10 principios que todo desarrollador deberia de seguir

KISS

Recordemos que los sistemas(en sofware, maquinaria o cualquier otro sistema) sencillos tienden a fallar menos, son mas mantenibles y faciles de entender, y que es pricisamente lo que KISS (keep it simple, stupid-- manten esto simple, tonto) trata de denotar. A simple vista pareceria sencillo de aplicar, sin embargo es muy comun encontrar en desarrollo de software logica retorcida, lineas revueltas de codigo e interminables scripts de codigo ilegibles debido en principio a que muchas veces al momento de escribir codigo, el desarrollador esta tan concentrado en resolver el problema que tiene asignado y olvida simplemente hacer un codigo elegante y limpio.

La aplicacion de este principio se puede llevar a muchos niveles, desde subdivir metodos dentro de una clase, realizar componentes pequeños y que solo realicen una funcion hasta grandes arquitecturas donde cada uno de los componentes tienen una funcion en expecifica. Recordemos que el desarrollo de software se basa en la union de muchos elementos pequeños y sencillos que trabajan en conjunto y realizan una tarea compleja.

|  |  |
| --- | --- |
| public class A{  private Boolean active;//by default null  public void setActive(Boolean actv){  this.active = (actv==null)?false:actv;  }  public Boolean isActive(){  return (actv==null)?false:this. active;  }  } | public class B{  private boolean active;//by default false  public void setActive(boolean actv){  this.active = actv;  }  public boolean isActive(){  return this.active;  }  } |

Como se aprecia en el ejemplo la clase B es mas sencilla,entender y mantener, en este caso si no es necesaria a existencia de un estado nulo es mucho mas sencillo usar un atributo primitivo en lugar de usar un objeto de tipo Boolean.

DRY

Don’t Repeat YourSelf, ataca a la duplicidad de codigo pues es comun que en ocasiones se tengan multiples secciones de codigo que realizan lo mismo en diferentes lugares del proyecto, pero inclusive dentro de la misma clase; la ventaja de evitar la duplicidad del codigo es evidente cuando se quiere modificar, adecuar o arreglar el codigo pues solo se tiene que hacer en un solo lugar, incrementadon la productividad, facilidad para implementar purbas unitarias y entregar una mejor calidad de codigo. Las recomendaciones para evitar duplicidad en multiples secciones del proyecto son auxiliarse de clases de costantes para valores duplicados en multiples lugares, aplicar abstracciones donde y cuando sea posible y utilizar argumentos genericos en lugar de clases fijas.

CustomGeneric logger example

SINGLE RESPONSABILITY