**Checklist de Revisión del Diseño**

|  |  |  |  |
| --- | --- | --- | --- |
| Estudiante | Adrián García López | Fecha | 9/3/2020 |
| Programa |  | # Programa |  |
| Instructor | Adriana González Ugalde | Lenguaje | C++ |

|  |  |
| --- | --- |
| Propósito | Guiarte en la conducción de una revisión efectiva de tu diseño |
| General | * Escribe en el primer renglón el nombre de las partes que vas a revisar (aquellas que son nuevas o que van a ser modificadas) * Revisa tu diseño parte por parte. No continúes con la siguiente parte hasta que no hayas revisado por completo la parte anterior. * Cuando estés revisando una parte, revisa una categoría a la vez * Cuando termines de revisar una categoría para una parte, márcala como revisada |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre de las partes: | | Base | Nuevas | Reusadas |  |
| Getters y setters | Revisar que siempre se hayan colocado getters y setters de variables individuales y de la clase entera (si es necesario) | X |  |  |  |
| Clases | Asegurarse de que cada clase tenga un propósito especifico y reducir el mayor numero de clases posibles. | X |  |  |  |
| Inclusión de librerías | Verificar que cada librería incluida tenga el propósito de ser usada de forma correcta y eficiente. | X |  |  |  |
| Optimización y eficiencia | Desarrollar código de la forma más optima y eficiente de manera que la complejidad algorítmica sea la menor posible utilizando algoritmos para reducir tiempo de ejecución | X |  |  |  |
| Redundancia | Evitar a toda costa redundancia de código y si es necesario repetir, crear una función que se encargue de reutilizar el código previamente implementado | X |  |  |  |
| Autodescripción | Evitar nombres de variables ambiguas (Ej. No tener variables llamadas var) y funciones con verbos y variables (Ej. validChar(char) es una buena función) | X |  |  |  |
| Destructores | Considerar la destrucción de un objeto en caso de que este no sea utilizado durante todo el código. | X |  |  |  |