PSP2.1 Project Plan Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03/2015 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | java |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Summary** | | **Plan** | | |  | **Actual** | | |  | **To Date** | | |
| Size/Hour | | 90 | | |  | 70 | | |  | 232 | | |
| Planned Time | |  | | |  |  | | |  |  | | |
| Actual Time | |  | | |  |  | | |  |  | | |
| CPI (Cost-Performance Index) | |  | | |  |  | | |  |  | | |
|  | |  | | |  |  | | |  | (Planned/Actual) | | |
| % Reuse | |  | | |  |  | | |  |  | | |
| % New Reusable | |  | | |  |  | | |  |  | | |
| Test Defects/KLOC or equivalent | |  | | |  |  | | |  |  | | |
| Total Defects/KLOC or equivalent | |  | | |  |  | | |  |  | | |
| Yield % | |  | | |  |  | | |  |  | | |
| ***% Appraisal COQ*** | |  | | |  |  | | |  |  | | |
| ***% Failure COQ*** | |  | | |  |  | | |  |  | | |
| ***COQ A/F Ratio*** | |  | | |  |  | | |  |  | | |
| ***PQI*** | |  | | |  |  | | |  |  | | |
|  | |  | | |  |  | | |  |  | | |
| **Program Size** | | **Plan** | | |  | **Actual** | | |  | **To Date** | | |
| Base (B) | | 0 | | |  | 0 | | |  |  | | |
|  | | (Measured) | | |  | (Measured) | | |  |  | | |
| Deleted (D) | | 0 | | |  | 0 | | |  |  | | |
|  | | (Estimated) | | |  | (Counted) | | |  |  | | |
| Modified (M) | | 0 | | |  | 0 | | |  |  | | |
|  | | (Estimated) | | |  | (Counted) | | |  |  | | |
| Added (A) | 20 | | | |  | 15 | | |  |  | | |
|  | (A+M − M) | | | |  | (T − B + D − R) | | |  |  | | |
| Reused (R) | 0 | | | |  | 0 | | |  |  | | |
|  | (Estimated) | | | |  | (Counted) | | |  |  | | |
| Added and Modified (A+M) | 20 | | | |  | 15 | | |  | 312 | | |
|  | (Projected) | | | |  | (A + M) | | |  |  | | |
| Total Size (T) | 20 | | | |  | 15 | | |  | 212 | | |
|  | (A+M + B − M − D + R) | | | |  | (Measured) | | |  |  | | |
| Total New Reusable |  | | | |  |  | | |  |  | | |
|  |  | | | |  |  | | |  |  | | |
| Estimated Proxy Size (E) |  | | | |  |  | | |  |  | | |
|  | |  | | |  |  | | |  |  | | |
| ***Upper Prediction Interval (70%)*** | |  | | |  |  | | |  |  | | |
| ***Lower Prediction Interval (70%)*** | |  | | |  |  | | |  |  | | |
|  | |  |  |  | | |  |  | | |  |  |

**(continued)**

**PSP2.1 Project Plan Summary (continued)**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Program # | 6 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Time in Phase (min.)** | **Plan** |  | | **Actual** | | | | |  | | **To Date** | | |  | | **To Date %** | |
| Planning | 15 |  | | 20 | | | | |  | | 97 | | |  | | 8 | |
| Design | 30 |  | | 49 | | | | |  | | 188 | | |  | | 14 | |
| Design Review | 20 |  | | 30 | | | | |  | | 60 | | |  | | 3 | |
| Code | 90 |  | | 70 | | | | |  | | 465 | | |  | | 41 | |
| Code Review | 35 |  | | 19 | | | | |  | | 39 | | |  | | 2 | |
| Compile | 0 |  | | 0 | | | | |  | | 0 | | |  | | 0 | |
| Test | 10 |  | | 15 | | | | |  | | 98 | | |  | | 9 | |
| Postmortem | 80 |  | | 65 | | | | |  | | 275 | | |  | | 22 | |
| Total | 280 |  | | 268 | | | | |  | | 1222 | | |  | | 100 | |
| ***Total Time UPI (70%)*** |  |  | |  | | | | |  | |  | | |  | |  | |
| ***Total Time LPI (70%)*** |  |  | |  | | | | |  | |  | | |  | |  | |
|  |  | |  | |  | | |  | |  | | | | |  | |  |
| **Defects Injected** | **Plan** | |  | | **Actual** | | |  | | **To Date** | | | | |  | | **To Date %** |
| Planning | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Design | 1 | |  | | 1 | | |  | | 2 | | | | |  | | 10 |
| Design Review | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Code | 2 | |  | | 1 | | |  | | 10 | | | | |  | | 90 |
| Code Review | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Compile | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Test | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Total Development | 3 | |  | | 2 | | |  | | 12 | | | | |  | | 100 |
|  |  | |  | |  | | |  | |  | | | | |  | |  |
| **Defects Removed** | **Plan** | |  | | **Actual** | | |  | | **To Date** | | | | |  | | **To Date %** |
| Planning | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Design | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Design Review | 1 | |  | | 1 | | |  | | 1 | | | | |  | | 9 |
| Code | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Code Review | 2 | |  | | 1 | | |  | | 1 | | | | |  | | 9 |
| Compile | 0 | |  | | 0 | | |  | | 0 | | | | |  | | 0 |
| Test | 0 | |  | | 1 | | |  | | 10 | | | | |  | | 91 |
| Total Development | 3 | |  | | 3 | | |  | | 12 | | | | |  | | 100 |
| After Development |  | |  | |  | | |  | |  | | | | |  | |  |
|  |  | | | | |  |  | | | | |  |  | | | | |
| **Defect Removal Efficiency** | **Plan** | | | | |  | **Actual** | | | | |  | **To Date** | | | | |
| Defects/Hour − Design Review | 4 | | | | |  | 1 | | | | |  | 1 | | | | |
| Defects/Hour − Code Review | 5 | | | | |  | 1 | | | | |  | 1 | | | | |
| Defects/Hour − Compile | 0 | | | | |  | 0 | | | | |  | 0 | | | | |
| Defects/Hour − Test | 5 | | | | |  | 1 | | | | |  | 1 | | | | |
| DRL (DLDR/UT) |  | | | | |  |  | | | | |  |  | | | | |
| DRL (Code Review/UT) |  | | | | |  |  | | | | |  |  | | | | |
| DRL (Compile/UT) |  | | | | |  |  | | | | |  |  | | | | |

PSP Time Recording Log

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03/2015 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | Java |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project** | **Phase** | **Start Date and Time** | **Int. Time** | **Stop Date and Time** | **Delta**  **Time** | **Comments** |
| 5 | Planing | 08:45 | 09:05 | 0 | 20 |  |
| 5 | Design | 09:11 | 10:00 | 0 | 49 |  |
| 5 | Review Design | 10:10 | 10:40 | 10 | 30 |  |
| 5 | Code | 10:45 | 11:40 | 15 | 70 |  |
| 5 | Review Code | 11:41 | 12:00 | 0 | 19 |  |
| 5 | Test | 12:30 | 12:45 | 0 | 15 |  |
| 5 | PostMortem | 13:10 | 14:00 | 15 | 65 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

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 | Wilman Rincon | Date | 07/03/2015 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | Java |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 6 |  | | 07/3 |  | 1 |  | 80 |  | Code |  | Review Code |  | 1 |  |  |
| Description: | | | En la clase APP no estaba llamando al método que calcula X | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 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: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 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: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |

Test Report Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | Java |

|  |  |
| --- | --- |
| Test Name/Number | Test 1/1 |
| Test Objective | Ingresar un valor de P y un valor para el Dof y encontrar el varlor X |
|  |  |
| Test Description | Encontrar el valor esperado de X |
|  |  |
|  |  |
|  |  |
|  |  |
| Test Conditions | Valor de P= 0.20 valor de dof = 6 |
|  |  |
|  |  |
|  |  |
|  |  |
| Expected Results | Valor x = 0.55338 |
|  |  |
|  |  |
|  |  |
| Actual Results | X= 1.21, el valor no coincide. |
|  |  |
|  |  |
|  |  |
|  |  |
| Test Name/Number |  |
| Test Objective |  |
|  |  |
| Test Description |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Test Conditions |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Expected Results |  |
|  |  |
|  |  |
|  |  |
| Actual Results |  |
|  |  |
|  |  |
|  |  |
|  |  |

Size Estimating Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | Java |
| Size Measure |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Estimated | | | | | | | | | | | |
| Base Parts |  | Base | |  | Deleted | |  | Modified | | |  | Added | |
|  |  |  | |  |  | |  |  | | |  |  | |
|  |  |  | |  |  | |  |  | | |  |  | |
|  |  |  | |  |  | |  |  | | |  |  | |
|  |  |  | |  |  | |  |  | | |  |  | |
| Total | B | |  | D | |  | M | |  | **BA** | | |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Actual | | | | | | | | | | |
| Base Parts |  | Base | |  | Deleted | |  | Modified | |  | Added | |
|  |  |  | |  |  | |  |  | |  |  | |
|  |  |  | |  |  | |  |  | |  |  | |
|  |  |  | |  |  | |  |  | |  |  | |
|  |  |  | |  |  | |  |  | |  |  | |
| Total |  | |  |  | |  |  | |  |  | |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Estimated | | | | | | | |  | Actual | | |
| Parts Additions |  | Type |  | Items |  | Rel. Size | |  | Size\* |  | Size\* |  | Items |
| CalcularSimpson |  | Calc |  | 4 |  | Medio | |  | 40 |  | 30 |  | 3 |
| App |  | Logic |  | 2 |  | Small | |  | 10 |  | 20 |  | 4 |
| MainView |  | Logic |  | 2 |  | Small | |  | 10 |  | 15 |  | 3 |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  |
| Total |  |  |  |  |  |  | PA | | 60 |  | 65 |  |  |

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

Size Estimating Template (continued)

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Program | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROBE Calculation Worksheet (Added and Modified) |  | Size |  | Time |
| Added size (A): A = BA+PA |  |  |  |  |
| Estimated Proxy Size (E): E = BA+PA+M |  | 25 |  |  |
| PROBE estimating basis used: (A, B, C, or D) |  | C |  |  |
| Correlation: (R2) |  |  |  |  |
| Regression Parameters: β0 Size and Time |  | 30.12 |  |  |
| Regression Parameters: β1 Size and Time |  | 0.94 |  |  |
| Projected Added and Modified Size (P): P = β0size + β1size\*E |  | 57.38 |  |  |
| Estimated Total Size (T): T = P + B - D - M + R |  | 547.38 |  |  |
| Estimated Total New Reusable (NR): sum of \* items |  |  |  |  |
| 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: |  |  |  |  |

PSP Process Improvement Proposal (PIP)

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03/2015 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | Java |

|  |
| --- |
| Problem Description |
| Briefly describe the problems that you encountered. |
| Para este ejercio fue más fácil la implementación del código |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| Proposal Description |
| Briefly describe the process improvements that you propose. |
|  |
| Con la experiencia ganada imprlementando el patrón de software MVC, ya se hace más fácil la realización |
| Del código. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| Other Notes and Comments |
| Note any other comments or observations that describe your experiences or improvement ideas. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

**Coding/Counting Standard Template**

|  |  |
| --- | --- |
| Purpose | To guide the development of programs |
| Counting Standard | * Count each physical line as one LOC. * Do not count blank lines and comment-only lines. * Be consistent about what you put on each physical line. * Do not count import nor package lines (at the beginning) * Do not count special flag-marked lines. We are definifing the below ones:   + //M for modified lines   + //D for deleted lines |
| Program Headers | Begin all programs with a descriptive header. |
| Header Format | /\*  \* Classname  \*  \* Version information  \*  \* Date  \*  \*/ |
| Reuse Instructions | * Describe how the program is used. Provide the declaration format, parameter values and types, and parameter limits. * Provide warnings of illegal values, overflow conditions, or other conditions that could potentially result in improper operation. |
| Reuse Example | /\*\*   \* Brief description of the method purpose an operation  \* @author Engineer’s name  \* @param p\_fechaDesde Fecha desde la cual se deben buscar datos  \* @return valor del resultado  \*/ |
| Identifiers | Use descriptive names for all variables, method names, constants, and other identifiers. Avoid single letter variables, use them only when they are short lived. |
| Identifier Naming Table | |  |  |  | | --- | --- | --- | | Classes | Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations | class Raster;  class ImageSprite; | | Interfaces | Interface names should be capitalized like class names. | interface RasterDelegate;  interface Storing; | | Methods | Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized. | run();  runFast();  getBackground(); | | Variables | Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore \_ or dollar sign $ characters, even though both are allowed.  Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters. | int i;  char c;  float myWidth; | | Constants | Names should be in uppercase. | static final int MIN\_WIDTH = 4;  static final int MAX\_WIDTH = 999;  static final int GET\_THE\_CPU = 1; | |

(continued)**Coding Standard Template (continued)**

|  |  |
| --- | --- |
| Comments | * Document the code so that the reader can understand its operation. * Comments should explain both the purpose and behavior of the code. * Comment variable declarations to indicate their purpose. |
| Comments types | A block comment should be preceded by a blank line to set it apart from the rest of the code.  /\*  \* Here is a block comment.  \*/  A single-line comment should be preceded by a blank line.  if (condition) {  /\* Handle the condition. \*/  ...  }  Very short comments can appear on the same line as the code they describe, but should be shifted far enough to separate them from the statements. If more than one short comment appears in a chunk of code, they should all be indented to the same tab setting.  Here's an example of a trailing comment in Java code:  if (a == 2) {  return TRUE; /\* special case \*/  } else {  return isPrime(a); /\* works only for odd a \*/  }  The // comment delimiter can comment out a complete line or only a partial line. It shouldn't be used on consecutive multiple lines for text comments  if (foo > 1) {  // Do a double-flip.  ...  }  else {  return false; // Explain why here.  }  //if (bar > 1) {  //  // // Do a triple-flip.  // ...  //}  //else {  // return false;  //} |
| Good Comment | If(record\_count > limit) /\* have all records been processed?  \*/ |
| Bad Comment | If(record\_count > limit) /\* check if record count exceeds limit  \*/ |
| Major Sections | Precede major program sections by a block comment that describes the processing that is done in the next section |
| Example | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  /  /\* The program section examines the contents of the array ‘grades’ and calcu- \*/  /\* lates the average class grade.  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  / |
| Blank Spaces | * Write programs with sufficient spacing so they do not appear crowded. * Separate every program construct with at least one space. |
| Indenting | * Avoid lines longer than 80 characters, since they're not handled well by many terminals and tools. * Indent every level of brace from the previous one. |
| Indenting  Example | someMethod(longExpression1, longExpression2, longExpression3,  longExpression4, longExpression5);  if ((condition1 && condition2)  || (condition3 && condition4)  ||!(condition5 && condition6)) {  doSomethingAboutIt();  } |

**Design Review Checklist**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03/2015 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | 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. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | YES | NO | N/A |
| Complete | Verify that the design covers all of the applicable requirements.   * All specified outputs are produced. | X |  |  |
| 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. |  |  | X |
| 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 |  |  |
| System Considerations | * Verify that the program does not cause system limits to be exceeded. * Verify that all safety conditions conform to the safety specifications. | 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 | X |  |  |

**Code Review Checklist**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Wilman Rincon | Date | 07/03/2015 |
| Program | 6 | Program # | 6 |
| Instructor | Luis Daniel Benavides | Language | JAVA |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | YES | NO | N/A |
| Complete | Verify that the code covers all of the design. | X |  |  |
| Initialization | Check variable and parameter initialization.   * at program initiation * at start of every loop * at class/function/procedure entry | X |  |  |
| Names | Check name spelling and use.   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? | X |  |  |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new |  |  | 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 |  |  |
| 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 |
| Class Definition | - Does each class have an appropriate constructor?  - Can the class inheritance hierarchy be simplified? | X |  |  |
| Computation/Numeric | - Are there any computations with mixed data types?  - Are parentheses used to avoid ambiguity?  - Are divisors tested for zero or noise? | X |  |  |
| Control Flow | - Will all loops terminate?  - Does each switch statement have a default case?  - Does every method terminate?  - Are all exceptions handled appropriately? | X |  |  |
| Comment | - Does every method, class, and file have an appropriate header comment?  - Does every attribute,variable or constant declaration have a comment? | X |  |  |
| Performance | - Can better data structures or more efficient algorithms be used?  - Are timeouts or error traps used for external device accesses? |  |  | X |