Registro de Revisión de Código

|  |
| --- |
| **Fecha de revisión:** 24 de Octubre de 2016 |
| **Producto a revisar:** Clase Máquina |
| **Autor/es:** |
| **Documentos asociados:** |
| **Revisores:** |

|  |  |  |
| --- | --- | --- |
| **Fallos en Variables, Atributos y Constantes** | **Si / No** | **Observaciones** |
| ¿Todas las variables están definidas correctamente con nombres significativos, coherentes y claros? | Si | Para realizar el UML JDeveloper crea una variable (newAtt), borrar en producto final. |
| ¿Se inicializaron antes de utilizarse? | Si |  |
| ¿Hay variables o constantes declaradas que nunca se utilizan? | Si | La variable que crea JDeveloper (newAtt). |
| ¿Las variables y constantes se declaran usando convenciones de nomenclatura? | Si | Inicio con minúscula y separación de palabra con mayúscula. |
| ¿Hay variables o atributos con nombres confusamente similares? | No | Solo en el constructor, pero se realiza así por convención. |
| ¿Pueden convertirse en locales las variables globales? | No |  |
| ¿Todas las variables de control de bucles se declaran e inicializan en el encabezamiento del bucle? | - | No aplica. |
| ¿Hay atributos que deben ser constantes? | Si | El código y la descripción de la máquina. |
| ¿Todos los atributos tienen modificadores de acceso apropiados(privado, protegido o público)? | Si |  |
| ¿Hay atributos estáticos que deberían no serlo o viceversa? | No |  |
| ¿El límite de los arrays coincide con la declaración de su tamaño? | - | No aplica. |
| ¿Hay variables asignadas dos veces sin modificarlas entre las asignaciones? | No |  |

|  |  |  |
| --- | --- | --- |
| **Fallos en Métodos** | **Si / No** | **Observaciones** |
| ¿Están comentados de forma clara y descriptiva el objetivo del método? | Si |  |
| ¿Los nombres son lo suficientemente descriptivos? | Si |  |
| ¿Se verifica que los parámetros posean valor antes de utilizarlos? | Si |  |
| ¿Requiere manejo de excepciones? | Si |  |
| ¿Los bloques de control IF THEN ELSE cubren todas las posibilidades? | Si |  |
| ¿Los bloques de flujo de control WHILE /FOR/REPEAT tienen condición o atributo que haga un corte de control y evite el ciclo infinito? | - | No aplica. |
| ¿Se necesita modularizar el método para su mejor mantenibilidad? | No |  |
| ¿Las sentencias de código están identadas de forma correcta? | Si |  |
| ¿Los métodos devuelven los valores correctos en todos sus puntos de retorno? | Si |  |
| ¿Todos los métodos tienen modificadores de acceso apropiados (privado, protegido o público)? | Si |  |
| ¿Hay métodos estáticos que deberían no serlo o viceversa? | No |  |

|  |  |  |
| --- | --- | --- |
| **Fallos en Clases** | **Si / No** | **Observaciones** |
| ¿Está definido su constructor? | Si | Tiene uno vacío para la serialización por XML. |
| ¿Los atributos están correctamente inicializados en el constructor? | Si |  |
| ¿Están declarados todos los getters y setters necesarios? | Si | Hay de más para la serialización. |
| ¿Los métodos o responsabilidades se corresponden con la definición de la Clase? | Si |  |
| ¿Alguna de las subclases tienen atributos comunes que deberían pertenecer a la superclase? | No | Las subclases no repiten atributos. |
| ¿Se puede simplificar la jerarquía de herencia de clases? | No |  |

|  |  |  |
| --- | --- | --- |
| **Fallos en cálculos numéricos** | **Si / No** | **Observaciones** |
| ¿Se realizan cálculos con tipos de datos mixtos? | - | No aplica. |
| ¿Hay desbordamiento posible durante los cálculos? | - | No aplica. |
| ¿Pueden provocarse errores de propagación por el redondeo sistemático? | - | No aplica. |
| ¿Se realizan divisiones por 0? | No |  |

|  |  |  |
| --- | --- | --- |
| **Fallos de control** | **Si / No** | **Observaciones** |
| ¿Todos los ciclos terminan? | - | No aplica. |
| ¿Existe código no alcanzable? | No |  |
| ¿Cada instrucción switch tiene un caso por defecto? | - | No aplica. |
| ¿Se utilizan adecuadamente las sentencias break en los switch? | - | No aplica. |
| ¿Se pueden modificar estructuras de if anidados por un switch? | No |  |

|  |  |  |
| --- | --- | --- |
| **Fallos de Entrada/Salida** | **Si / No** | **Observaciones** |
| ¿Se usan todas las variables de entrada? | Si |  |
| ¿Se asignan valores a las variables de salida? | - | No aplica. |
| ¿Provocan corrupción de código las entradas no esperadas? | Si | Contempladas en precondición. |
| ¿Todos los archivos son abiertos antes de su uso? | - | No aplica. |
| ¿Todos los archivos son cerrados después de su uso? | - | No aplica. |
| ¿Se manejan todas las excepciones de E/S se manera razonable? | - | No aplica. |
| ¿Se comprueba la existencia de archivos antes de intentar acceder a ellos? | - | No aplica. |

|  |  |  |
| --- | --- | --- |
| **Fallos en Interfaz** | **Si / No** | **Observaciones** |
| ¿Es correcto el número de parámetros en la invocación de las funciones? | Si |  |
| ¿Concuerdan los tipos de parámetros formales y reales? | Si |  |
| ¿El orden y valores de los parámetros invocantes son correctos? | Si |  |
| ¿Se usan los resultados de las funciones? | Si |  |
| ¿Existen funciones no invocadas? | No |  |
| ¿Concuerdan las unidades de parámetros formales y reales? (por ejemplo peso argentino o dólar) | - | No aplica. |

|  |  |  |
| --- | --- | --- |
| **Fallos de comparación/ Relacional** | **Si / No** | **Observaciones** |
| Para cada expresión booleana, ¿son correctos los operadores de comparación? | - | No aplica. |
| ¿Pueden simplificarse las expresiones booleanas por leyes de Morgan? | No |  |
| ¿Es correcta cada expresión booleana? | Si |  |
| ¿Hay comparaciones entre variables de tipos incompatibles? | No |  |
| ¿Se cubren todas las ramas de tres vías menor, igual y mayor? | - | No aplica. |

|  |  |  |
| --- | --- | --- |
| **Fallos en comentarios** | **Si / No** | **Observaciones** |
| ¿Todos los métodos, clases y archivo cabecera tienen un comentario apropiado? | No | Falta comentar la clase y los constructores, getters y setter no se consideran. |
| ¿Los comentarios ayudan a comprender el código? | Si |  |
| ¿Hay comentarios excesivos? | No |  |
| ¿Los comentarios de métodos declaran las precondiciones, postcondiciones e invariantes de su contrato? | No | Falta invariantes. |
| ¿Los comentarios son coherentes con el código que acompañan? | Si |  |

|  |  |  |
| --- | --- | --- |
| **Fallos en la modularidad** | **Si / No** | **Observaciones** |
| ¿Hay código repetitivo que podría sustituirse con la invocación a un método? | No |  |
| ¿Se utilizan las bibliotecas de Java en las clases apropiadas? | Si |  |

|  |  |  |
| --- | --- | --- |
| **Fallos en la Gestión de Almacenamiento** | **Si / No** | **Observaciones** |
| ¿Se instancian objetos fehacientemente antes de usarlos? | Si |  |
| ¿Los punteros de objetos que ya no se utilizan se asignan a null? | No | Lo hace el garbage collector. |

|  |  |  |
| --- | --- | --- |
| **Fallos de omisión** | **Si / No** | **Observaciones** |
| ¿El código implementa todo el diseño? | Si |  |
| ¿Hay requerimientos que no se implementaron? | No |  |

|  |
| --- |
| **Observaciones Generales** |
| **Sugerencias de acciones correctivas**  - Hacer constante atributos código y descripción.  - Comentario de clase.  - Poner los strings distintos de null como pre condición. |