#### Design Review Checklist Template

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Eduardo Sánchez | Date |  |
| Program |  | Program # |  |
| Instructor |  | 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 | * The program meets the requirements? * The output is what the user expect? * The User have an input entrance * The input is required * Do a manual call of all the design (corrida manual) * Is language agnostic? * The system can be escalated? |  |  |  |  |
| Logic | * The data types are thoughtful for scalability? * Proper data structures?   + Time complexity   + Size complexity * Proper algorithms?   + Time complexity   + Size complexity * Recursion functions have kickback? * Inputs are validated? * The functions have a correction skill for bad inputs? * All loops are not infinite? * Is the design achieving the scope of the problematic? * Is the strategy added to the functions? * If sql DB   + The schema of the database is correct? (scalability).   + Are we validating all the use cases? |  |  |  |  |
| Names | * Variables with proper names and self-explanatories. * Function with proper names and self-explanatories. * If the name is too general use comments * Camel Syntaxes * BEGIN and END of loops, sections, functions, classes. * In DB   + The tables or nodes are lowercase?   + Is normalized or un-normalized? |  |  |  |  |
| System Consideration | * Javator is setup with the correct src folder? * The JDK is updated? * The sockets or services for the connection to the DB are properly set or available? * Do we need another service? (ex: apache, gulp, etc) |  |  |  |  |
| Functional Use | * All the functions methods have a strategy or are self-explanatory? * All the buttons are used? * Easy input? * The input is tolerable to the user? * Clear output? (What the user wants) * The GUI or UI is usable? * The system is reusable or have a interface ability? * Is the design modular? (For maintaining and scalability) * The design will support the stress of multiple users? |  |  |  |  |
| External Limits | * Verify the inputs to the problem in the limits of them. * Is the program verify for outer limits? |  |  |  |  |
| Internal Limits | * In the use of loops, verify if the limits are correctly * What’s the behavior in external limits? (Jan/01/1970) |  |  |  |  |
| Special Cases | * Limit cases for inputs are validates? * Are the errors handled? |  |  |  |  |
| Names | * Are we following the code standard for the Camel? * Are the names of functions and variables self-explanatories? * The really complex variables names have a comment with explanation? |  |  |  |  |
| Language Syntax’s | * Are all the primitive’s data types correctly used? (ex: double or Double)? * Are all the methods called in proper way? * When declaring an Object, is initialized properly? |  |  |  |  |

**Code Review Checklist Template**

|  |  |  |  |
| --- | --- | --- | --- |
| Student | Eduardo Sánchez | Date | 04/08/2016 |
| Program | 7 | Program # | 7 |
| Instructor | Patricia 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. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | main | Input | Output | Probe | Program | Student |
| Complete | * Code covers all of the design | Y | Y | Y | Y | Y | Y |
| Imports | * Please insert the proper imports, just the one you NEED!!! | Y | Y | Y | Y | Y | Y |
| Main | * Is the main in the proper class? | Y | - | - | - | - | - |
| Classes Constructors | * Is well wrote? * All the variables are initialized? * Are u using this. Just when needed? | Y  Y  Y | Y  Y  Yy | Y  Y  Y | Y  -  y | Y  Y  Y | Y  Y  Y |
| Classes Variables | * Are all the variables private? * Do they have getters && setters? | Y  Y | Y  y | Y  Y | -  - | Y  Y | Y  Y |
| Classes functions | * Output correct? * Input correct? * Name correct? * Comments? | Y  Y  Y  N | Y  Y  Y  N | Y  Y  Y  N | Y  Y  Y  N | Y  Y  Y  Y  Y | Y  Y  Y  Y |
| (), {}, [] | * Begin && end? | y | Y | Y | Y | Y | Y |
| Output formatters | * If println: Correct capitalization? * Printf: Correct the sequence of values? * Is the GUI outputters correct | -  -  y | -  -  y | -  -  y | Y  Y  - | Y  Y  Y | Y  Y  Y |
| Logic operators | * Check every logic | y | Y | Y | Y | Y | Y |
| Line-by-line check | * Instruction syntax * Proper punctuation ;;;; | Y  y | Y  y | Y  y | Y  y | Y  y | Y  y |