Curso de desarrollo de software

Practica Calificada 1

Reglas para la Prueba Calificada

- Queda terminantemente prohibido el uso de herramientas como ChatGPT, WhatsApp, o cualquier herramienta similar durante la realización de esta prueba. El uso de estas herramientas, por cualquier motivo, resultará en la anulación inmediata de la evaluación.
- Las respuestas deben presentarse con una explicación detallada, utilizando términos técnicos apropiados. La mera descripción sin el uso de terminología técnica, especialmente términos discutidos en clase, se considerará insuficiente y podrá resultar en que la respuesta sea marcada como incorrecta.
- Toda respuesta que incluya código debe ser acompañada de una explicación. La ausencia de esta explicación implicará que la pregunta se evaluará como cero.
- Se permite el uso de imágenes para explicar o complementar las respuestas, siempre y cuando estas sean de creación propia y relevantes para la respuesta, como es el caso de la verificación de los pasos de pruebas.
- Cada estudiante debe presentar su propio trabajo. Los códigos iguales o muy parecidos entre sí serán considerados como una violación a la integridad académica, implicando una copia, y serán sancionados de acuerdo con las políticas de la universidad.
- Todos los estudiantes deben subir sus repositorios de código a la plataforma del curso, según las instrucciones proporcionadas. La fecha y hora de la última actualización del repositorio serán consideradas como la hora de entrega.
- La claridad, orden, y presentación general de las evaluaciones serán tomadas en cuenta en la calificación final. Se espera un nivel de profesionalismo en la documentación y presentación del código y las respuestas escritas.
- No se otorgará tiempo adicional después de la hora de entrega establecida. Todas las entregas deben completarse dentro del plazo especificado.

Instrucciones de Entrega para la prueba calificada

Para asegurar una entrega adecuada y organizada de su prueba calificada, sigan cuidadosamente las siguientes instrucciones. El incumplimiento de estas pautas resultará en que su prueba no sea revisada, sin importar el escenario.

- Cada estudiante debe tener un repositorio personal en la plataforma designada para el curso (por ejemplo, GitHub, GitLab, etc.). El nombre del repositorio debe incluir su nombre completo o una identificación clara que permita reconocer al autor del trabajo fácilmente.
- Dentro de su repositorio personal, deben crear una carpeta titulada **PracticaCalificada1-CC3S2**. Esta carpeta albergará todas las soluciones de la prueba.
- Dentro de la carpeta **PracticaCalificada1-CC3S2**, deben crear tres subcarpetas adicionales, nombradas **Ejercicio1**, **Ejercicio2**, y **Ejercicio3**, respectivamente. Cada una de estas subcarpetas contendrá las soluciones y archivos correspondientes a cada ejercicio de la prueba.

Ejemplo de estructura:
/PracticaCalificada1-CC3S2
/Ejercicio1
/Ejercicio2
/Ejercicio3

- Todas las respuestas y soluciones deben ser documentadas, completas y presentadas en archivos Markdown (.md) dentro de las subcarpetas correspondientes a cada ejercicio. Los archivos Markdown permiten una presentación clara y estructurada del texto y el código.
- Cada archivo Markdown de respuesta debe incluir el nombre del autor como parte del nombre del archivo. Esto es para asegurar la correcta atribución y reconocimiento del trabajo individual.
- No se aceptarán entregas en formatos como documentos de Word (.docx) o archivos PDF. La entrega debe cumplir estrictamente con el formato Markdown (.md) como se especifica.
- Es crucial seguir todas las instrucciones proporcionadas para la entrega de su prueba calificada. Cualquier desviación de estas pautas resultará en que su prueba no sea considerada para revisión.
- Antes de la entrega final, verifique la estructura de su repositorio, el nombramiento de las carpetas y archivos, y asegúrese de que toda su documentación esté correctamente formateada y completa.

Pregunta 1: Aplicación de las 3A y TDD en un Proyecto Gradle (3 puntos)

Instrucciones:

- Descargue las actividades de las 3A y la actividad AAA y TDD desde la plataforma del curso.
- Inicie un proyecto Gradle en su entorno de desarrollo. Este proyecto será la base para implementar las soluciones a las actividades proporcionadas. Puedes usar lo que hayas avanzado en el curso.

Actividades de las 3A: https://github.com/kapumota/Actividades-CC3S2/blob/main/2024-1/Actividad3-AAA.md

- Revise detenidamente los requisitos y descripciones de los problemas proporcionados.
- Documente su comprensión y el análisis del problema en un archivo Markdown dentro de su proyecto Gradle.
- Responde las preguntas con detalle.
    - Diseñe los procesos que ofrezcan una solución eficiente al problema analizado.

Actividad AAA y TDD:

- Implementa la solución al problema propuesto utilizando la metodología dada en la actividad - TDD. Esto significa que debe comenzar escribiendo pruebas unitarias para los requisitos especificados antes de implementar la lógica que hace pasar estas pruebas.
- Las pruebas y la implementación deben ser añadidas al proyecto Gradle. Asegúrese de que todas las pruebas sean ejecutadas correctamente utilizando las capacidades de construcción y prueba de Gradle.

Entrega

- Asegúrese de que su proyecto Gradle incluya todas las soluciones y documentación requeridas para las actividades de las 3A y TDD.
- Siga las instrucciones de entrega especificadas previamente para nombrar y organizar su repositorio y las carpetas dentro de él.
- Suba su proyecto completo al repositorio personal en la plataforma del curso antes de la fecha y hora de cierre.

Evaluación:

- Solamente se evaluarán aquellas entregas que incluyan completas tanto las actividades de las 3A como la actividad de TDD. Cada actividad debe ser completada de acuerdo con los requisitos proporcionados.
- Se evaluará la corrección, eficiencia, y legibilidad del código; así como la adecuación de las pruebas unitarias desarrolladas según los principios de TDD
- La claridad y precisión en la documentación del análisis, algoritmo, serán criterios importantes en la evaluación.

Ejercicio 2: Haz pasar las pruebas - Uso de una clase interna ExpirableEntry (10 puntos)

- 8 puntos para pasar el código en Java
- 2 puntos para pasar el código en Kotlin

Ejercicio: ¡haz pasar las pruebas!

- Explora usando una clase interna ExpirableEntry
- Intente no utilizar clases de colección integradas; Listas, mapas o conjuntos.
- Prueba ejemplos de Java y Kotlin.

Paso 1: Entender las Pruebas

- El primer paso es entender qué es lo que las pruebas unitarias están verificando. Esto incluye identificar los métodos que se están probando, los casos de uso que cubren, y los resultados esperados para cada prueba.

Paso 2: Revisar el código existente

- Antes de realizar cualquier cambio, revisa detenidamente el código existente que acompaña a las pruebas. Comprende la lógica actual, la estructura del código, y cómo se relaciona con los requisitos expresados en las pruebas.

Paso 3: Identificar fallas y oportunidades de mejora

- Ejecuta el conjunto de pruebas unitarias para ver cuáles fallan. Esto te dará una idea clara de qué partes del código necesitan ser modificadas o corregidas.
- Basándote en los fallos de las pruebas, identifica los errores lógicos, de sintaxis, o de diseño en el código existente.

Paso 4: Modifica el código

- Modifica el código gradualmente, abordando un fallo de prueba a la vez. Esto te ayuda a aislar los problemas y asegurar que cada cambio mejora el estado del código sin introducir nuevos errores.
- Después de cada cambio, vuelve a correr las pruebas para verificar que se ha corregido el error sin afectar otras funcionalidades.

Paso 5: Refinamiento y optimización

- Una vez que todas las pruebas pasan, revisa el código modificado para oportunidades de optimización y refinamiento. Esto puede incluir mejorar la legibilidad, reducir la complejidad, o aplicar mejores prácticas de codificación.
- Cada vez que realices una mejora, asegúrate de que las pruebas sigan pasando para confirmar que no has introducido nuevos errores.
- Explica tus resultados.

No hay puntajes intermedios.

Ejercicio 3: Cucumber y expresiones regulares (7 puntos)

En este ejercicio, obtendrás algo de práctica en la creación de expresiones regulares. Construiremos las expresiones regulares en el lenguaje Gherkin. He proporcionado un archivo de cadenas de Gherkin que deben y no deben reconocerse, y debes completar el archivo Stepdefs.java para reconocer (¡o no! para los ejemplos negativos) las definiciones de Gherkin. Solo se te va a entregar los archivos necesarios para el ejercicio, forma parte de tu trabajo crear el proyecto incluyendo las dependencias adecuadas.

Si estás trabajando con Gradle y Cucumber en IntelliJ IDEA, la integración de estas herramientas te facilita ejecutar las pruebas y revisar los resultados directamente desde el IDE, incluyendo las pruebas ignoradas o aquellas que no coinciden con ninguna expresión regular en stepDefs.java de Cucumber. Estas son pruebas en las que la cadena Cucumber no coincide con ninguna de las expresiones regulares.

Como alternativa, puedes ejecutar gradle desde la línea de comandos usando gradlew (windows) o gradle (unix o mac).

Dado el archivo Test.feature proporcionado, todas las pruebas del primer escenario deberían aprobarse y todas las pruebas de los escenarios posteriores deberían "ignorarse".

Tu entregable es un archivo Stepdefs.java modificado que reconoce correctamente las expresiones regulares correctas. Lo ejecutaremos con un superconjunto de pruebas en el archivo Test.feature.
No hay puntajes intermedios. Asegúrate de que se muestren que pasen las pruebas dadas y por que colocas tus respuestas así en los archivos dados.