Índice

[Enunciado del Trabajo Práctico 3](#_Toc333011728)

[Instructivo para la ejecución de la lista de chequeo 4](#_Toc333011729)

[Lista de chequeo ejecutada 5](#_Toc333011730)

[Resultados del proceso de Inspección del Producto 7](#_Toc333011731)

# Enunciado del Trabajo Práctico

#### Monitoreo de Colmenas

“Miel Serrana” es un grupo de apicultores que se dedican a la producción de miel y otros productos que se obtienen de las abejas. Con el objeto del monitoreo de las colmenas, han solicitado un sistema web que cumpla con este propósito. Se ha solicitado a una consultora de la ciudad de Córdoba que sea la responsable del desarrollo del producto.

El proyecto mediante el cual se desarrollará el producto solicitado es denominado por sus integrantes “iBee”. La consultora se propuso para esta instancia mejorar el tiempo de entrega del producto al cliente.

El conjunto de colmenas que un apicultor tiene en un determinado lugar físico o zona se denomina apiario. Para cada apiario se debe registrar la fauna de la zona, la flora de la zona y el clima. Para el registro del clima se debe tener en cuenta temperatura, humedad, índice UV y viento. La actualización permanente es muy importante, debido a que para cada atributo que se mide del mismo se definen alarmas que deben dispararse automáticamente en la estación central de cada apiario. El sistema de alarmas debe ser cuidadosamente testeado, ya que una faya en este módulo puede hacer que el producto fracase. La consultora se propuso disminuir la densidad de defectos encontrados en el testing.

La consultora tiene definidas reglas dentro de su proceso para la escritura del código, a las que iBee debe adaptarse. Entre ellas, utilizar notación camelCase para los nombres de clases, métodos y atributos, y utilizar nombres de variables que sean representativas de lo que significan. Además, dentro de la consultora son muy exigentes con los comentarios dentro del código.

|  |  |
| --- | --- |
| **Consigna:** | Proponga un checklist para utilizar como herramienta en un proceso formal de inspección de código, luego ejecute este checklist y genere el reporte correspondiente. |
| **Objetivo:** | Que el alumno pueda generar una herramienta para ejecutar un proceso formal de inspección de código, identificando los puntos a evaluar en el proceso de inspección de código.  Qué el alumno pueda ejecutar un proceso formal de inspección de código, identificando hallazgos en una porción de código. |
| **Propósito:** | Familiarizarse con proceso formal de inspección de código. |
| **Entradas:** | Conceptos teóricos sobre el tema, desarrollados en clase. Filminas. Estándar de Inspección de Código.  Paper 11\_Best\_Practices\_For\_Peer\_Code\_Review\_Final.  Enunciado, consigna y porción de código. Estándar de Reporte de Inspección. |
| **Salida:** | Se presentará en el pizarrón la revisión de código y los hallazgos identificados. |
| **Instrucciones:** | 1. Lea atentamente el paper 11\_Best\_Practices\_For\_Peer\_Code\_Review\_Final. 2. Investigue diferentes propuestas de checklist para realizar un proceso formal de inspección de código. 3. En base al paper leído, la investigación realizada y el material teórico disponible, genere un checklist con al menos diez preguntas para utilizar en un proceso formal de inspección de código que incluya:    1. Una propuesta de encabezado para identificar unívocamente una ejecución del checklist.    2. Preguntas cerradas incluidas en el checklist.    3. Opciones para cada pregunta.    4. Opciones de resultados posibles de le ejecución del checklist (para indicar si se pasó la inspección).    5. Instructivo para completar una ejecución del checklist, incluyendo la forma en que se llega a una conclusión de resultado (ej: cuántas respuestas de una determinada opción implican que no se pasó la inspección).    6. Preguntas de control de ejecución de la inspección para realizar un control de calidad y validar que se están teniendo en cuenta buenas prácticas definidas. 4. Lea y analice la siguiente porción de código. 5. Realice una inspección del código mediante el método formal de inspección y prepara el reporte de inspección correspondiente respetando el estándar. |

# Instructivo para la ejecución de la lista de chequeo

|  |  |
| --- | --- |
| **Instrucciones para el chequeo** | |
| **1º** | Complete la cabecera con los datos iniciales. |
| **2º** | Si no está familiarizado con esta lista de chequeo, léala completamente antes de iniciar la tarea. |
| **3º** | Comience la inspección del código utilizando la lista como guía de revisión. Mientras lo hace, marque la respuesta a cada pregunta (puede marcar la opción NA si la pregunta no aplica al código revisado). |
| **4º** | Después de responder la última pregunta, complete la hora de fin en el encabezado. Con este dato calcule el tiempo que demoró en hacer la revisión y escríbalo en el casillero correspondiente. |
| **5º** | Luego cuente la cantidad de respuestas por Sí, por No y por NA; anote las cantidades en los casilleros correspondientes. |
| **6º** | Para sacar la conclusión siga los siguientes criterios: |
|  | Siempre que el porcentaje de respuestas por ***"Sí"*** sea menora a las respuestas por ***"No"***, rechace el producto inspeccionado. |
|  | Porcentaje mayor al 40% de respuestas por ***"No"***, rechace el producto inspeccionado. |
|  | Porcentaje mayor al 10% y menor al 40% de respuestas por ***"No"***, puede aceptar el producto inspeccionado con observaciones. |
|  | Porcentaje del 10% o menor de respuestas por ***"No"***, acepte el producto inspeccionado. |
| **7º** | Por último, complete los casilleros restantes en el apartado Resultados. |

# Lista de chequeo ejecutada

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lista de chequeo para revisión de código** | | | | | | | | | |
| **Proyecto:** | | | **iBee** | | | | | | |
| **Fecha Revisión:** | | | 17/08/2012 | | **Hora de inicio:** | | 23:38 | | |
| **Hora de fin:** | | 23:51 | | |
| **Revisor:** | | | Santiago Falces, Enzo Biancato, Antonio Hermes, Rodrigo Liberal | | | | | | |
| **Producto:** | | | **Nombre** | | | **Versión** | **Autor** | | |
| apiario.java | | | - | Carolina Massa | | |
|  | | |  |  | | |
| **Aspectos a Revisar** | | | | | | | | | |
|  | | **1. Estructura de codificación** | | | | | **Sí** | **No** | **NA** |
| **1.** | | ¿El código implementa correcta y completamente el diseño? | | | | |  |  | X |
| **2.** | | ¿El código corresponde al estándar de codificación aplicado? | | | | |  | X |  |
| **3.** | | ¿Está el código bien estructurado, consistente en estilo y formateado consistentemente? | | | | | X |  |  |
| **4.** | | ¿Está el código libre de fragmentos olvidados o rutinas de prueba? | | | | | X |  |  |
| **5.** | | ¿Todos los atributos tienen asignado los nombres descritos en la especificación? | | | | |  |  | X |
| **6.** | | ¿Se utilizan en el código todos los componentes externos o funciones de librerías posibles? | | | | |  |  | X |
| **7.** | | ¿Está el código libre de bloques repetidos que podrían ser integrados en un solo método? | | | | | X |  |  |
| **8.** | | ¿Está el código libre de métodos que sean muy complejos y que deban ser redefinidos en métodos más modulares? | | | | | X |  |  |
|  | | **2. Documentación** | | | | | **Sí** | **No** | **NA** |
| **9.** | | ¿Está el código clara y adecuadamente documentado con un estilo de comentario fácil de mantener? | | | | |  | X |  |
| **10.** | | ¿Son consistentes con el código los comentarios de cada clase? | | | | |  |  | X |
| **11.** | | ¿Los comentarios ayudan en la comprensión de del código? | | | | |  |  | X |
|  | | **3. Variables y Atributos** | | | | | **Sí** | **No** | **NA** |
| **12.** | | ¿Están todas las variables definidas con un nombre claro y consistente? | | | | |  | X |  |
| **13.** | | ¿Tienen todas las variables asignado su tipo correctamente? | | | | |  | X |  |
| **14.** | | ¿Está el código libre de variables redundantes? | | | | |  | X |  |
| **15.** | | ¿Todos las variables y atributos están correctamente inicializados? | | | | |  | X |  |
| **16.** | | ¿Todos los atributos tienen los modificadores de acceso adecuados (private, protected, public)? | | | | |  | X |  |
| **17.** | | ¿Todas las variables se han declarado en el ámbito correcto? | | | | | X |  |  |
|  | | **4. Operaciones Aritméticas** | | | | | **Sí** | **No** | **NA** |
| **18.** | | ¿El código previene sistemáticamente los errores de redondeo? | | | | |  |  | X |
| **19.** | | ¿El código comprueba que los divisores no sean ceros o nulos? | | | | |  |  | X |
| **20.** | | ¿Están todos los cálculos entre paréntesis para evitar la ambigüedad? | | | | |  |  | X |
|  | | **5. Bucles y Estructuras de control** | | | | | **Sí** | **No** | **NA** |
| **21.** | | ¿Están todos los bucles, ramas y estructuras lógicas completas, correctas y apropiadamente anidadas? | | | | |  | X |  |
| **22.** | | ¿Está el código libre de condicionales que carecen de funcionalidad dentro de los bucles? | | | | | X |  |  |
| **23.** | | ¿Está el código libre de condicionales “IF” anidados que pueden convertirse en una sentencia “SWITCH”? | | | | |  |  | X |
|  | | **6. Métodos** | | | | | **Sí** | **No** | **NA** |
| **24.** | | ¿Son correctos los nombres de los métodos utilizados de acuerdo con las convenciones de nombres (tales como camelCase o get y set)? | | | | |  | X |  |
| **25.** | | ¿Los métodos poseen todos los parámetros necesarios y correctos? | | | | | X |  |  |
| **26.** | | ¿Los nombres de los métodos son representativos con respecto a su funcionalidad que ejecutan? | | | | | X |  |  |
| **27.** | | ¿Todos los métodos retornan sus valores correctamente? | | | | |  | X |  |
| **28.** | | ¿Todos los métodos tienen modificadores de acceso correctos (private, protected, public)? | | | | |  | X |  |
| **29.** | | ¿Los métodos constructores de la clase están correctamente definidos? | | | | | X |  |  |
|  | | **7. Excepciones** | | | | | **Sí** | **No** | **NA** |
| **30.** | | ¿El código maneja correctamente todas las excepciones que se podrían presentar? | | | | |  | X |  |
|  | |  |  |  |  |  | **Sí** | **No** | **NA** |
|  | |  |  |  |  | **Cantidad** | 8 | 12 | 10 |
|  | |  |  |  |  | **Porcentaje** | 27% | 40% | 33% |
|  | |  |  |  |  |  |  |  |  |
| **Resultados** | | | | | | | | | |
|  |  | |  |  |  |  |  |  |  |
| **Líneas de código revisadas** | | | | 27 | **Duración de la revisión** | | 0:13 | | |
|  |  | |  |  |  |  |  |  |  |
| **Anotaciones de revisión del autor** | | | | **Buenas** | **Malas** | **No posee** |  |  |  |
|  |  | X |  |  |  |
|  |  | |  |  |  |  |  |  |  |
| **Conclusión** | | | **Aceptado** | | **Aceptado con observaciones** | | **Rechazado** | | |
|  | |  | | X | | |
| **Observaciones** | | | - | | | | | | |
|

# Resultados del proceso de Inspección del Producto

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Reporte de Inspección** | | | | | | | |
| **Fecha de Inicio:** | | | 16/08/2012 | | | | |
| **Revisores:** | | | **Nombre** | | | **Rol** | |
| Carolina Massa | | | Autor | |
| Enzo Biancato | | | Moderador | |
| Antonio Hermez | | | Anotador | |
| Santiago Falces | | | Lector | |
| Rodrigo Liberal | | | Inspector | |
| Antonio Hermez | | | Inspector | |
| Santiago Falces | | | Inspector | |
|  | | |  | |
| **Alcance de la revisión:** | | | Fragmento de la Clase Apiario – Proyecto “IBee” | | | | |
| **Productos de Trabajo Inspeccionados** | | | | | | | |
| **Nombre** | | | | **Versión** | **Autor** | | |
| apiario.java | | | | - | Carolina Massa | | |
|  | | | |  |  | | |
| **Hallazgos** | | | | | | | |
| **Nº** | **Fecha** | **Descripción** | | | | | **Severidad** |
| 1 | 08-ago | En la línea número 1 no se respeta la notación camelCase para el nombre de la clase. | | | | | Cosmético |
| 2 | 08-ago | El nombre del método "asignarC" (Línea 14) no es representativo del propósito del método. | | | | | Cosmético |
| 3 | 08-ago | En la línea 2, 3 ,4 los nombres de los atributos no son identifican a lo que representan. | | | | | Menor |
| 4 | 08-ago | En la línea 2, 3 ,4 los atributos no están inicializados con valores por defecto. | | | | | Menor |
| 5 | 08-ago | En la línea 15 el atributo “asig” no identifica a lo que representa. | | | | | Menor |
| 6 | 08-ago | En la línea 19 no se respeta la notación camelCase para el nombre del método (DESASIGNARColmena). | | | | | Menor |
| 7 | 08-ago | En la línea 20 se declara una variable que no se utiliza (bandera) y no esta inicializada. | | | | | Menor |
| 8 | 08-ago | En la línea 22 el orden de los parámetros de la estructura iterativa “for” no es correcto. | | | | | Mayor |
| 10 | 08-ago | En la línea 23 en el lado derecho de la comparación, dentro de la sentencia "if", se hace referencia a un nombre de clase y no al objeto pasado como parámetro. | | | | | Mayor |
| 11 | 08-ago | En la línea 5 el nombre del atributo (ColmenasAsignadas) no respeta la convención camelCase. | | | | | Menor |
| 12 | 08-ago | En la línea 21 el nombre de la variable (esAsignado) se contrapone a su propósito real. | | | | | Menor |
| 13 | 08-ago | En la línea 26 el tipo de retorno (Integer) es incorrecto de acuerdo a lo declarado en la cabecera del método (Boolean). | | | | | Mayor |
| 14 | 08-ago | En el método desasignarColmenta el valor de retorno nunca modifica su valor. | | | | | Mayor |
| 15 | 08-ago | En todo el código inspeccionado no se presenta ningún comentario en la clase ni en los métodos. | | | | | Mayor |
| **Decisión Final:** | | | Rechazar | | | | |