Pontificia Universidad JAVERIANA Cali | Facultad de Ingeniería y Ciencias

Proyecto 1 – Programación orientada a objetos

Septiembre 2021

Muchachos,

Este es nuestro primer proyecto de programación donde van a comenzar a aplicar los conceptos que hemos aprendido en el curso respecto: constructores, variables de instancia, separación de responsabilidades, asociación, agregación y paso de mensajes para llamar a los objetos.

Hagan el proyecto con el ánimo de aprender. **Este proyecto lo pueden hace en grupos de máximo tres personas.**

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 normalmente durante una sesión de discusión luego de la sustentación y está compuesta por:

- Número, fecha, autor, nombre del trabajo, tipo de trabajo (1. Aplicado, 2. Investigación), 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 está definido por la dirección de los posgrados. Eventualmente podría ser ajustados por la dirección de los posgrados.
- En el acta para cada criterio de evaluación se incluye la calificación del jurado número 1 y la calificación del jurado número dos y los comentarios específicos para el criterio.
- El acta permite incluir observaciones adicionales y comentarios específicos sobre las condiciones para la aprobación del trabajo final.

  Adjunto a 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 Figura 1 presenta el diagrama de casos de uso que describe las funcionalidades con las que la directora de posgrado espera contar en el sistema.

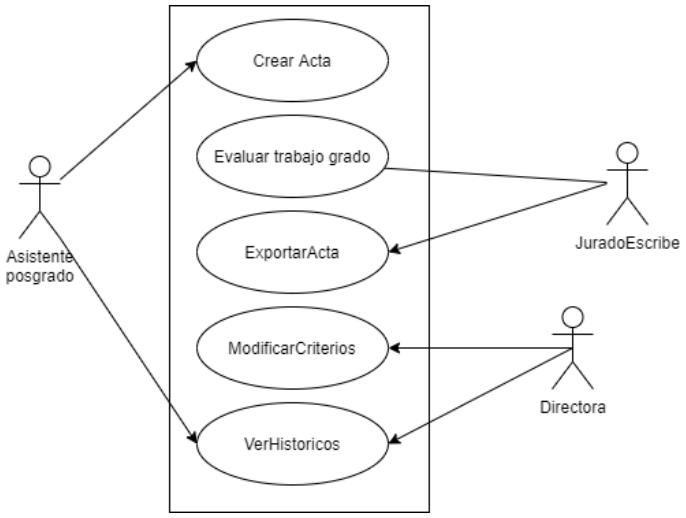


FIgura 1 Diagrama de casos de uso - posgrados

En concreto, en términos de **historias de usuario** el sistema debe cumplir lo siguiente:

1. Como **asistente** de maestría **quiero** 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 (investigación o aplicado), nombre director, nombre codirector (si existe), nombre jurado 1 y nombre jurado 2. Yo soy la encarga de crear el acta **para que** los jurados puedan posteriormente hacer la evaluación.

FIgura 2. Diagrama de flujo Acta

2. Como **jurado** del trabajo **quiero** hacer la evaluación de una tesis de maestría. La información de la evaluación quedará registrada en un acta de evaluación creada previamente en el sistema por la asistente de maestría. Nota: un mismo jurado es el encargado de ingresar al sistema la calificación dada por el jurado uno y por el jurado dos. Adicionalmente él es encargado de escribir las observaciones para cada criterio de evaluación y las observaciones generales del trabajo.
3. Como **jurado** del trabajo **quiero** saber la nota final del trabajo de grado para cada acta, una vez haya asignado una calificación para todos los criterios. Según la calificación el trabajo de grado puede quedar: aprobado o rechazado. La nota final 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.
4. Como **jurado** del trabajo **quiero** exportar la información del acta en un archivo de texto una vez haya calculado la nota final.
5. Como **directora** de los posgrados en software **quiero** modificar la descripción o el peso de los criterios de evaluación que se tienen en cuenta para elaborar las actas.
6. Como **directora** de los posgrados en software y asistente de la maestría **quiero** ver información que resuma las actas del sistema. En concreto queremos ver el número del acta, la fecha, el nombre del estudiante, el estado, la nota, nombre de los jurados y director.

A fin de mantener las estadísticas disponibles aunque el sistema se cierre y se vuelva abrir, el sistema debe mantener en un archivo de texto (en formato csv) con esta información. Este archivo se cargará al abrir el sistema.

La Figura 2 presenta el diagrama de flujo del proceso del acta desde su creación hasta la evaluación:

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

Requisitos no funcionales

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

Bonus:

Agregue puntos que me sorprenda. Según el nivel de sorpresa la calificación puede variar entre 0.1 y 0.5 décimas adicionales. Por ejemplo podrían explorar una librería para exportar las actas en PDFs 😊

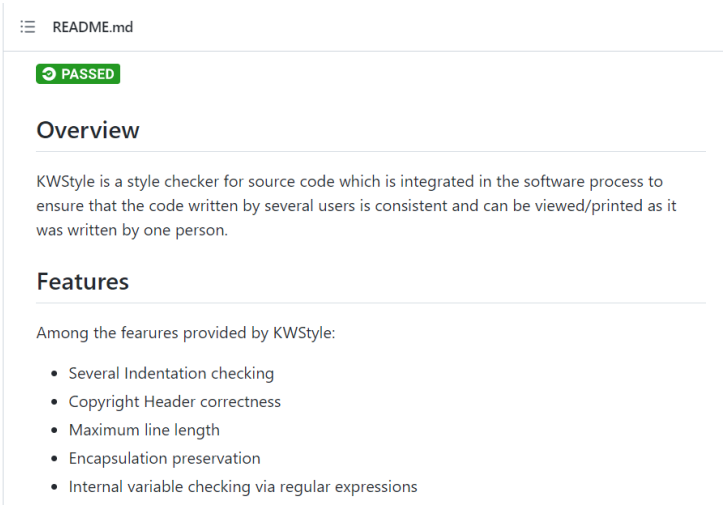
Mínimos esperados (no cuentan en la nota, pero sin esto no califico el proyecto):

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 indentación y organización del código.
- Código documentado para hacerlo claro para cualquier lector. En la documentación recuerde que sus comentarios deben explicar por qué se hacen las cosas más que solo describir literamente las líneas de código.
- Buenas prácticas de programación: buen nombramiento, no números mágicos, uso de constantes.

Entregables

- **Código fuente del programa.**
- **Informe de autoevaluación:** documento en el que explique qué problemas tuvo al hacer su proyecto, qué aprendió, qué le gustó, 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.
- **README en el repositorio [Manual técnico]:** Con presentación general del proyecto, principales funcionales, y explicación de las principales funcionalidades de la aplicación funcionando y el diagrama UML (uno por equipo). La siguiente figura muestra una imagen de ejemplo de un README disponible en https://github.com/Kitware/KWStyle. Tenga en cuenta que para documentar el README en Github debe usar MarkDown (aquí puede encontrar más info https://www.markdownguide.org).



README.md

PASSED

Overview

KWStyle is a style checker for source code which is integrated in the software process to ensure that the code written by several users is consistent and can be viewed/printed as it was written by one person.

Features

Among the fearures provided by KWStyle:

- Several Indentation checking
- Copyright Header correctness
- Maximum line length
- Encapsulation preservation
- Internal variable checking via regular expressions

Todos los entregables deben ser subidos a un repositorio de git en una carpeta llamada ***POSGSOFT+iniciales***. 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 14 de septiembre 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***. Las sustentaciones serán el 15 de septiembre de manera grupal e individual en el horario de clase.

Durante la clase del 13 de septiembre solucionaremos dudas asociadas al proyecto. También puede contactarme en otros momentos para solucionar estas dudas.

Calificación

- Autoevaluación: 10 %
- Calificación del compañero: 10%. En trabajos individuales la autoevaluación tendrá un 20% de peso.
- Calificación de la profesora compuesta por entregables, diseño, funcionalidad, estilo de codificación y mejores prácticas: 80%. La rubríca de evaluación explica los elementos que se consideran para la calificación del proyecto.
- Propiedad intelectual: valor entre 0 y 1 que multiplica la calificación total. Se demuestra durante la sustentación.
- Nota final = (criterios evaluacion%) * propiedadIntelectual

Rúbrica de evaluación

La nota de 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 (10%)	Los informes contienen toda la información solicitada y tiene alta calidad en cuanto a estilo y formato.	Se entregaron todos los informes. En términos de contenido están completos pero podría mejorar en cuanto a estilo o formato	Se entregaron todos los informes. En términos de contenido están completos pero podría mejorar en cuanto a estilo Y formato	Faltan algunos de los informes pero los entregados tiene buen estilo y formato	Faltan algunos de los informes y los entregados necesitan mejoras de estilo y formato	No se entregaron los informes
Diseño (30%)	El diseño responde a los requisitos. Se detectaron	El diseño responde a los requisitos. Se detectaron	El diseño responde a los requisitos. Faltó detectar	Faltó detectar la mayoría de clases importantes	El diseño no satisface los requisitos	No se entregó el diseño

	todas las clases importantes y para cada clase se detectaron los atributos y métodos importantes. Usa las relaciones correctas	todas las clases importantes y para cada clase se detectaron los atributos y métodos importantes. Tiene algunas relaciones incorrectas	algunas clases importantes. Faltó detectar algunos de los atributos y métodos importantes. Tiene algunas relaciones incorrectas	Faltó detectar muchos de los atributos y métodos importantes. Tiene muchas relaciones incorrectas		
Funcionalidad (30%)	Cumplió con todos los requisitos.	Fueron desarrollados mínimo el 75% de los requisitos	El diseño responde al 75% de los requisitos / podría mejorarse	Fueron desarrollados mínimo el 25% de los requisitos	Fueron desarrollados menos del 25% de los requisitos	La funcionalidad no responde a lo solicitado
Estilo de codificación (5%)	El código se encuentra correctamente indentado, 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 indentado- 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 alguno de los ítems de calidad del estilo de codificación o alguna se cumple con mala calidad	Faltan dos de los ítems de calidad del estilo de codificación o dos se cumple con mala calidad	No hay código fuente suficiente para evaluar el estilo de codificación.	No cumple con el estándar de nombramiento No se encuentra correctamente indentado No está divido adecuadamente
Mejores prácticas (5%)	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.	No hay código fuente suficiente para evaluar las buenas prácticas.	No se entregó proyecto o el proyecto no cubre al menos el 25% de las funcionalidades. No es posible evaluarlo .

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.