un sistema de competencias de programación. La intención es que los estudiantes aprendan a programar mediante la solución de problemas con diferentes niveles de dificultad.
Cada competencia está conformada por diferentes problemas de acuerdo a las necesidades de aprendizaje que se requieran. A mayor complejidad de un problema, mayor será su puntaje. En cada competencia participan diferentes equipos los cuales están conformados por uno o más estudiantes. Cada equipo realiza las soluciones a los problemas planteados en sus máquinas locales y cuando creen tener la solución correcta la suben a **ECIJudge**. Un equipo puede hacer tantas subidas de soluciones como requiera. Para aceptar una solución como correcta, esta debe pasar el conjunto de pruebas asociado al problema respectivo. Cada prueba tiene un orden establecido de ejecución (de la más simple a la más compleja), una entrada y una salida esperada. Una prueba pasa si la salida de la ejecución de la solución coincide con la salida esperada.

**Creación de competencias:** La ECI es la responsable de crear las diferentes competencias y los problemas y pruebas respectivas.
**Registro de soluciones:** Cuando un equipo cree tener la solución, sube el código a ECIJudge para ser evaluado. La solución es evaluada de acuerdo al orden de llegada. Cada solución puede estar en uno de los siguientes estados: Pendiente, Rechazada y Aceptada. (P)ending, R)ejected, A)ccepted)
**Cálculo de tabla de posiciones:** Cada cierto intervalo de tiempo, se realiza una actualización de la tabla de posiciones asociada a una competencia. En dicha tabla de posiciones aparecen los equipos inscritos en la competencia ordenados de forma descendente por la cantidad de puntos acumulados en sus soluciones a los problemas de la competencia.




(Los contenedores de ECIJudge son **HashMap**
Los otros son **ArrayList**)

| *Class HashMap<K,V>* | |
|---|---|
| V | **get(Object key)**<br>Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key. |
| V | **put(K key, V value)**<br>Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced. |
| V | `remove(Object key)`<br>Removes the mapping for the specified key from this map if present. |

| *Class LocalDateTime* | |
|---|---|
| boolean | `isAfter(ChronoLocalDateTime<?> other)`<br>Checks if this date-time is after the specified date-time. |
| static LocalDateTime | **now()**<br>Obtains the current date-time from the system clock in the default time-zone. |

| *Class Duration* | |
|---|---|
| int | **toNanos()**<br>Converts this duration to the total length in nanoseconds expressed as a `long` |

## (25%) MEMORIA

Presente el mapa de memoria correspondiente a:

- La competencia “Trees Algorithms” inicio el 03/02/2020. Esta competencia tiene dos problemas. Un problema llamado “DFS” que otorga 5 puntos y pertenece al tema “Recursion”. “DFS” tiene 1 test configurado. La entrada de dicho test es “SortingTest” y la salida “eginorssttt”. El otro problema es “BFS” que otorga 2 puntos con tema “Recursion”.
- El equipo “assemblers” subió una solución, para el problema DFS del contest “Trees Algorithms”, identificada con el número 1 y código fuente “print(‘test’)”. Dicha solución se encuentra en estado pendiente.

## (35%) CÓDIGO

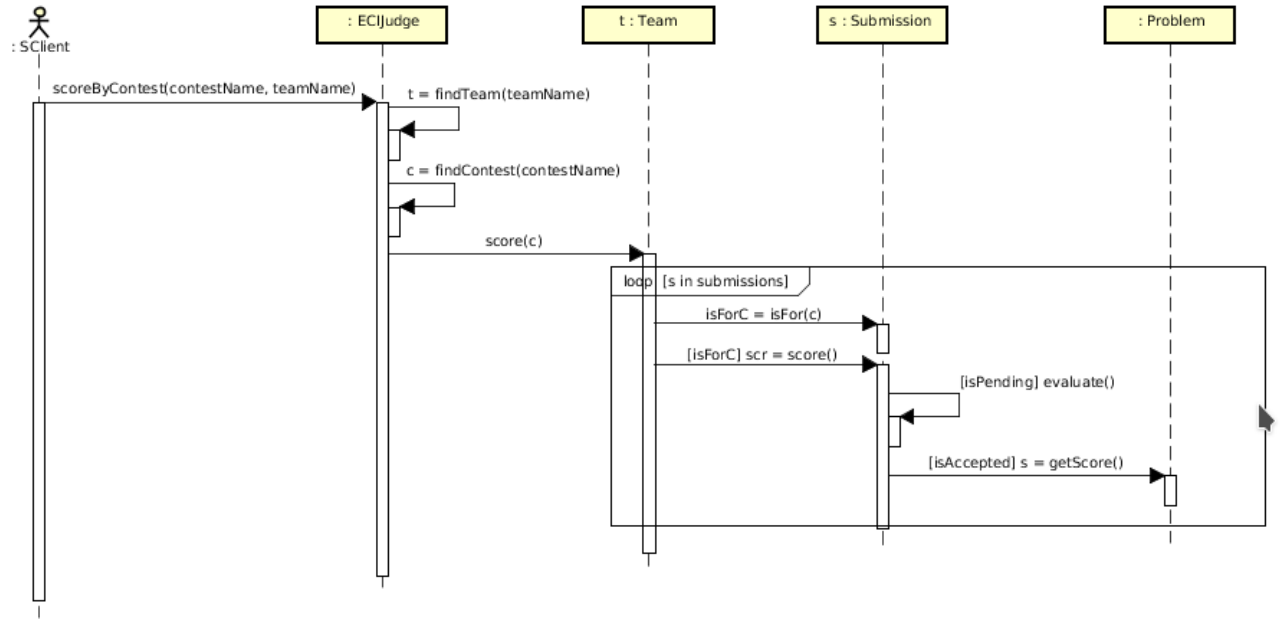
Desarrolle el método especificado y diseñado siguiendo MDD. (Debe seguir los puntos MDD y sólo construir lo que está diseñado)

**MDD**

1. Estudie el diagrama de secuencia y la especificación (documentación + encabezado) del método
2. Actualice el diagrama de clase con los nuevos elementos
3. Escriba el código de la clase responsable inicial (encabezado y atributos). Documente el invariante.
4. Implemente cada uno de los métodos correspondientes a la solución. Indique el encabezado de la clase en la que está escribiendo (no incluya los atributos) e incluya la documentación (si no está documentado). **No construya ni documente los básicos (get, set, is)**

En **ECIJudge**

```
public int scoreByContest(String contestName, String teamName)
Calculates the score of a team on a specific contest
Parameters:
        contestName  – Name of the contest
        teamName      – Name of the team
Returns:
        Accumulated score of the team on the specific contest
```




## (20%) DISEÑO

Diseñe un método de la clase **ECIJudge** (especificación y diagrama de secuencia decorado) que permita encontrar el nombre del estudiante que forma un equipo de un solo integrante y que tiene el mayor score en una competencia específica. No olvide validar las condiciones que se necesitan para hacer el cálculo. (Código robusto)

## (20%) CONCEPTOS

1. ¿Cómo clasificamos los modificadores en POOB? ¿Cuales pertenecen a cada grupo? ¿Cuando usar cada modificador?

2. ¿Qué es la sobrecarga de métodos? ¿Que se usa para identificar el método que se debe ejecutar? ¿De que se compone eso que se usa para identificar el método que se debe ejecutar?