**Proyecto 1 – Programación orientada a objetos**

**Marzo 2021**

Muchachos,

Este es nuestro primer proyecto de programación donde van a comenzar a aplicar los conceptos que hemos aprendido en el curso respecto: abstracción, encapsulamiento, constructores, variables de instancia, separación de responsabilidades, asociación, agregación .

Hagan el proyecto con el ánimo de aprender. Durante las dos clases de la próxima semana la idea es que avancen en el proyecto por equipos y que yo pueda resolverles dudas.

**Descripción**

La dirección de los posgrados en ingeniería de software de la Pontificia Universidad Javeriana Cali quiere hacer un sistema de información que facilite la calificación de los trabajos de grado de maestría cuando los estudiantes realizan su sustentación pública. La directora espera que el sistema entregue un archivo de texto con los resultados de la calificación obtenida por el estudiante y los comentarios relacionados con la evaluación. Esta evaluación se registra en un acta de evaluación que es diligenciada luego de la sustentación por los dos jurados participantes y está compuesta por:

* Número, fecha, autor, nombre del trabajo, tipo de trabajo ( 1. Aplicado, 2. Investigación ) periodo, director, codirector (algunas veces existe un codirector), jurado 1, jurado 2.
* Criterios de evaluación. Actualmente son 8 criterios de evaluación pero podrían extenderse en el futuro. Cada criterio tiene un identificador, un texto que es el texto que se presenta a los evaluadores y un porcentaje de ponderación. El porcentaje de ponderación se define desde el momento que se define el criterio. La sumatoria de las ponderaciones suma el 100%
* En el acta para cada criterio de evaluación se definen las observaciones de parte de los jurados, la calificación del jurado número 1 y la calificación del jurado número dos.
* A partir de las calificaciones obtenidas en cada criterio, el sistema debe calcular la nota final que corresponde al promedio de las calificaciones de cada criterio multiplicado por su porcentaje de ponderación. Con una nota superior a 3.5 el trabajo de grado es aprobado. De lo contrario el trabajo de grado es reprobado.
* El acta permite incluir observaciones adicionales y comentarios específicos sobre las condiciones para la aprobación del trabajo final.

*Al finalizar este documento puede encontrar un ejemplo del acta actual, aunque en el momento para cada criterio solo se pone una nota unificada, se espera que con el sistema cada jurado ponga su nota y el sistema pondere las calificaciones por criterios.*

La directora del posgrado espera contar con un sistema de información que:

1. Permita a la asistente de la maestría crear una nueva acta: Esta acta tiene la información de la fecha, el número del acta, nombre del estudiante, nombre del trabajo, tipo de trabajo, director, codirector (si existe), jurado 1 y jurado 2. El estado del acta cuando es creada por la asistente es “abierto”.
2. Permita al jurado 1 y jurado 2 diligenciar las calificaciones de cada criterio de evaluación para las actas ya creadas por la asistente (modificar el acta)
3. Calcule la nota final del trabajo de grado para cada acta, una vez los jurados hayan asignado una calificación para todos los criterios (modifica el acta). Según la calificación el trabajo de grado puede quedar: aprobado, pendiente, o rechazado.
4. Permita a los jurados ingresar observaciones adicionales y comentarios específicos (modificar el acta)
5. Permita “cerrar el acta”. Esto significa que no es posible hacer modificaciones en la información ya diligenciada.
6. Permita guardar la información de un acta en un archivo de texto. Sólo se podrán almacenar en archivos de texto las actas cerradas.
7. Permita lista todas las actas que se encuentran almacenadas en el sistema: Mostrará el número del acta, la fecha, el nombre del estudiante, el estado, la nota si la tiene y el estado.
8. Permita listar únicamente las actas abiertas / cerradas. Misma información del campo anterior pero solo para las actas en el estado solicitado.

Además de lo anterior, la directora de posgrados quiere saber:

1. Cuántos y cuáles trabajos de grado relacionados con industria (trabajo de tipo Aplicado) se han desarrollado
2. Cuántos y cuáles trabajos de grado de investigación se han desarrollado
3. Consultar cuántos trabajos de grado ha dirigido un profesor dado.
4. Consultar de cuales trabajos de grado ha sido jurado una persona
5. Ver la lista (sin repetidos) de todos los jurados que han participado en las actas registradas
6. Consultar las actas de los trabajos de grado pendientes y rechazados.
7. Crear información de jurados. Un jurado es interno si es un profesor de la universidad y externo si no lo es.
8. Consular la lista de jurados internos y consultar la lista de jurados externos.
9. Consultar la lista de criterios de evaluación
10. Eliminar un acta de grado ( podrán ser eliminadas únicamente las actas no cerradas)

*Puede asumir información si necesita algo que no está explícito en este documento*

**Requisitos no funcionales**

El sistema debe guardar la información en archivos binarios que se cargan al inicio de la aplicación. De esta manera aunque se cierre y vuelva a abrir el programa la asistente puede acceder a la información.

Investigue como usar enums en C++ e inclúyalos para el manejo de estad

**Bonus:**

Agregue puntos que me sorprenda. Según el nivel de sorpresa la calificación puede variar entre 0.1 y 0.5 décimas adicionales.

**Mínimos esperados (no cuentan en la nota, pero sin esto no califico el proyecto):**

En la elaboración de su programa debe considerar:

* Uso continuo de git para mantener el histórico de avance de su proyecto con comentarios claros sobre los cambios de su programa.
* Tener un menú usando do while y switch case para acceder a las diferentes opciones del programa.
* Tener un makefile para compilar el programa.
* Cumplir con el estándar de codificación lowerCamelCase, en el que las operaciones inician con un verbo en infinitivo. Las palabras compuestas inician en minúsculas y la segunda palabra tiene la primera letra en mayúsculas Ejm: *mostrarActas.* Las variables tienen nombres que semánticamente se relacionan con la función que cumplen. El cumplimiento de este estándar es obligatorio.
* Buena identación y organización del código.
* Código documentado para hacerlo claro para cualquier lector.
* Buenas prácticas de programación: buen nombramiento, no números mágicos, uso de constantes.

**Entregables**

* **Informe de autoevaluación:** documento en el que explique qué problemas tuvo al hacer su proyecto, qué aprendió, qué le gusto, que no le gustó, qué hizo cada uno de los miembros del equipo, qué nota se pondrían de manera individual y qué nota le pondrían al compañero junto con la justificación. Cada persona del equipo debe entregar un informe individual.
* **Manual técnico:** Imagen del diagrama de clases e imágenes con explicaciones de las principales funcionalidades de la aplicación funcionando (uno por equipo).
* **Código fuente del programa.**

Todos los entregables deben ser subidos al repositorio de git en una carpeta llamada ***POSGSOFT***. En Teams uno de los miembros del equipo sube la URL de repositorio donde quedaron subidos los archivos. Puede subir la información a su repositorio máximo hasta las 11:55 pm del 21 de marzo del 2021. Si prefiere dar la nota de su compañero de manera anónima puede comunicarse conmigo. ***Los documentos deben tener una calidad de redacción , ortografía y presentación esperadas a nivel universitario.***

**Calificación**

* Autoevaluación: 15 %
* Calificación del compañero: 15%. En trabajos individuales la autoevaluación tendrá un 30% de peso.
* Calificación de la profesora: 70%.
* Propiedad intelectual: valor entre 0 y 1 que multiplica la calificación total. Se demuestra durante la sustentación.
* Nota final = (calificaciónProfesora\*70% + autoevaluación\*15% + calificaciónCompañero\*15%) \* propiedadIntelectual

**Rúbrica de evaluación**

La nota del la profesora será dada al finalizar la sustentación según los criterios que se describen a continuación y su capacidad para sustentar su trabajo.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **5** | **4** | **3** | **2** | **1** | **0** |
| Entregables | Los informes están bien escritos, son claros | Fueron entregados todos los informes, se podría mejorar su calidad. | Faltan algunos de los informes, aunque los entregados son de buena calidad | Se entregaron solo algunos de los informes | Se entregaron solo algunos de los informes o los entregados tienen mala calidad | No se entregaron los informes |
| Diseño  (30%) | Excedió las expectativas | El diseño permite responder a todos los requisitos. | El diseño responde al 75% de los requisitos / podría mejorarse | El diseño responde solo al 50% de los requisitos / podría mejorarse | El diseño responde a mínimo el 25% de los requisitos | El diseño no responde con los requisitos. |
| Funcionalidad  (30%) | Excedió las expectativas | Cumplió con todos los requisitos. | Fueron desarrollados mínimo el 75% de los requisitos | Fueron desarrollados mínimo el 50% de los requisitos | Fueron desarrollados mínimo el 25% de los requisitos | Fueron desarrollados menos del 25% de los requisitos |
| Estilo de codificación  (10%) | El código se encuentra correctamente identado, los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. El código tiene documentación interna para facilitar la revisión. | La mayoría del código se encuentra correctamente identado-  La mayoría de los nombres de los atributos y las funciones cumplen con el estándar de nombramiento. La mayoría del código tiene documentación interna para facilitar la revisión | Falta una de las cosas del estilo de codificación o alguna se cumple con mala calidad | Faltan dos de las cosas del estilo de codificación o se cumplen con mala calidad: | Faltan tres de las cosas del estilo de codificación o se cumplen con mala calidad: | No cumple con el estándar de nombramiento  No se encuentra correctamente indentado  No está divido adecuadamente |
| Mejores prácticas  (15%) | El código muestra mejores prácticas de desarrollo siempre. Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. | El código muestra mejores prácticas de desarrollo en la mayoría de los casos, pero falta mejorar algunos de los siguientes aspectos Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones | El código muestra buenas prácticas de desarrollo pero falta mejorar  dos de los siguientes aspectos  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones | El código aplica pocas buenas prácticas de desarrollo. Falta mejorar tres de los siguientes aspectos:  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. | El código aplica muy pocas buenas prácticas de desarrollo.  Falta mejorar cuatro de los siguientes aspectos:  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. | El código no considera buenas prácticas de desarrollo.  Falta mejorar cinco o más de los siguientes aspectos:  Reúso, separación de operaciones, buen manejo de ciclos, simplicidad,  programación defensiva  validaciones. |

**Rúbrica de propiedad intelectual**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 0.8 | 0.6 | 0.4 | 0.2 | 0 |
| **Sustentación** | Es evidente que el estudiante entiende el código que desarrolló lo explica con claridad y responde correctamente a las preguntas. | La sustentación es buena pero se evidenció inseguridad del estudiante para explicar algunas partes del trabajo desarrollado o para responder algunas preguntas. | La sustentación es aceptable se evidencia que el estudiante desarrolló el código pero le cuesta trabajo explicar aspectos del código. | La sustentación es regular se evidenció inseguridad del estudiante para explicar gran parte del trabajo desarrollado o para responder muchas de las preguntas. Parece que el código no hubiera sido desarrollado por el estudiante. | El estudiante demuestra que entiende partes del código, pero no tiene claro cómo se relacionan con la funcionalidad solicitada. | Se evidencia que el estudiante no entiende el código desarrollado, no es capaz de responder a las preguntas formuladas de manera correcta. |