**Test Report Template**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Making Solutions | Date | 12/04/15 |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| **Test Name/Number** | **TEST 01. Construir proyecto luego de integración** |
| **Test Objective** | Verificar construcción satisfactoria del proyecto luego de que cada miembro del equipo subiera al repositorio sus componentes |
|  |  |
| **Test Description** |  |
|  | Se ejecuta la sincronización de cambios realizados en el repositorio, para localmente construir el proyecto. |
|  |  |
|  |  |
| **Test Conditions** | Actualizar la sincronización de cambios del repositorio en el código fuente. |
|  |  |
|  |  |
| **Expected Results** | Build exitoso del proyecto: *Criterio de aceptación* |
|  | *Construcción correcta del proyecto* |
|  |  |
| **Actual Results** | Incorrect package |
|  | Al sincronizar los cambios del grupo se presenta el error de paquete, se genera error de compilación. |
|  |  |
|  |  |
|  |  |
| **Test Name/Number** | **TEST 02. Construir proyecto luego de integración** |
| **Test Objective** | Verificar que las firmas de los métodos definidos sean compatibles para todos los llamados. |
|  |  |
| **Test Description** |  |
|  | Se ejecuta la sincronización de cambios realizados en el |
|  | repositorio, para localmente construir el proyecto. |
|  |  |
|  |  |
| **Test Conditions** | Actualizar la sincronización de cambios del repositorio en el código fuente. |
|  |  |
|  |  |
| **Expected Results** | Build exitoso del proyecto: *Criterio de aceptación* |
|  | *Construcción correcta del proyecto* |
|  |  |
|  | Las firmas de los métodos: |
| **Actual Results** | RespuestaAnalisis analizar DirectorioFuente(String ruta, LenguajeEnum lenguaje) de la clase Controlador y RespuestaAnalisis analizarDirectorioFuente(LenguajeEnum lenguaje , String ruta) de la clase Analizador no concuerdan y generan error de compilación. |
|  |  |
|  |  |
| **Test Name/Number** | **TEST 03. Construir proyecto luego de integración** |
| **Test Objective** | Verificar que el objeto retornado como respuesta desde la clase Analizador, cumpla con los atributos necesarios para construir la respuesta en la clase VistaTexto. |
|  |  |
| **Test Description** | Se ejecuta la sincronización de cambios realizados en el repositorio, para localmente construir el proyecto. |
|  |  |
| **Test Conditions** | Actualizar la sincronización de cambios del repositorio en el código fuente. |
|  |  |
|  |  |
| **Expected Results** | Build exitoso del proyecto: *Criterio de aceptación* |
|  | *Construcción correcta del proyecto* |
| **Actual Results** | Incompatible types: |
|  | Required: List <Metodo> |
|  | found: Collection<Metodo> |
|  |  |
| **Test Name/Number** | **TEST 04. Pruebas de sistema** |
| **Test Objective** | Verificar que la ruta tomada como directorio fuente sea leída correctamente en todos los sistemas operativos. |
|  |  |
| **Test Description** | A continuación de la construcción del proyecto debe ser ejecutado, luego de esto se debe validar que la ruta sea tomada correctamente. |
|  |  |
| **Test Conditions** | Compilación y construcción exitosa del proyecto por el equipo |
|  |  |
|  |  |
| **Expected Results** | El directorio fuente es cargado correctamente. *Criterio de aceptación. Ubicación de directorio fuente exitoso* |
|  |  |
|  |  |
| **Actual Results** | Proyecto: TSP.Model  Número de clases: 0  Resumen de Clases |
|  | La ruta especificada no corresponde a una ruta válida para el sistema. |
|  |  |
| **Test Name/Number** | **TEST 05. Pruebas de sistema** |
| **Test Objective** | Validar que el contador de líneas implementado retorne los resultados esperados de acuerdo al estándar de código especificado, y con la integración de código realizada. |
|  |  |
| **Test Description** | Luego de efectuar la integración de código, se ejecuta la prueba de conteo. |
|  |  |
| **Test Conditions** | La ejecución del proyecto se hace posterior a la integración.  Conteo para proyecto: ciclo1ms  Ubicación de directorio fuente: resources/src |
|  | Directorio ubicado: edu\uniandes\ms\ |
| **Expected Results** | Conteo LOCS. *Criterio de aceptación. Ejecución de conteo exitoso.* |
|  | **Proyecto: ciclo1ms**  Número de clases: 14  Resumen de Clases  Nombre de la clase: Controlador.java  Cantidad líneas de la clase: 4  Resumen de métodos:  Nombre del método: analizarDirectorioFuente  Cantidad lineas del método: 3  Nombre de la clase: Main.java  Cantidad líneas de la clase: 4  Resumen de métodos:  Nombre del método: main  Cantidad lineas del método: 3  Nombre de la clase: Clase.java  Cantidad líneas de la clase: 25  Resumen de métodos:  Nombre del método: Clase  Cantidad lineas del método: 2  Nombre del método: addMetodo  Cantidad lineas del método: 2  Nombre del método: getListaMetodos  Cantidad lineas del método: 2  Nombre del método: setListaMetodos  Cantidad lineas del método: 2  Nombre del método: getNombre  Cantidad lineas del método: 2  Nombre del método: setNombre  Cantidad lineas del método: 2  Nombre del método: getCantidadLineasClase  Cantidad lineas del método: 2  Nombre del método: setCantidadLineasClase  Cantidad lineas del método: 2  Nombre del método: getLlamados  Cantidad lineas del método: 2  Nombre del método: setLlamados  Cantidad lineas del método: 2  Nombre de la clase: Metodo.java  Cantidad líneas de la clase: 22  Resumen de métodos:  Nombre del método: Metodo  Cantidad lineas del método: 1  Nombre del método: getNombre  Cantidad lineas del método: 2  Nombre del método: setNombre  Cantidad lineas del método: 2  Nombre del método: getCantidadLineasMetodo  Cantidad lineas del método: 2  Nombre del método: setCantidadLineasMetodo  Cantidad lineas del método: 2  Nombre del método: getCantidadNumeroLLamados  Cantidad lineas del método: 2  Nombre del método: setCantidadNumeroLLamados  Cantidad lineas del método: 2  Nombre del método: getCantidadPuntosDecision  Cantidad lineas del método: 2  Nombre del método: setCantidadPuntosDecision  Cantidad lineas del método: 2  Nombre de la clase: RespuestaAnalisis.java  Cantidad líneas de la clase: 13  Resumen de métodos:  Nombre del método: getNombreProyecto  Cantidad lineas del método: 2  Nombre del método: setNombreProyecto  Cantidad lineas del método: 2  Nombre del método: getRespuestaConteo  Cantidad lineas del método: 2  Nombre del método: setRespuestaConteo  Cantidad lineas del método: 2  Nombre del método: RespuestaAnalisis  Cantidad lineas del método: 2  Nombre de la clase: RespuestaConteo.java  Cantidad líneas de la clase: 15  Resumen de métodos:  Nombre del método: RespuestaConteo  Cantidad lineas del método: 2  Nombre del método: getListaClases  Cantidad lineas del método: 2  Nombre del método: setListaClases  Cantidad lineas del método: 2  Nombre del método: getLlamadoMetodos  Cantidad lineas del método: 2  Nombre del método: setLlamadoMetodos  Cantidad lineas del método: 2  Nombre del método: getNumeroClases  Cantidad lineas del método: 2  Nombre de la clase: Analizador.java  Cantidad líneas de la clase: 11  Resumen de métodos:  Nombre del método: analizarDirectorioFuente  Cantidad lineas del método: 10  Nombre de la clase: Buscador.java  Cantidad líneas de la clase: 14  Resumen de métodos:  Nombre del método: getArchivos  Cantidad lineas del método: 3  Nombre del método: getArchivosDirectorio  Cantidad lineas del método: 10  Nombre de la clase: ContadorJava.java  Cantidad líneas de la clase: 53  Resumen de métodos:  Nombre del método: ejecutarConteo  Cantidad lineas del método: 41  Nombre del método: isClassLine  Cantidad lineas del método: 2  Nombre del método: isMethodLine  Cantidad lineas del método: 2  Nombre del método: isCodeLine  Cantidad lineas del método: 2  Nombre del método: getNombreMetodo  Cantidad lineas del método: 5  Nombre de la clase: FabricaContador.java  Cantidad líneas de la clase: 11  Resumen de métodos:  Nombre del método: getInstancia  Cantidad lineas del método: 10  Nombre de la clase: IContador.java  Cantidad líneas de la clase: 2  Resumen de métodos:  Nombre del método: ejecutarConteo  Cantidad lineas del método: 1  Nombre de la clase: LenguajeEnum.java  Cantidad líneas de la clase: 1  Resumen de métodos:  Nombre de la clase: IVista.java  Cantidad líneas de la clase: 3  Resumen de métodos:  Nombre del método: analizarDirectorioFuente  Cantidad lineas del método: 1  Nombre del método: pintar  Cantidad lineas del método: 1  Nombre de la clase: VistaTexto.java  Cantidad líneas de la clase: 24  Resumen de métodos:  Nombre del método: analizarDirectorioFuente  Cantidad lineas del método: 7  Nombre del método: pintar  Cantidad lineas del método: 16 |
|  |  |
|  |  |
| **Actual Results** | Los resultados se presentan correctamente |
|  |  |
| **Test Name/Number** | **TEST 06. Pruebas de sistema** |
| **Test Objective** | Validar que el reporte presentado por consola sea claro y contenga la información solicitada en el requerimiento. |
|  |  |
| **Test Description** | Ejecutar el proyecto con un directorio fuente valido para conteo. |
|  |  |
| **Test Conditions** | La ejecución del proyecto se hace posterior a la integración.  Conteo para proyecto: ciclo1ms  Ubicación de directorio fuente: resources/src |
|  | Directorio ubicado: edu\uniandes\ms\ |
|  |  |
| **Expected Results** | Reporte de conteo con el resultado:  Clases ubicadas en el directorio fuente  Métodos identificados por clase  Cantidad de LOC's por Clase  Cantidad de LOC's por método |
| **Actual Results** | El reporte se presenta con una descripción clara y completa |

**Revisiones de código**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Jorge Beltrán | Date | 12/04/15 |
| Revisa | Angela Suárez |  |  |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes: LenguajeEnum  Controlador | | Cumple | Observación |
| Complete | Verify that the code covers all of the design. | X |  |
| Includes | Verify that the includes are complete. | X |  |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry | X | Aplica para Enum |
| Calls | Check function call formats.   * pointers * parameters * use of ‘&’ |  | No aplica |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X |  |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL | X |  |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use |  | No aplica |
| Output Format | Check the output format.   * Line stepping is proper. * Spacing is proper. |  | No aplica |
| () Pairs | Ensure that () are proper and matched. | X |  |
| Logic Operators | * Verify the proper use of ==, =, ||, and so on. * Check every logic function for (). |  | No aplica |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  | Correción de puntuación |
| Standards | Ensure that the code conforms to the coding standards. | X |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Jhon Osorio | Date | 12/04/15 |
| Revisa | Angela Suárez |  |  |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes: IVista  VistaTexto | | Cumple | Observación |
| Complete | Verify that the code covers all of the design. | X |  |
| Includes | Verify that the includes are complete. | X | Imports |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry | X | Los loop manejados son apropiadamente definidos |
| Calls | Check function call formats.   * pointers * parameters * use of ‘&’ | X |  |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X |  |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL | X |  |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use | X |  |
| Output Format | Check the output format.   * Line stepping is proper. * Spacing is proper. | X |  |
| () Pairs | Ensure that () are proper and matched. | X |  |
| Logic Operators | * Verify the proper use of ==, =, ||, and so on. * Check every logic function for (). | X |  |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  | Puntuación en Clase y Metodo |
| Standards | Ensure that the code conforms to the coding standards. | X |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Camilo Marroquín | Date | 12/04/15 |
| Revisa | Angela Suárez |  |  |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes: RespuestaAnalisis  RespuestaConteo | | Cumple | Observación |
| Complete | Verify that the code covers all of the design. | X |  |
| Includes | Verify that the includes are complete. | X | Imports |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry |  | No aplica |
| Calls | Check function call formats.   * pointers * parameters * use of ‘&’ | X |  |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X | Los atributos de la clase se definen correctamente.  Cada atributo de la clase es definido como privado y se definen correctamente los métodos de acceso. |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL |  | No aplica |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use |  | No aplica |
| Output Format | Check the output format.   * Line stepping is proper. * Spacing is proper. | X |  |
| () Pairs | Ensure that () are proper and matched. | X |  |
| Logic Operators | * Verify the proper use of ==, =, ||, and so on. * Check every logic function for (). |  | No aplica |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  | Puntuación |
| Standards | Ensure that the code conforms to the coding standards. | X |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Angela Suárez | Date | 12/04/15 |
| Revisa | Camilo Marroquín |  |  |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes: IContador  ContadorJava  FabricaContador | | Cumple | Observación |
| Complete | Verify that the code covers all of the design. | X |  |
| Includes | Verify that the includes are complete. | X | Imports |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry | X | El while utilizado para recorrer las líneas del archivo es correctamente definido. |
| Calls | Check function call formats.   * pointers * parameters * use of ‘&’ | X |  |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X | Los atributos en cada clase se identifican claramente y son coherentes de acuerdo al dato que manejan |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL | X |  |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use | X |  |
| Output Format | Check the output format.   * Line stepping is proper. * Spacing is proper. | X |  |
| () Pairs | Ensure that () are proper and matched. | X |  |
| Logic Operators | * Verify the proper use of ==, =, ||, and so on. * Check every logic function for (). | X |  |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  | Puntuación |
| Standards | Ensure that the code conforms to the coding standards. | X |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Sergio Forero | Date | 12/04/15 |
| Revisa | Angela Suarez |  |  |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes: Clase  Metodo | | Cumple | Observación |
| Complete | Verify that the code covers all of the design. | X |  |
| Includes | Verify that the includes are complete. | X | Imports |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry | X | No aplica |
| Calls | Check function call formats.   * pointers * parameters * use of ‘&’ |  | No aplica |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X | Los atributos definidos para las clases evaluadas son consistentes con el valor que contendrán |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL | X | No aplica |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use |  | No aplica |
| Output Format | Check the output format.   * Line stepping is proper. * Spacing is proper. | X |  |
| () Pairs | Ensure that () are proper and matched. | X |  |
| Logic Operators | * Verify the proper use of ==, =, ||, and so on. * Check every logic function for (). |  | No aplica |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  | Puntuación |
| Standards | Ensure that the code conforms to the coding standards. | X |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Johans Gonzalez | Date | 12/04/15 |
| Revisa | Angela Suarez |  |  |
| Program | TSP 01 | Program # | 1 |
| Instructor | LUIS BENAVIDES | Language | JAVA |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |
| --- | --- | --- | --- |
| Componentes: Analizador  Buscador | | Cumple | Observación |
| Complete | Verify that the code covers all of the design. | X |  |
| Includes | Verify that the includes are complete. | X | Imports |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry | X | La iteraciones efectuadas son de buena calidad y correctamente definidas |
| Calls | Check function call formats.   * pointers * parameters * use of ‘&’ | X |  |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X | Los atributos definidos para las clases evaluadas son consistentes con el valor que contendrán |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL | X |  |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use | X |  |
| Output Format | Check the output format.   * Line stepping is proper. * Spacing is proper. | X |  |
| () Pairs | Ensure that () are proper and matched. | X |  |
| Logic Operators | * Verify the proper use of ==, =, ||, and so on. * Check every logic function for (). | X | Las asignaciones realizadas en la clase Buscador estan correctamente definidas. |
| Line-by-line check | Check every line of code for   * instruction syntax * proper punctuation |  | Puntuación |
| Standards | Ensure that the code conforms to the coding standards. | X |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed | X |  |