### Program 5 requirements

|  |  |
| --- | --- |
| Program 5 requirements | Using PSP2, write a program to numerically integrate a function using Simpson’s rule. Use the t distribution as the function.  Thoroughly test the program. At a minimum, calculate the values for the t distribution integral for the values in Table 1. Expected values are also included in Table 1. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | | **Expected Value** | **Actual Value** |
| ***x*** | ***dof*** | ***P*** |  |
| 0 to x= 1.1 | 9 | 0.35006 |  |
| 0 to x= 1.1812 | 10 | 0.36757 |  |
| 0 to x= 2.750 | 30 | 0.49500 |  |

Table 1

PSP Time Recording Log

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project** | **Phase** | **Start Date and Time** | **Int. Time** | **Stop Date and Time** | **Delta**  **Time** | **Comments** |
| PSP2 | Plan | 22/02/2015 07:30 pm |  | 22/02/2015 07:45 pm | 15 | Ejecución de la fase de planeación del proyecto, se inición el día sábado en la noche con el análisis de lo que se debe realizar, durante este proceso no se presentaron interrupciones |
| PSP2 | Design | 23/02/2015 11:02 pm |  | 23/02/2015 11:14 pm | 12 | Se realize el diseño en la noche del día lunes luego de realizar la lectura del taller a realizar |
| PSP2 | Code | 23/02/2015 11:15 pm |  | 24/02/2015 11:30 pm | 15 | Se realiza el programa con base en lo diseñado y validado en el enunciado. |
| PSP2 | Code | 24/02/2015 01:05 pm |  | 24/02/2015 01:40 pm | 35 | Se continua con la fase de codificación al día siguiente, se realizan las interfaces, modelos y se inicia con la codificación de la clase calculator. |
| PSP2 | Code | 25/02/2015 02:45 am |  | 25/02/2015 03:45 am | 60 | Se realiza la codificación para el cálculo de la función gamma, y el cálculo del valor p con las sumatorias |
| PSP2 | Compile | 25/02/2015 03:45 am |  | 25/02/2015 03:45 am | 0 | Se ejecuta compilación garantizando que se ejecute sin problema las pruebas de test |
| PSP2 | Unit Test | 25/02/2015 03:45 am |  | 25/02/2015 04:20 am | 35 | Se identificaron errores en los test que fueron corregidos |
| PSP2 | PostMortem | 25/02/2015 04:20 am | 5 | 25/02/2015 05:20 am | 30 | Documentación final y empaquetamiento de solución, se present una interrupción de 5 minutos. |
|  |  |  |  |  |  |  |

Time Recording Log Instructions

|  |  |
| --- | --- |
| Purpose | * Use this form to record the time you spend on each project activity. * For the PSP, phases often have only one activity; larger projects usually have multiple activities in a single process phase. * These data are used to complete the Project Plan Summary. * Keep separate logs for each program. |
| General | * Record all of the time you spend on the project. * Record the time in minutes. * Be as accurate as possible. * If you need additional space, use another copy of the form. * If you forget to record the starting, stopping, or interruption time for an activity, promptly enter your best estimate. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Project | Enter the program name or number. |
| Phase | Enter the name of the phase for the activity you worked on, e.g. Planning, Design, Test. |
| Start Date and Time | Enter the date and time when you start working on a process activity. |
| Interruption Time | * Record any interruption time that was not spent on the process activity. * If you have several interruptions, enter their total time. * You may enter the reason for the interrupt in comments. |
| Stop Date and Time | Enter the date and time when you stop working on that process activity. |
| Delta Time | Enter the clock time you actually spent working on the process activity, less the interruption time. |
| Comments | Enter any other pertinent comments that might later remind you of any unusual circumstances regarding this activity. |

PSP Defect Recording Log

|  |  |
| --- | --- |
| Defect Types |  |
| 10 Documentation | 60 Checking |
| 20 Syntax | 70 Data |
| 30 Build, Package | 80 Function |
| 40 Assignment | 90 System |
| 50 Interface | 100 Environment |

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| PSP2 |  | | 25/02/2015 |  | 001 |  | 80 |  | 1 |  | 1 |  | 12 |  |  |
| Description: | | |  | | | | | | | | | | | | | |
| Se presento error en la codificación porque las variables no se habían asignado, estaban con los valores iniciales. | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| PSP2 |  | | 25/02/2015 |  | 002 |  | 80 |  | 1 |  | 1 |  | 5 |  |  |
| Description: | | |  | | | | | | | | | | | | | |
| Se generó un valor diferente al esperado porque el numerador había quedado como denominador en la función de calcular f(x) | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| PSP2 |  | | 25/02/2015 |  | 002 |  | 80 |  | 1 |  | 1 |  | 15 |  |  |
| Description: | | |  | | | | | | | | | | | | | |
| En el cálculo del valor del factorial se debía multiplicar por 0.5 para contemplar el valor del factorial de ½ de lo contrario los valores daban dobles a lo esperado. | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
|  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
|  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
|  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
|  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |

PSP Defect Recording Log Instructions

|  |  |
| --- | --- |
| Purpose | * Use this form to hold data on the defects that you find and correct. * These data are used to complete the Project Plan Summary form. |
| General | * Record each defect separately and completely. * If you need additional space, use another copy of the form. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Project | * Give each program a different name or number. * For example, record test program defects against the test program. |
| Date | Enter the date on which you found the defect. |
| Number | * Enter the defect number. * For each program or module, use a sequential number starting with 1 (or 001, etc.). |
| Type | * Enter the defect type from the defect type list summarized in the top left corner of the form. * Use your best judgment in selecting which type applies. |
| Inject | * Enter the phase when this defect was injected. * Use your best judgment. |
| Remove | Enter the phase during which you fixed the defect. (This will generally be the phase when you found the defect.) |
| Fix Time | * Enter the time that you took to find and fix the defect. * This time can be determined by stopwatch or by judgment. |
| Fix Ref. | * If you or someone else injected this defect while fixing another defect, record the number of the improperly fixed defect. * If you cannot identify the defect number, enter an X. |
| Description | Write a succinct description of the defect that is clear enough to later remind you about the error and help you to remember why you made it. |

PSP Process Improvement Proposal (PIP)

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |
| --- |
| Problem Description |
| Briefly describe the problems that you encountered. |
|  |
| En la ejecución de este PSP se presento el problema de generar un programa que calcule la integral por |
| Medio de la sumatoria y regla de Simpson`s, al ejecutar este PSP se siguió el proceso definido para |
| Estimación y permitió tener una major ejecución del proceso. |
|  |
|  |
|  |
| Proposal Description |
| Briefly describe the process improvements that you propose. |
|  |
| El objetivo de este psp |
| y cada summary, esto podría ayudar a guiar en el cálculo de los valores correspondientes a UPI, LPI, el |
| Range y los valores de P y B1 para el caso del PROBE C, aunque no se tiene claridad los tiempos, se |
| Puede tener una guía con unos valores estimados. |
|  |
|  |
|  |
|  |
| Other Notes and Comments |
| Note any other comments or observations that describe your experiences or improvement ideas. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

PSP Process Improvement Proposal (PIP) Instructions

|  |  |
| --- | --- |
| Purpose | * To provide a way to record process problems and improvement ideas * To provide an orderly record of your process improvement ideas * To record any other noteworthy observations |
| General | Use the PIP form to   * record process improvement ideas as they occur to you * establish priorities for your improvement plans * describe lessons learned and unusual conditions   Keep PIP forms on hand while using the PSP.   * Record process problems even without proposed solutions. * Submit a PIP with each PSP assignment report. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Problem Description | Briefly describe any problems or experiences that led to this PIP. |
| Proposal Description | Describe the proposed improvement as explicitly as possible. |
| Other Notes and Comments | Briefly describe any other observations or facts that would later help you to   * remember what you did while writing this program * remember an idea for a future improvement * explain to your instructor something you did and why you did it |

|  |  |
| --- | --- |
| Estándar de codificación | |
| Purpose | Guía la implementación o desarrollo de un programa en JAVA |
| Program Headers | Toda clase debe tener una cabecera que identifique su function y fecha de desarrollo. |
| Header Format | /\*\*  \* @Number Program: Número de programa  \* @author: Nombre de quien desarrolla la clase  \* @version: Fecha en la que se desarrolla la clase  \* @Description: Breve descripción de la clase  \*/ |
| Listing Contents | Lista de métodos de la clase |
| Contents Example | /\*\*  /\* Listing Methods  \* Method1 (Constructor)  \* Method2  \* …  \*/ |
| Identifiers | Use nombres nemotécnicos que sean descriptivos con la funcionalidad u objetivos de cada variable, método, clase, constante y demás elementos que utilice en el desarrollo del programa.  La estructura de estos nombres debe iniciar en minúscula y si el nombre es compuesto la siguiente palabra inicia en Mayúscula. |
| Identifier Example | metodoCalculador  calculaFormulas |
| Comments | Los comentarios se dan a nivel de clase y métodos, en caso de documentar una parte específica del código se realizará antes de la línea, no al finalizar la línea y se explicará su funcionalidad |
| Good Comment | /\* Condicional para validación de límite \*/  If(record\_count > limit) |
| Bad Comment | If(record\_count > limit) /\* Condicional para validación de límite \*/ |
| Major Sections | En caso de requerirse una descripción de una sección de código se comentará indicando el inicio y fin de la sección y al iniciar su respectiva descripción. |
| Example | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* Section 1: Sección del programa que realiza calculos aritméticos  \*/  Métodos…  /\*\*  \* Fin de section 1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Blank Spaces | - Separe la definición de métodos, definición de variables, definición de rutinas con al menos un espacio.  - Separe cada programa de construcción o método con al menos un espacio. |

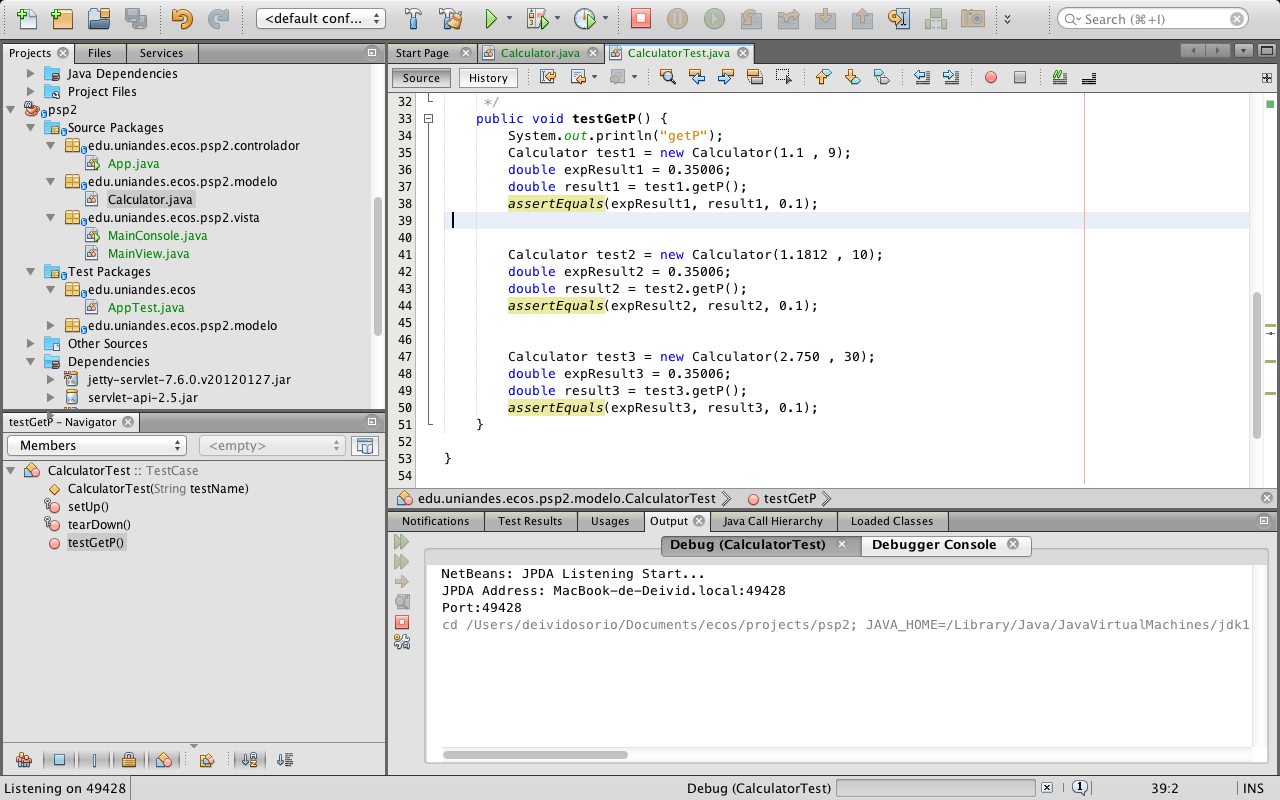
|  |  |
| --- | --- |
|  | |
| Indenting | * Idente cada línea de código según su nivel * En caso de requerir abrir un corchete siempre abralo al final de la instrucción y al cerrar cierrelo en una sola línea * Si tiene instrucciones condicionales de una única línea no require corchetes a menos que la instrucción esté dentro de un case |
| Indenting Example | while (miss\_distance > threshold){  success\_code = move\_robot (target \_location);  if (success\_code == MOVE\_FAILED) {  printf(“The robot move has failed.\n”);  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  if(miss\_distance > threshold)  printf(“The robot move has failed.\n”); |
| Capitalization | * Todas las variables o métodos se definen iniciando en minúscula y posterior a la primera palabra con letra capital * No define variables en mayuscula sostenida * Sólo para constants utilice mayusculas sostenida o para numeraciones dentro de una clase |
| Capitalization Examples | Int miVarible = 0;  Public String metodoCalculadora{}  #define DEFAULT-NUMBER-OF-STUDENTS 15  int class-size = DEFAULT-NUMBER-OF-STUDENTS; |
| Methods Declaration | - Toda definición de método debe tener su modificador definido, estos pueden ser:  Public  Private  Protected  - Los nombres de las clases se definen en letra capial |
| Methods Declaration Examples | Public Class MiPrimeraClase{} |

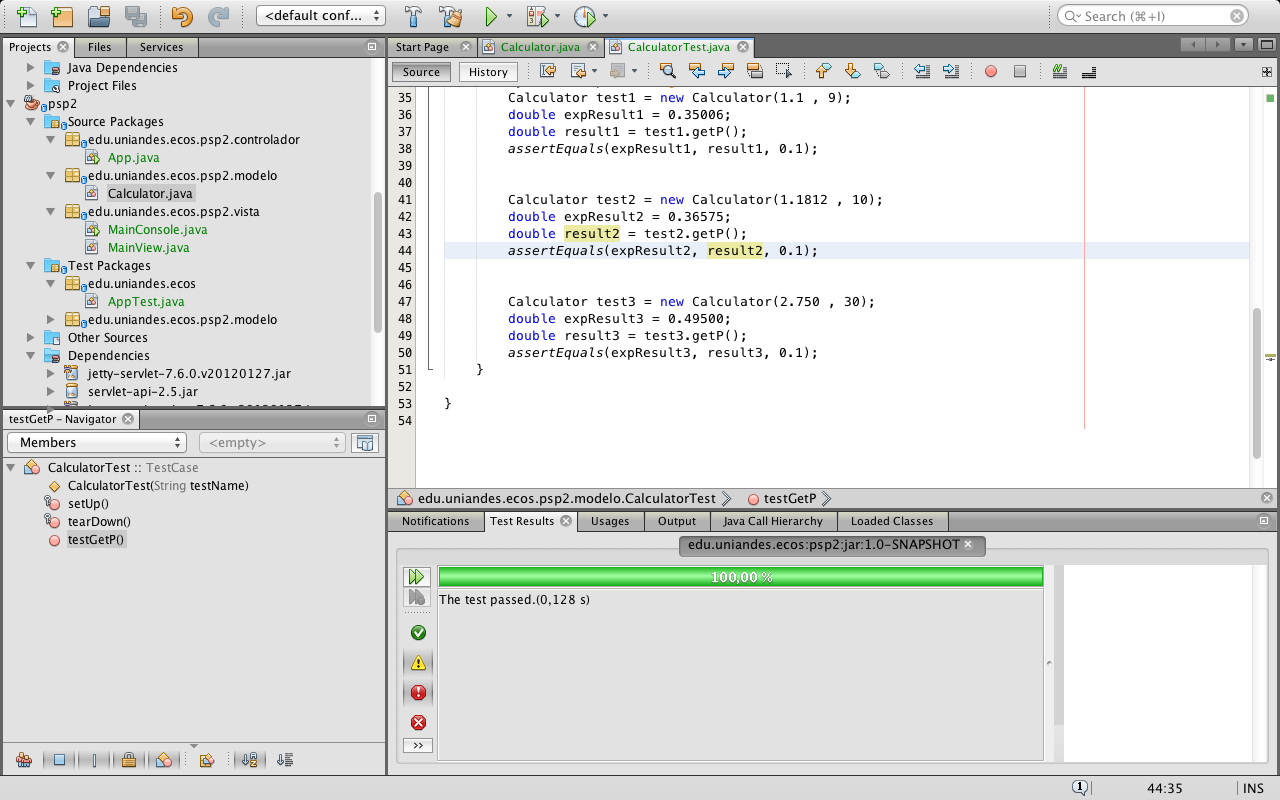
Test Report Template

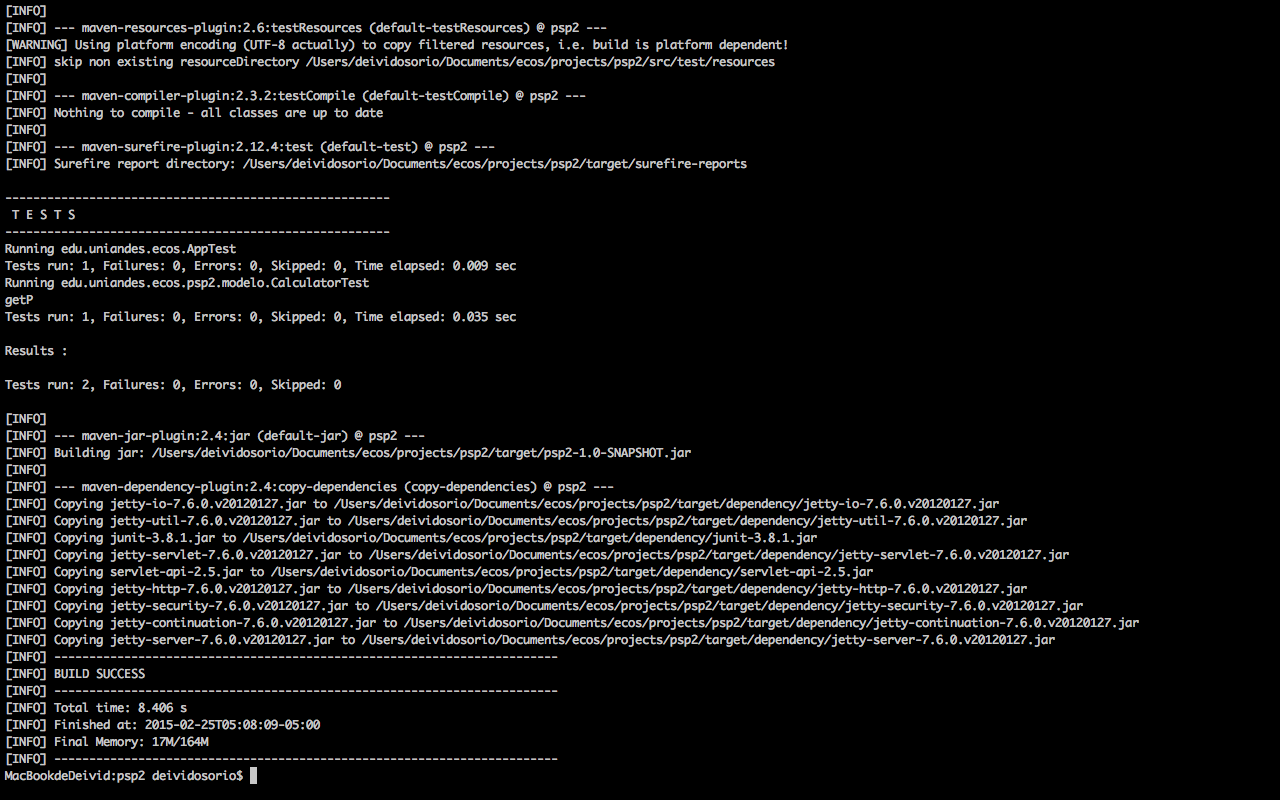
|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |  |
| --- | --- |
| Test Name/Number | Test number 1 |
| Test Objective | Validar el resultado del programa al ejecutarlo con los datos suministrados |
|  | en el enunciado del taller. |
| Test Description |  |
|  | Se dá como valor de X de 0 hasta 1.1 y dof valor 9, Esperanto como resultado |
|  | Un favor de 0.35006 |
|  |  |
|  |  |
| Test Conditions | X: P: 0.35006.35006 de 0 hasta 1.1 y dof valor 9, esperanto no se tiene claridad los tiempos, se pod{o to 1.1} |
|  | dof: 9P: 0.35006.35006 de 0 hasta 1.1 y dof valor 9, esperanto no se tiene claridad los tiempos, se pod99 |
|  |  |
|  |  |
|  |  |
| Expected Results | Se espera el resultado de P así: |
|  | P: 0.35006 |
|  |  |
|  |  |
|  |  |
|  |  |
| Actual Results | Se obtiene como resultado de P: |
|  | P: 0.350065 |
|  |  |
|  |  |
|  |  |
| Test Name/Number | Test number 2 |
| Test Objective | Validar el resultado del programa al ejecutarlo con los datos suministrados |
|  | en el enunciado del taller. |
| Test Description |  |
|  | Se dá como valor de X de 0 hasta 1.1812 y dof valor 10, Esperanto como resultado |
|  | Un favor de 0.36757 |
|  |  |
|  |  |
| Test Conditions | X: P: 0.35006.35006 de 0 hasta 1.1 y dof valor 9, esperanto no se tiene claridad los tiempos, se pod{o to 1.1812} |
|  | dof: 10P: 0.35006.35006 de 0 hasta 1.1 y dof valor 9, esperanto no se tiene claridad los tiempos, se pod99 |
|  |  |
|  |  |
|  |  |
| Expected Results | Se espera el resultado de P así: |
|  | P: 0.36757 |
|  |  |
|  |  |
|  |  |
|  |  |
| Actual Results | Se obtiene como resultado de P: |
|  | P: 0.36756 |
|  |  |
|  |  |
| Test Name/Number | Test number 3 |
| Test Objective | Validar el resultado del programa al ejecutarlo con los datos suministrados |
|  | en el enunciado del taller. |
| Test Description |  |
|  | Se dá como valor de X de 0 hasta 2.750 y dof valor 30, Esperanto como resultado |
|  | 0.49500 |
|  |  |
|  |  |
| Test Conditions | X: P: 0.35006.35006 de 0 hasta 1.1 y dof valor 9, esperanto no se tiene claridad los tiempos, se pod{o to 2.750} |
|  | dof: 30P: 0.35006.35006 de 0 hasta 1.1 y dof valor 9, esperanto no se tiene claridad los tiempos, se pod99 |
|  |  |
|  |  |
|  |  |
| Expected Results | Se espera el resultado de P así: |
|  | P: 0.49500 |
|  |  |
|  |  |
|  |  |
|  |  |
| Actual Results | Se obtiene como resultado de P: |
|  | P: 0.49500 |
|  |  |
|  |  |

Pantallas de test:







Test Report Template Instructions

|  |  |
| --- | --- |
| Purpose | * To maintain a record of the tests run and the results obtained * To be sufficiently complete so that you can later re-run the same tests and get the same results * To facilitate regression testing of modified or reused programs |
| General | * Expand this table or use multiple copies as needed. * Report all the tests that were successfully run. * Be as brief and concise as possible. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Test Name/Number | Uniquely identify each test for each program.   * the same tests with different data * the same data with different tests |
| Test Objective | Briefly describe the objective of the test. |
| Test Description | Describe each test’s data and procedures in sufficient detail to facilitate its later use as a regression test. |
| Test Conditions | * List any special configuration, timing, fix, or other conditions of the test. * When multiple tests are run with different parameters or under varying conditions, separately list each. |
| Expected Results | List the results that the test should produce if it runs properly. |
| Actual Results | List the results that were actually produced. |

Size Estimating Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |
| Size Measure |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Estimated | | | | | | | | | | | |
| Base Parts |  | Base | |  | Deleted | |  | Modified | | |  | Added | |
|  |  |  | |  |  | |  |  | | |  |  | |
| App.java |  | 65 | |  | 0 | |  | 0 | | |  | 5 | |
|  |  |  | |  |  | |  |  | | |  |  | |
|  |  |  | |  |  | |  |  | | |  |  | |
| Total | B | | 65 | D | | 0 | M | | 0 | **BA** | | | 5 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Actual | | | | | | | | | | |
| Base Parts |  | Base | |  | Deleted | |  | Modified | |  | Added | |
|  |  |  | |  |  | |  |  | |  |  | |
| App.java |  | 65 | |  | 12 | |  | 5 | |  | 0 | |
|  |  |  | |  |  | |  |  | |  |  | |
|  |  |  | |  |  | |  |  | |  |  | |
| Total |  | | 65 |  | | 12 |  | | 5 |  | | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Estimated | | | | | | | |  | Actual | | |
| Parts Additions |  | Type |  | Items |  | Rel. Size | |  | Size\* |  | Size\* |  | Items |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
| Modelo Calculator |  | Math |  | 15 |  | L | |  | 90 |  | 51 |  | 9 |
| Vista de consola MainC |  | ViewC |  | 1 |  | M | |  | 20 |  | 30 |  | 1 |
| Vista web MainView |  | ViewE |  | 2 |  | M | |  | 37 |  | 36 |  | 2 |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
| Total |  |  |  |  |  |  | PA | | 147 |  | 117 |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | Estimated |  | Actual |
| Reused Parts | |  | Size |  | Size |
|  | |  |  |  |  |
|  | |  | 0 |  | 0 |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |
|  | |  |  |  |  |
| Total | R | | 0 |  | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROBE Calculation Worksheet (Added and Modified) |  | Size |  | Time |
| Added size (A): A = BA+PA |  | 152 |  |  |
| Estimated Proxy Size (E): E = BA+PA+M |  | 152 |  |  |
| PROBE estimating basis used: (A, B, C, or D) |  | C |  | C |
| Correlation: (R2) |  | NA |  | NA |
| Regression Parameters: β0 Size and Time |  | 0 |  | 0 |
| Regression Parameters: β1 Size and Time |  | 1.16 |  | 1.50 |
| Projected Added and Modified Size (P): P = β0size + β1size\*E |  | 176.32 |  |  |
| Estimated Total Size (T): T = P + B - D - M + R |  | 224.32 |  |  |
| Estimated Total New Reusable (NR): sum of \* items |  |  |  |  |
| Estimated Total Development Time: Time = β0time + β1time\*E |  |  |  | 228 |
| Prediction Range: Range |  | 157.024 |  | 159,6 |
| Upper Prediction Interval: UPI = P + Range |  | 381.344 |  | 387.6 |
| Lower Prediction Interval: LPI = P – Range |  | 67.269 |  | 68.4 |
| Prediction Interval Percent: |  | 70% |  | 70% |

Size Estimating Template Instructions

|  |  |
| --- | --- |
| Purpose | Use this form with the PROBE method to make size estimates. |
| General | * A part could be a module, component, product, or system. * Where parts have a substructure of methods, procedures, functions, or similar elements, these lowest-level elements are called items. * Size values are assumed to be in the unit specified in size measure. * Avoid confusing base size with reuse size. * Reuse parts must be used without modification. * Use base size if additions, modifications, or deletions are planned. * If a part is estimated but not produced, enter its actual values as zero. * If a part is produced that was not estimated, enter it using zero for its planned values. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. * Enter the size measure you are using. |
| Base Parts | If this is a modification or enhancement of an existing product   * measure and enter the base size (more than one product may be entered as base) * estimate and enter the size of the deleted, modified, and added size to the base program   After development, measure and enter the actual size of the base program and any deletions, modifications, or additions. |
| Parts Additions | If you plan to add newly developed parts   * enter the part name, type, number of items (or methods), and relative size * for each part, get the size per item from the appropriate relative size table, multiply this value by the number of items, and enter in estimated size * put an asterisk next to the estimated size of any new-reusable additions   After development, measure and enter   * the actual size of each new part or new part items * the number of items for each new part |
| Reused Parts | If you plan to include reused parts, enter the   * name of each unmodified reused part * size of each unmodified reused part   After development, enter the actual size of each unmodified reused part. |

**PSP1.1 Project Plan Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Summary** | **Plan** | | |  | **Actual** | | |  | **To Date** | | |
| Size/Hour | 152/260 = 0.5846 | | |  | 165/187 = 0.8823 | | |  | 1.4646 | | |
| ***Planned Time*** | 260 | | |  |  | | |  | 580 | | |
| ***Actual Time*** |  | | |  | 187 | | |  | 527 | | |
| ***CPI (Cost-Performance Index)*** | | | |  |  | | |  | 1.100 | | |
|  |  | | |  |  | | |  | (Planned/Actual) | | |
| ***% Reused*** | 5% | | |  | 0% | | |  | 0% | | |
| ***% New Reusable*** | 5% | | |  | 0% | | |  | 0% | | |
|  |  | | |  |  | | |  |  | | |
| **Program Size** | **Plan** | | |  | **Actual** | | |  | **To Date** | | |
| Base (B) | 65 | | |  | 65 | | |  |  | | |
|  | (Measured) | | |  | (Measured) | | |  |  | | |
| Deleted (D) | 0 | | |  | 12 | | |  |  | | |
|  | (Estimated) | | |  | (Counted) | | |  |  | | |
| Modified (M) | 0 | | |  | 5 | | |  |  | | |
|  | (Estimated) | | |  | (Counted) | | |  |  | | |
| Added (A) | 5 | | |  | 0 | | |  |  | | |
|  | (A+M − M) | | |  | (T − B + D − R) | | |  |  | | |
| Reused (R) | 0 | | |  | 0 | | |  | 10 | | |
|  | (Estimated) | | |  | (Counted) | | |  |  | | |
| Added and Modified (A+M) | 10 | | |  | 117 | | |  | 139 | | |
|  | (Projected) | | |  | (A + M) | | |  |  | | |
| Total Size (T) | 147 | | |  | 165 | | |  | 334 | | |
|  | (A+M + B − M − D + R) | | |  | (Measured) | | |  |  | | |
| Total New Reusable | 0 | | |  | 0 | | |  | 0 | | |
|  |  | | |  |  | | |  |  | | |
| Estimated Proxy Size (E) | 152 | | |  |  | | |  |  | | |
|  |  | | |  |  | | |  |  | | |
| **Time in Phase (min.)** | **Plan** |  | **Actual** | | |  | **To Date** | | |  | **To Date %** |
| Planning | 10 |  | 15 | | |  | 92 | | |  | % |
| Design | 10 |  | 12 | | |  | 72 | | |  | % |
| Code | 180 |  | 100 | | |  | 1026 | | |  | % |
| Compile | 0 |  | 0 | | |  | 127 | | |  | % |
| Test | 20 |  | 35 | | |  | 235 | | |  | % |
| Postmortem | 40 |  | 25 | | |  | 275 | | |  | % |
| Total | 260 |  | 187 | | |  | 1797 | | |  | 100% |
|  |  |  |  | | |  |  | | |  |  |
| **Defects Injected** |  |  | **Actual** | | |  | **To Date** | | |  | **To Date %** |
| Planning |  |  | 0 | | |  | 0 | | |  | 0% |
| Design |  |  | 0 | | |  | 0 | | |  | 0% |
| Code |  |  | 0 | | |  | 0 | | |  | 0% |
| Compile |  |  | 3 | | |  | 14 | | |  | 100% |
| Test |  |  | 0 | | |  | 0 | | |  | 0% |
| Total Development |  |  | 0 | | |  | 14 | | |  | 100% |
|  |  |  |  | | |  |  | | |  |  |
| **Defects Removed** |  |  | **Actual** | | |  | **To Date** | | |  | **To Date %** |
| Planning |  |  | 0 | | |  | 0 | | |  | 0 |
| Design |  |  | 0 | | |  | 0 | | |  | 0 |
| Code |  |  | 0 | | |  | 0 | | |  | 0 |
| Compile |  |  | 0 | | |  | 0 | | |  | 0 |
| Test |  |  | 0 | | |  | 0 | | |  | 0 |
| Total Development |  |  | 0 | | |  | 0 | | |  | 0 |
| After Development |  |  | 0 | | |  | 0 | | |  |  |

|  |  |  |
| --- | --- | --- |
| PSP1.1 Plan Summary Instructions | |  |
| Purpose | To hold the plan and actual data for programs or program parts | |
| General | * Use the most appropriate size measure, either LOC or element count. * “To Date” is the total actual to-date values for all products developed. * A part could be a module, component, product, or system. | |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. | |
| Summary | * Enter the added and modified size per hour planned, actual, and to-date. * ***Enter the planned and actual times for this program and prior programs.*** * ***For planned time to date, use the sum of the current planned time and the planned times for the prior programs.*** * ***CPI = (To Date Planned Time)/(To Date Actual Time).*** * ***Reused % is reused size as a percentage of total program size.*** * ***New Reusable % is new reusable size as a percentage of added and modified size.*** | |
| Program Size | * Enter plan base, deleted, modified, reused, new reusable, and total size from the Size Estimating template. * Enter the plan added and modified size value (A+M) from projected added and modified size (P) on the Size Estimating template. * Calculate plan added size as A+M – M. * Enter estimated proxy size (E) from the Size Estimating template. * Enter actual base, deleted, modified, reused, total, and new reusable size from the Size Estimating template. * Calculate actual added size as T-B+D-R and actual added and modified size as A+M. * Enter to-date reused, added and modified, total, and new reusable size. | |
| Time in Phase | * Enter plan total time in phase from the estimated total development time on the Size Estimating template. * Distribute the estimated total time across the development phases according to the To Date % for the most recently developed program. * Enter the actual time by phase and the total time. * To Date: Enter the sum of the actual times for this program plus the to-date times from the most recently developed program. * To Date %: Enter the percentage of to-date time in each phase. | |
| Defects Injected | * Enter the actual defects by phase and the total actual defects. * To Date: Enter the sum of the actual defects injected by phase and the to-date values for the most recent previously developed program. * To Date %: Enter the percentage of the to-date defects injected by phase. | |
| Defects Removed | * To Date: Enter the actual defects removed by phase plus the to-date values for the most recent previously developed program. * To Date %: Enter the percentage of the to-date defects removed by phase. * After development, record any defects subsequently found during program testing, use, reuse, or modification. | |

Task Planning Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | | | Plan | | | | | | | Actual | | |
| Program/Part | Phase | Task Name | Task Hours | Cumulative Task Hours | Week Due | Week | Week Predicted | Planned Value (PV) | Cumulative  PV | Task Hours | Cumulative EV | Week |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Totals | | |  |  |  |  |  |  |  |  |  |  |

Task Planning Template Instructions

|  |  |
| --- | --- |
| Purpose | * To estimate the development time for each project task * To compute the planned value for each project task * To estimate the planned completion date for each task * To provide a basis for tracking schedule progress even when the tasks are not completed in the planned order |
| General | * Complete the Schedule Planning and Task Planning templates together. * Select tasks that have explicit completion criteria, i.e., plan completed, program compiled and defects corrected, etc. * Expand this template or use multiple pages as needed. * Include every significant task. * Use task names and numbers that support the activity and are consistent with the project work breakdown structure. * Note that most support tools will do the earned-value calculations. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Program/Part | Enter the program or part to which the task relates. |
| Phase | Enter the phase for each task. |
| Task Name | Enter task names and/or numbers in the order in which you expect to complete them. |
| Plan: Task Hours | Enter the total planned hours for each task. |
| Plan: Cumulative Task Hours | Enter the cumulative sum of the total planned task hours. |
| Plan: Week Due | If the task has a specific due date, enter the week due here. |
| Plan: Week | * On the Schedule template, find the plan cumulative schedule hours entry that equals or just exceeds each cumulative task hours entry on this form. * The week number in that row of the Schedule template is the plan week number for the task on Task template. * If several weeks on the Schedule template have the same cumulative value, enter the earliest week number. |
| Plan: Week Predicted | * On the Schedule template, find the predicted cumulative earned value entry that equals or just exceeds each cumulative planned value entry on this form. * The week number in that row of the Schedule template is the predicted week number for the task on the Task template. * If several weeks on the Schedule template have the same cumulative value, enter the earliest week number. |
| Plan: Planned Value (PV) | * Total the planned hours for all tasks. * Find the percentage each task's planned hours is of total hours. * Enter this percentage as the planned value for each task. |
| Plan: Cumulative PV | Enter the cumulative sum of the planned values. |
| Actual: Task Hours | When a task is completed, enter the hours spent on the task. |
| Actual: Cumulative Earned Value (EV) | * Each week, total the EV for all completed tasks and enter that total beside the latest completed task. * Also enter the weekly and cumulative total EV on the Schedule template. |
| Actual: Week | As a task is completed, enter the week number it was completed. |

Schedule Planning Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | Program # | 5 |
| Instructor | Luis Daniel Benavides Navarro | Language | Java |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Plan | | | Actual | | | | Predicted |
| Week  No. | Date | Schedule  Hours | Cumulative Schedule Hours | Cumulative  Planned  Value | Schedule  Hours | Cumulative  Schedule Hours | Week  Earned Value | Cumulative  Earned Value | Cumulative  Predicted Earned Value |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Schedule Planning Template Instructions

|  |  |
| --- | --- |
| Purpose | * To record the estimated and actual hours expended by calendar period * To relate the task planned value to the calendar schedule |
| General | * Expand this template or use multiple pages as needed. * Complete in conjunction with the Task Planning template. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Week No. | * From the project start, enter a week number, typically starting with 1. * For very small projects, it may be more convenient to use days instead of weeks. |
| Date | * Enter the calendar date for each week. * Pick a standard day in the week (for example, Monday). |
| Plan: Schedule Hours | * Enter the planned number of schedule hours that you expect to spend working on the project each week. * Consider non-work time such as vacations, holidays, etc. * Consider other committed activities such as e-mail, courses, meetings, and other projects. |
| Plan: Cumulative Schedule Hours | Enter the planned cumulative schedule hours through each week. |
| Plan: Cumulative Planned Value | For each week   * take the plan cumulative schedule hours from the Schedule template * on the Task template, find the task with nearest equal or lower plan cumulative task hours and note its plan cumulative value * enter this cumulative value in the Schedule template for that week * if the cumulative value for the prior week still applies, enter it again |
| Actual: Schedule Hours | At the end of each week, enter the actual schedule hours worked in that week. |
| Actual: Cumulative Schedule Hours | At the end of each week, calculate and enter the actual cumulative schedule hours for the week. |
| Actual: Week Earned Value | At the end of each week, calculate the total earned value for each task completed during the week and enter here. |
| Actual: Cumulative Earned Value | At the end of each week, calculate the cumulative earned value for the week. |
| Predicted: Cumulative Predicted Earned Value | At the end of each week, recalculate the cumulative predicted earned value for the current week through to the end of the schedule.   * Enter the current week’s actual cumulative earned value as the current week’s cumulative predicted earned value. * Calculate the average actual earned value per hour worked on the job to date (Actual Cumulative EV/Actual Cumulative Schedule Hours). * For each week n, starting with the next week, multiply the average earned value per planned hour by the planned hours for week n. Add the result to the cumulative predicted earned value for the preceding week and enter in the cumulative predicted earned value for that week. Repeat for each week until the cumulative predicted earned value reaches 100. |

#### Design Review Checklist Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | 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. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tipo | Ítems | Calculator | MainView | MainConsole | App |
| Estàndar | * Cada paquete debe cumplir con el estándar de nombres * Cada clase debe cumplir con el estándar de codificación | X | X | X | X |
| Completitud | * El programa debe cumplir con el alcance definido * Las salidas del programa deben contener todo lo requerido por el programa * Las entradas del programa son las definidas por el alcance del requerimiento | X | X | X | X |
| Lógica | * El programa debe cumplir con un patron de diseño definido * Las clases deben heredar de una manera coherente y lógica | X | X | X | X |
| Integración | * Las entidades deben ser únicas | X | X | X | X |
| Seguridad e integridad | * La visibilidad de todas las clases debe ser explícita * Todo try debe tener su catch respective * La visibilidad de todos los métodos debe ser explícita | X | X | X | X |
| Especificación de clases | * Comentarios y documentación en cada clase y método * Los nombres de variables, métodos y clases deben ser nemotécnicos y claros. | X | X | X | X |
| Pruebas unitarias | * Toda clase de lógica debe tener su respectiva clase de test | X | X | X | X |

**Code Review Checklist Template**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Deivid Alexander Osorio Barrera | Date | 22/02/2015 |
| Program | Integración numérica con regla de Simpson | 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. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tipo | Ìtems | Calculator | MainView | MainConsole | App |
| Inicialización | * todas las variables de métodos deben estar inicializadas * todas las variables de ciclos deben estar definidas e inicializadas | X | X | X | X |
| Control de excepciones | * Todos los errores deben ser capturados con try y cacth * No deben haber exceptions genericas | - | X | X | X |
| Ciclos | * Todos los ciclos deben tener un valor de retorno en variable y no definido * Cada ciclo debe tener un indice de increment y una condición de finalización | X | X | X | X |
| Sintaxis lógica | * Las igualdades deben ser definidas con valores == y no únicamente = * Las comparaciones se definen con && y || | X | X | X | X |
| Identación | El código fuente debe cumplir las normas de identación definidas en el estándar de codificación | X | X | X | X |
| Variables | Las variables que se definen en una clase deben utilizar this y super | X | X | X | X |
| Entrada / Salida | Todas las conexiones deben estar cerradas  No se deben abrir conexiones dentro de un ciclo | X | X | X | X |
| Integridad | No se debe modificar el dato de entrada de cada método si no la copia dentro de cada método. | X | X | X | X |