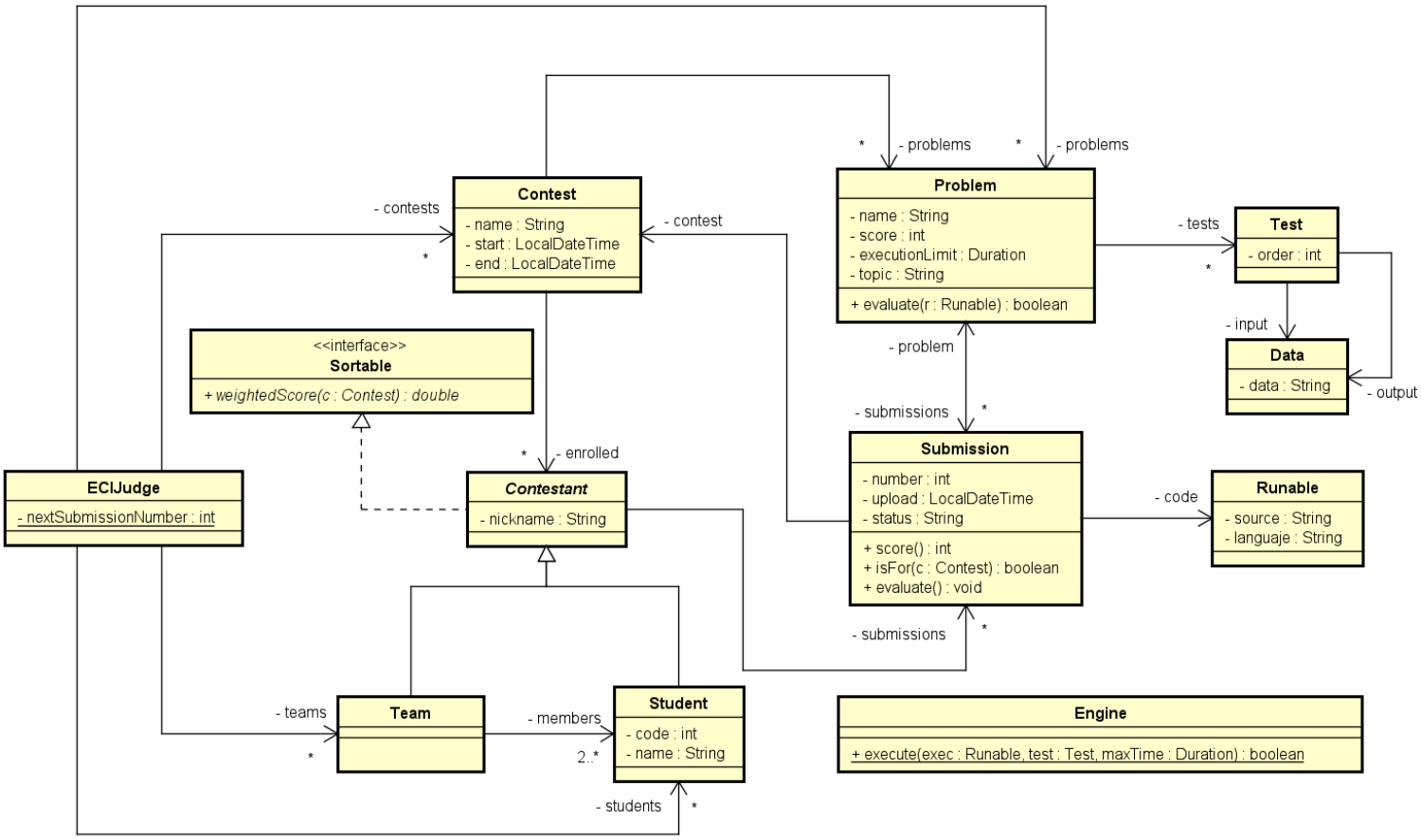
ESCUELA COLOMBIANA DE INGENIERÍA
PROGRAMACIÓN ORIENTADA A OBJETOS
Parcial segundo tercio. Nota esperada: _____
S11: 2020-01

ECIJudge

La **Escuela** quiere extender el sistema desarrollado para las compentencias de programación. **ECIJudge** ahora debe permitir (i) que en las competencias existan dos tipos de participantes: estudiantes o equipos (ii) considerar en el cálculo de la tabla de posiciones de los participantes el número de estudiantes que los conforman (iii) incluir diferentes lenguajes de programación.



(Tod
os los contenedores son **HashMap**)

Interface Sortable	
double	**`weightedScore`(`Contest` c)** Calculates the weighted score of the sortable in a specific contest. -- NO_ENROLLED If the contestant is no enrolled in the contest NO_VALID If the contestant is invalid NO_TESTS If any problem has no tests

Class Pair<K, V>	
	_(K first, V second) Creates a Pair with the specified components

I. (35%) PREPARANDO SORTABLE

Implemente los métodos necesarios para cumplir con los compromisos asociados a la interfaz `Sortable`. No olvide el manejo de excepciones. Esta interfaz va a permitir que participantes (Contestant) de una misma competencia (Contest) pueden ser ordenados de acuerdo al puntaje ponderado.

NOTAS:

- Debe presentar el diseño **COMPLETO** (NO OLVIDE MDD) del método **weightedScore** teniendo en cuenta la especificación del método.
- Debe incluir el diseño del método evalue ya que este debe cambiar cambia por la inclusión de los diferentes lenguajes de programación. Recuerde que evalue es el encargado de cambiar el estado de la solución: P)ending, R)ejected y A)ccepted.

MDD
1. Complete la documentación del método
2. Realice el(los) diagrama(s) de secuencia (adicione el manejo de excepciones con otro color)
2. Actualice el diagrama de clases con los nuevos elementos
3. Escriba las fuentes completas (código y documentación) correspondientes a la solución. Menos **Engine**).

II. (20%) DISEÑANDO

Diseñe el siguiente método.

MDD
1. Estudie la especificación (documentación + encabezado) del método
2. Realice el diagrama de secuencia (adicione el manejo de excepciones con otro color)
3. Actualice el diagrama de clases con los nuevos elementos

En ECIJudge

public List<Pair<Contestant, Double>> scoreBoard(**`String`** `contestName`) **throws** ECIJudgeException

Calculates the score board of a contest based on the weighted score.

Parameters:

name – The contest's name

Returns:

The sorted list of pairs <contestant, weighted score>

Throws:

ECIJudgeException - NO_ENROLLED If the contest has no enrolled contestants
ECIJudgeException - NO_TESTS If any problem has no tests

III. (25%) EXTENDIENDO

Se desea extender el sistema para permitir que se puedan realizar soluciones en lenguajes gráficos. Por tanto, las soluciones podrán tener código fuente (String) o un diagrama de flujo (Nuevo tipo: **co.edu.eci.util.Flow**). Se necesita adicionar la capacidad de ejecutar este nuevo tipo de solución. Pista: varios tipos de Engine

- Realice los cambios necesarios en el diagrama de clases.
- Implemente dichos cambios, solo los estructurales.
- Actualice el diseño del método **evaluate** en donde sea necesario ¿Debería lanzar alguna excepción?. No olvide agregar los métodos al diagrama de clases.

IV. (20%) CONCEPTOS

1. ¿Diferencias entre excepciones chequeadas y no chequeadas? ¿Deberíamos atrapar una excepción chequeada (argumente)?
2. ¿Diferencias entre Clase abstracta e interfaz? ¿Tipos de métodos en cada una? ¿Cuando usar una clase abstracta o una interfaz?