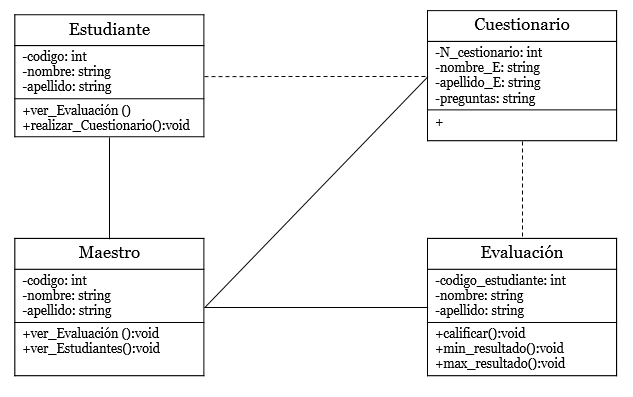
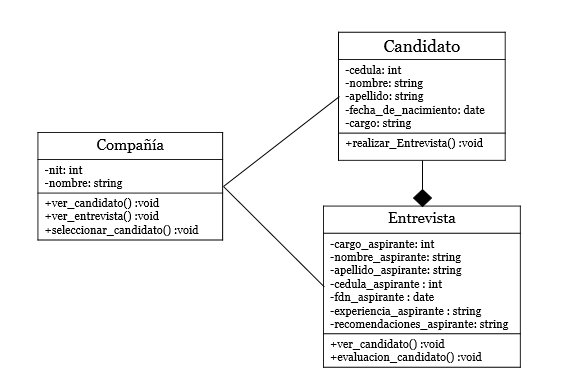
**TAREA 1.**

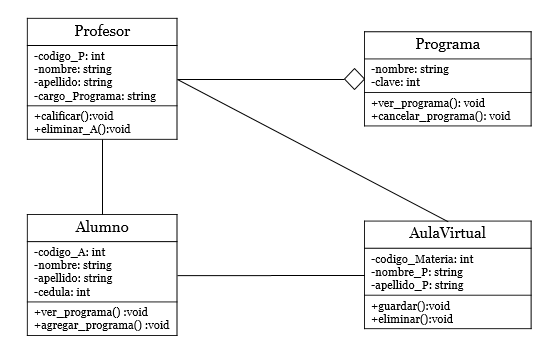
1. Casos de Uso (UML)
2. Un maestro de ciencias basicas (Matematicas, Fisica, Quimica) tiene un salon por cada estudiantes el evaluará sus conocimientos en un cuestionario y el tiempo en el que se demora en cada una de las diferentes preguntas, a partir de eso escogerá a 5 estudiantes para participar en una maratón de ciencias básicas.



1. Dentro de una compañía se necesitan contratar a jefes de las diferentes zonas (Administracion, Recursos Humanos, tecnología), para ello 20 candidatos tendrán una entrevista donde 4 serán elegidos al finalizar el total de las entrevistas.



1. En un programa el profesor Arq. y Pat. de Software debe inscribir a sus alumno en el aula virtual, donde se necesita una clave del profesor y del alumno, cada alumno tendrá un nombre, apellido, documento (T.I. o cédula) y fecha de nacimiento.



2. ¿Qué es un patrón de Software?

* Cada uno de ellos define la solución para resolver un determinado problema, facilitando además la reutilización del código fuente.
* **Patrones de creación**: utilizados para crear y configurar de clases y objetos.
* **Patrones estructurales**: su objetivo es desacoplar las interfaces e implementar clases y objetos. Crean grupos de objetos.
* **Patrones de comportamiento**: se centran en la interacción entre asociaciones de clases y objetos definiendo cómo se comunican entre sí.

3. ¿Qué es Arquitectura de Software?

* Forma de trabajar en un sistema, como construir nuevos módulos, pero también debe dejar intuir el tipo de aplicación que describe. Nos ayuda a entender mejor de qué trata nuestro software, a centrarnos en el dominio de nuestra aplicación, que es el valor real y que al fin y al cabo es la razón que nos lleva a escribir software.

4. Características de la Arquitectura de Software.

* Parte del diseño de software.
* Nivel del diseño de software donde se definen la estructura y propiedades globales del sistema.
* Incluye sus componentes, las propiedades observables de dichos componentes y las relaciones que se establecen entre ellos.
* Un aspecto crítico: Una arquitectura errónea puede llevar a problemas incontables.
* Representación de alto nivel de la estructura del sistema describiendo las partes que lo integran.
* Puede incluir los patrones que supervisan la composición de sus componentes y las restricciones al aplicar los patrones.
* Trata aspectos del diseño y desarrollo que no pueden tratarse adecuadamente dentro de los módulos que forman el sistema.

5. ¿Qué es TDD?

* *Test Driven Development* es una metodología donde hacemos todo al revés. Por un momento vamos a **dejar de programar** para dedicarnos a **escribir las pruebas**.
* Gracias a TDD podemos agilizar nuestro proceso de creación de código. **Nos preocupamos más por el qué y el por qué antes del cómo**. Todo el código que necesitamos para implementar eso que planeamos con las pruebas será problema un poco más adelante. Por ahora, lo que importa es asegurarnos que estamos resolviendo los problemas correctos.
* Dedicarnos a escribir las pruebas.
* Gracias a TDD podemos agilizar nuestro proceso de creación de código. Nos preocupamos más por el qué y el por qué antes del cómo. Todo el código que necesitamos para implementar eso que planeamos con las pruebas será problema un poco más adelante. Por ahora, lo que importa es asegurarnos que estamos resolviendo los problemas correctos.