

# Bienvenido a tu Certificación

Ya estás un paso más cerca de ser un Analista del Conocimiento - Dimensión Programador!

El examen consta de 6 ejercicios basados en los conocimientos exigidos por el 111 Mil y por la industria, para convertirte en programador junior. Tené en cuenta y leé con mucha atención las siguientes pautas para la correcta realización y aprobación del examen:

1- El examen tiene una duración máxima de 3 (tres) horas. Deberás enviar los resultados antes de cumplir ese tiempo.

2- Es necesario responder TODOS los ejercicios para poder aprobar el examen.

3- El resultado es APROBADO o DESAPROBADO, no tiene puntaje.

4- Los veedores estarán presentes para verificar que el examen se tome en las condiciones adecuadas.

5- Está prohibido utilizar el chat, el correo electrónico (fuera del uso normal para abrir este formulario), o cualquier página web que no sea este Google Form.

6- Tené presente que el teclado no reconoce la tecla "TAB" para escribir código Java, por lo que deberás usar 3 (tres) veces la tecla "ESPACIO" para poner sangrías (indentar/tabular) al alinear el código.

7- En caso de no encontrar el tipeo de una tecla, podrás usar el mapa de caracteres, que podés encontrarlo en: "Tecla Windows + R" y en el cuadro de texto escribir "charmap", y apretar "Enter". También podés encontrarlo en "Inicio--> Accesorios ---> Herramientas del Sistema ---> Mapa de caracteres".

8- Cuando en el punto siguiente el formulario te pida el código de seguridad, tenés que solicitárselo a tu veedor.

Te deseamos mucha suerte y a trabajar en los ejercicios!

**\*Obligatorio**

**Dirección de correo electrónico \***

Tu dirección de correo electrónico

**Esta pregunta es obligatoria**

**Datos Personales**

**DNI (sin puntos) \***

Tu respuesta

**Apellidos \***

Tu respuesta

**Nombres \***

Tu respuesta

**Fecha de Nacimiento \***

Fecha

**Situación de Examen \***

SIN CURSADA - LIBRE

Realicé el curso del Plan 111Mil

**Sede de Certificación \***

Elige

**Solicitud de Certificado de Asistencia \***

En caso de necesitar un certificado que indique que estuviste presente en el examen para tu trabajo u otra organización indicalo aquí. El mismo te llegará en forma digital, al correo electrónico que indicaste al comienzo de este formulario

Sí

No

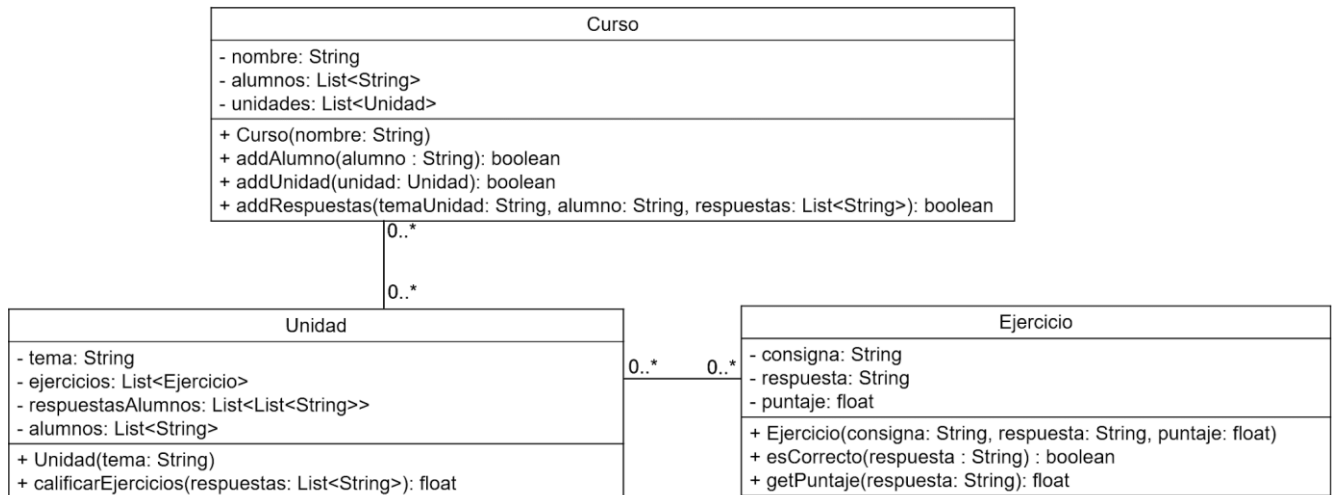
## Contexto del examen - Unidad y Ejercicios

Sebastián es profesor de informática. Al principio, cuando tenía un solo curso de alumnos, Sebastián podía fácilmente tener registro de las unidades y los ejercicios que daba a sus alumnos. Sin embargo, en los últimos tiempos Sebastián empezó a dar clases en más cursos, de modo que ahora necesita una forma fácil de mantener organizadas las unidades temáticas que enseña en cada curso, los ejercicios que da a los alumnos para cada unidad, qué alumnos resolvieron los ejercicios y cómo les fue a los alumnos con las resoluciones. Como sabe que que cursamos el programa 111Mil se puso en contacto con nosotros para que lo ayudemos a construir el sistema, que en esta primera etapa del proyecto deberá registrar las unidades, los ejercicios que lo componen y las resoluciones de los alumnos.

## Ejercicio 1 - Implementar desde el diagrama de clases

Dado el siguiente diagrama UML del sistema, implementar la clase Unidad. Para ello tener en cuenta que:

- Las listas se deben inicializar vacías dentro del constructor.
- El método `calificarEjercicios` debe sumar los puntajes obtenidos para cada una de las respuestas en la lista pasada por parámetro. Asumir que los ejercicios y las respuestas se encuentran en la misma posición de las listas. Por ejemplo, la respuesta en la posición 0 se corresponde con el ejercicio de la posición 0. Para esto se puede utilizar el método `getPuntaje` de la clase `Ejercicio`, el cual dada una respuesta retorna el puntaje correspondiente.



## Implemente la clase Unidad

Tu respuesta



## Ejercicio 2 - Implementar una funcionalidad a partir de un enunciado

A Sebastián le gustó la primera implementación del programa que le resuelve sus problemas. Pero se le ocurrió otra funcionalidad que quiere agregar al sistema:

- Dado un tema de unidad y una calificación, obtener la lista de alumnos del curso que no obtuvieron una calificación mayor a la dada.

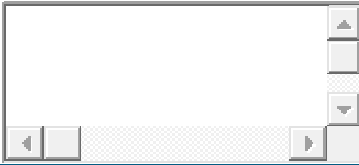
Implemente la funcionalidad indicando a qué clase corresponde el/los método/s que satisface/n dicha funcionalidad.

Tenga en cuenta que la clase Unidad además de los métodos implementados en el Ejercicio 1 cuenta con los siguientes métodos:

- El método getCalificacion de la clase Unidad retorna -1 si el alumno pasado por parámetro no resolvió los ejercicios correspondientes, o un valor mayor o igual a cero si los resolvió.
- El método getTema de Unidad retorna el tema de la unidad.
- El método esDeTema de Unidad retorna verdadero si el tema pasado por parámetro es igual al tema de la Unidad.

Indique en qué clase/s iría dicha funcionalidad e implemente el/los método/s necesarios para satisfacer este nuevo requerimiento

Tu respuesta



## Ejercicio 3 - Seguimiento de Código

Considerando el siguiente método en la clase Curso:

```
public List<String> xxx(String alumno){  
  
    List<String> xx = new ArrayList<String>();  
    Iterator<Unidad> it = unidades.iterator();  
    while(it.hasNext()){  
        Unidad u = it.next();  
        if(u.getCalificacion(alumno) > -1)  
            xx.add(u.getTema());  
    }  
    return xx;  
}
```

¿Qué se imprimirá al ejecutar el siguiente código? Tenga en cuenta que:

- El método `getCalificacion` de la clase `Unidad` retorna -1 si el alumno no resolvió los ejercicios de la unidad, y un valor mayor o igual a cero si los resolvió.
- El método `getTema` de `Unidad` retorna el tema de la unidad.
- El método `addRespuestas` de `Curso` busca la unidad del tema correspondiente e invoca el método `addRespuestas` de dicha unidad con el alumno y las respuestas pasadas por parámetro.

```
Curso c = new Curso("Computación Primero");  
c.addAlumno("pedro");  
c.addAlumno("luis");
```

```
Unidad u1 = new Unidad("uso de procesador de texto");  
u1.addEjercicio(ejercicio1); //asumir que ejercicio1 ya se encuentra creado  
u1.addEjercicio(ejercicio2); //asumir que ejercicio2 ya se encuentra creado
```

```
Unidad u2= new Unidad("uso de planilla de cálculo");  
u2.addEjercicio(ejercicio3); //asumir que ejercicio3 ya se encuentra creado  
u2.addEjercicio(ejercicio4); //asumir que ejercicio4 ya se encuentra creado
```

```
c.addUnidad(u1);  
c.addUnidad(u2);
```

```
c.addRespuestas("uso de procesador de texto", "pedro", respuestasPedro); //asumir que  
respuestasPedro ya se encuentra creada
```

```
c.addRespuestas("uso de planilla de cálculo", "luis", respuestasLuis); //asumir que
respuestasLuis ya se encuentra creada
c.addRespuestas("uso de planilla de cálculo", "luis", respuestasLuis); //asumir que
respuestasLuis ya se encuentra creada
```

```
c.addRespuestas("uso de procesador de texto", "luis", respuestasLuisTexto); //asumir que
respuestasLuisTexto ya se encuentra creada
```

```
System.out.println(c.xxx("luis"));
System.out.println(c.xxx("pedro"));
```

## Qué imprime?

Tu respuesta

## Ejercicio 4 - Documentación con Javadoc

Elaborar la documentación técnica utilizando Javadoc del método `calificarEjercicios` implementado en el Ejercicio 1. Incluya tanto la descripción del método como los tags que correspondan.

## Javadoc

Tu respuesta

## Ejercicio 5 - Base de Datos

Se ha modelado el siguiente diagrama de Entidad Relación.



## Consulta SQL

Dado el diagrama de entidad-relación parcial correspondiente al sistema que estamos desarrollando, Sebastián quiso obtener para los alumnos que resolvieron más de dos

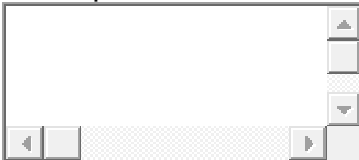
ejercicios de la unidad del tema “uso de procesador de texto”, su nombre y cantidad de ejercicios resueltos. Los alumnos deben estar ordenados de forma descendente de acuerdo a la cantidad de ejercicios resueltos. Para ello, Sebastián escribió la siguiente consulta:

```
SELECT alumno FROM respuesta
INNER JOIN ejercicio ON respuesta.ejercicio_idEjercicio = ejercicio.idEjercicio
WHERE ejercicio.unidad_idUnidad
IN (SELECT idUnidad FROM unidad
WHERE tema <> "uso de procesador de texto")
ORDER BY alumno
```

Sin embargo, la consulta no obtiene lo que Sebastián quiere. Modificar la consulta para cumplir con la especificación de Sebastián.

## SQL

Tu respuesta



## Ejercicio 6 - Mapeo a Hibernate

Dado el diagrama de entidad-relación presentado en el ejercicio anterior y el diagrama UML presentado en el ejercicio 1, escriba la línea del archivo de mapeo de Hibernate (en formato XML o anotación) correspondiente al mapeo del atributo “consigna” de la clase y tabla Ejercicio.

## Hibernate

Tu respuesta



**SIGUIENTE**