|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project** | **Phase** | **Start Date and Time** | **Int. Time** | **Stop Date and Time** | **Delta Time** | **Comments** |
| Psp2 | Planning | 18/03/2015 19:00 |  | 18/03/2015 19:20 | 20 |  |
|  | Design | 18/03/2015 19:20 |  | 18/03/2015 19:39 | 19 |  |
|  | Code | 18/03/2015 19:39 |  | 18/03/2015 20:30 | 51 |  |
|  | Comp | - |  | - |  | - |
|  | Test | 18/03/2015 20:30 |  | 18/03/2015 20:40 | 10 | Defecto en test 3 |
|  | Code | 18/03/2015 20:40 |  | 18/03/2015 20:50 | 10 | Corrección |
|  | Test | 18/03/2015 20:50 |  | 18/03/2015 20:55 | 5 | Nuevo ejecución de test |

|  |
| --- |
| **Time Recording Log** |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **Language** Java |

**PSP Defect Recording Log**

|  |
| --- |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **Language** Java |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project** | **Date** | **Number** | **Type** | **Inject** | **Remove** | **Fix Time** | **Fix Ref.** |
| Psp2 | 18 de Marzo de 2015 | 1 | 80 | code |  |  |  |

Description

Salida : Valor errado en ejecución de prueba 3

PSP Process Improvement Proposal (PIP)

|  |
| --- |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **Language** Java |

|  |
| --- |
| Problem Description |
| Briefly describe the problems that you encountered. |
|  |
| Error en el ajuste del valor esperado en la prueba número 3. |
|  |
|  |
|  |
| Proposal Description |
| Briefly describe the process improvements that you propose. |
|  |
| Seguir mas de cerca la especificación de los requerimientos |
|  |
|  |
|  |
|  |
| Other Notes and Comments |
| Note any other comments or observations that describe your experiences or improvement ideas. |
|  |
| Es necesario ser mas detallista en los requerimientos, ya que con un solo texto que no se tome la |
| Implementación puede tomar un camino equivocado |
|  |

|  |  |
| --- | --- |
| Java Coding Standard | |
| Propósito | Estandarizar el esquema de codificación en los programas escritos en JAVA |
| Encabezado de los programas | El encabezado de las clases principales debe contener:  Número de programa  Nombre de autor  Fecha en la que se inicia la implementación del programa  Descripción del programa |
| Formato de encabezado | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Asignación de programa: Número de programa  /\* Nombre: Nombre de autor  /\* Fecha Fecha en la que se inicia la implementación del programa  /\* Descripción: Descripción del programa  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Lista de contenido | La lista de contenidos describe los principales aspectos que deben ser tenidos en cuenta al hacer uso de programas a implementar |
| Contenido | Instrucciones de reutilización  Instrucciones de modificación  Declaración de atributos  Declaración de métodos |
| Instrucciones de reutilización | * **Formato de declaración:** * Las declaraciones deben hacerse por línea, no se aceptan varias declaraciones en una línea. * La declaración de variables debe hacerse siempre al inicio del método. * La declaración del método debe hacerse en una línea * Luego de la palabra clave que describa un ciclo siempre debe existir un espacio. * La llave que da inicio al método debe encontrarse en el mismo renglón de la declaración del método. * La sentencia if siempre debe llevar la llave que inicie el método y la llave de cierre. * Para separar los métodos se utilizará una línea en blanco * Para separar variables locales y la primera sentencia de un método se utilizará una línea. |
| Ejemplos de Instrucciones de reutilización | * **Declaraciones lineas separadas:** * String ejemplo; * String aplicacion; * String arreglo[]; * **Declaraciones inicio método:** * public String getSize(){ * int size =0; * } * **Declaración de método:** * public String getSize(){ * **Espacio en palabras clave:** * for ( * **Ubicación llave inicio:** * public String getSize(){ * **Sentencia if:** * if (size > 0) { * **Separación método:** * public String getSize(){ * **}** * public String getDimension(){ * **}** * **Separación variable local:** * public String getSize(){ * int size=0; * System.Out.Println("size" + size); * **}** |
| Identificadores | Los identificadores deben de cierta manera describir la propiedad del atributo. |
| Ejemplo identificador | String nombreIdentificador; /\* Declaración correcta  String nI ; /\* Declaración incorrecta |
| Comentarios | * Los comentarios deben ser claros y concisos * Debe evitarse comentarios redundantes * Los comentarios no deben contar con una cantidad excesiva de caracteres |
| Buen comentario | If(nombreIdentificador.equals("correcto")) /\* Es el nombre de identificador aplicado correctamente? \*/ |
| Mal commentario | If(nombreIdentificador.equals("correcto")) /\* valida en estado del identificador |
| Secciones proncipales | Las principales secciones del programa deben estar precedidas por la línea de descripción de la sección. |
| Ejemplo | /\*\*  \* Ejecuta el conteo de líneas de una clase de acuerdo a estandar definido  \*  \* @param archivo Archivo donde ejecutará el conteo de líneas  \* @param url Ubicación del archivo  \*/ |
| Uso de mayúsculas y Minúsculas | * Todas las constantes deben definirse con maayúsculas y si cuentan con varias palabras deben separarse con guión al piso. * Los nombres de atributos siempre contendrán minúsculas y para varias palabras se usará una letra mayúscula. |
| Ejemplo de uso de mayúsculas y minúsulas | * private static final String PALABRA\_CLAVE\_CLASS = "class"; * private int cantidadDirectorios; |

Test Report Template

|  |
| --- |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **Language** Java |

|  |  |
| --- | --- |
| Test Name/Number | 1 |
|  |  |
|  |  |
| Test Description | Integración numérica utilizando la regla de simpson y distribución T |
|  |  |
|  |  |
|  |  |
|  | El usuario ingresa los valores: |
| Test Conditions | Número de segmentos = 10 |
|  | Valor para x=1,1 |
|  | Valor de dof=9 |
|  |  |
|  |  |
| Expected Results | El valor de x = 0.35006 |
|  |  |
|  |  |
|  |  |
| Actual Results | El valor de x = 0.3500589042865573 |
|  |  |

|  |  |
| --- | --- |
| Test Name/Number | 2 |
|  |  |
|  |  |
| Test Description | Integración numérica utilizando la regla de simpson y distribución T |
|  |  |
|  |  |
|  |  |
|  | El usuario ingresa los valores: |
| Test Conditions | Número de segmentos = 10 |
|  | Valor para x=1.1812 |
|  | Valor de dof=9 |
|  |  |
|  |  |
| Expected Results | El valor de x = 0.36757 |
|  |  |
|  |  |
|  |  |
| Actual Results | El valor x es : 0.3661027750821984 |
|  |  |

|  |  |
| --- | --- |
| Test Name/Number | 3 |
|  |  |
|  |  |
| Test Description | Integración numérica utilizando la regla de simpson y distribución T |
|  |  |
|  |  |
|  |  |
|  | El usuario ingresa los valores: |
| Test Conditions | Número de segmentos = 10 |
|  | Valor para x=2,750 |
|  | Valor de dof=30 |
|  |  |
|  |  |
| Expected Results | El valor de x = 0.49500 |
|  |  |
|  |  |
|  |  |
| Actual Results | El valor x es : 33.74128538330874 |
|  |  |

Size Estimating Template

|  |
| --- |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **Language** Java |

|  |  |
| --- | --- |
| Size Measure |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Estimated | | | | | | | | | | | |
| Base Parts |  | Base | |  | Deleted | |  | Modified | | |  | Added | |
| Calculo |  | 0 | |  | 0 | |  | 0 | | |  | 50 | |
| App |  | 0 | |  | 1 | |  | 1 | | |  | 10 | |
| IntegracionControlador |  | 0 | |  | 0 | |  | 3 | | |  | 0 | |
| PaintingText |  | 6 | |  | 0 | |  | 3 | | |  | 0 | |
| PaintingWeb |  | 10 | |  | 0 | |  | 2 | | |  | 0 | |
| Integracion |  | 0 | |  | 0 | |  | 0 | | |  | 20 | |
| Total | B | | 16 | D | | 1 | M | | 9 | **BA** | | | 80 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Actual | | | | | | | | | | |
| Base Parts |  | Base | |  | Deleted | |  | Modified | |  | Added | |
| Calculo |  | 0 | |  | 0 | |  | 0 | |  | 60 | |
| App |  | 0 | |  | 0 | |  | 0 | |  | 4 | |
| IntegracionControlador |  | 0 | |  | 0 | |  | 0 | |  | 0 | |
| PaintingText |  | 8 | |  | 2 | |  | 4 | |  | 5 | |
| PaintingWeb |  | 15 | |  | 4 | |  | 5 | |  | 6 | |
| Integracion |  | 0 | |  | 0 | |  | 0 | |  | 15 | |
|  |  |  | |  |  | |  |  | |  |  | |
|  |  |  | |  |  | |  |  | |  |  | |
|  |  |  | |  |  | |  |  | |  |  | |
| Total |  | | 23 |  | | 6 |  | | 9 |  | | 89 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Estimated | | | | | | | |  | Actual | | |
| Parts Additions |  | Type |  | Items |  | Rel. Size | |  | Size\* |  | Size\* |  | Items |
| App |  | Controller |  | 1 |  | L | |  | 30 |  | 25 |  | 1 |
| IntegracionControlador |  | Controller |  | 3 |  | L | |  | 45 |  | 55 |  | 3 |
| Calculo |  | Calculo |  | 6 |  | L | |  | 60 |  | 100 |  | 5 |
| Integracion |  | Calculo |  | 6 |  | M | |  | 8 |  | 30 |  | 6 |
| PaintingText |  | Presentación |  | 2 |  | S | |  | 6 |  | 4 |  | 1 |
| PaintingWeb |  | Presentación |  | 2 |  | S | |  | 30 |  | 40 |  | 2 |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
| Total |  |  |  |  |  |  | PA | | 179 |  | 254 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | Estimated | |  | Actual | |
| Reused Parts | |  | Size | |  | Size | |
| PaintingText | |  | 6 | |  | 4 | |
| PaintingWeb | |  | 30 | |  | 40 | |
|  | |  |  | |  |  | |
| Total | R | | 36 | |  | 44 | |
| PROBE Calculation Worksheet (Added and Modified) | | |  | | Size |  | | Time |
|  | | |  | |  |  | |  |
| Added size (A): A = BA+PA | | |  | | 259 |  | |  |
| Estimated Proxy Size (E): E = BA+PA+M | | |  | | 259 |  | |  |
| PROBE estimating basis used: (A, B, C, or D) | | |  | | C |  | | C |
| Correlation: (R2) | | |  | |  |  | |  |
| Regression Parameters: β0 Size and Time | | |  | | 0 |  | | 0 |
| Regression Parameters: β1 Size and Time actual total added and modified size to date/plan total added and modified size to date | | |  | | 0,12 |  | |  |
| Projected Added and Modified Size (P): P = β0size + β1size\*E | | |  | | 41 |  | |  |
| Estimated Total Size (T): T = P + B - D - M + R | | |  | | 285 |  | |  |
| Estimated Total New Reusable (NR): sum of \* items | | |  | | 20 |  | |  |
| Estimated Total Development Time: Time = β0time + β1time\*E | | |  | |  |  | |  |
| Prediction Range: Range | | |  | |  |  | |  |
| Upper Prediction Interval: UPI = P + Range | | |  | |  |  | |  |
| Lower Prediction Interval: LPI = P - Range | | |  | |  |  | |  |
| Prediction Interval Percent: | | |  | |  |  | |  |

**Design Review Checklist**

|  |
| --- |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **Language** Java |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective design 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. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Complete | Verify that the design covers all of the applicable requirements.   * All specified outputs are produced. * All needed inputs are furnished. * All required includes are stated. | X |  |  |  |
| Logic | Use a trace table, mathematical proof, or similar method to verify the logic.   * Verify that program sequencing is proper.   Stacks, lists, and so on are in the proper order.  Recursion unwinds properly.   * Verify that all loops are properly initiated, incremented, and terminated. * Examine each conditional statement and verify all cases. | X |  |  |  |
| Special Cases | * Check all special cases. * Ensure proper operation with empty, full, minimum, maximum, negative, and zero values for all variables. * Protect against out-of-limits, overflow, and underflow conditions. * Ensure “impossible” conditions are absolutely impossible. * Handle all possible incorrect or error conditions. | \_ |  |  |  |
| Functional Use | * Verify that all functions, procedures, or methods are fully understood and properly used. * Verify that all externally referenced abstractions are precisely defined. | X |  |  |  |
| Names | Verify that   * all special names are clear, defined, and authenticated * the scopes of all variables and parameters are self-evident or defined * all named items are used within their declared scopes | \_ |  |  |  |
| Standards | Ensure that the design conforms to all applicable design standards. | X |  |  |  |

**Code Review Checklist**

|  |
| --- |
| **Student** Angela Edith Suárez Torres **Date** 18 de Marzo de 2015 |
| **Program** Psp2 **Program # 5** |
| **Instructor** Luis Daniel Benavides Navarro **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. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |
| 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 | X |  |  |  |
| 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 |  |  |  |