**Solid**

**Ogólnie:**Zasady pozwlające zając się większością problemów związanych z designem software’u.  
Akronim od pierwszych liter 5 zasad, o których później.  
Wymyślone przez Feathersa.  
S- single responibility princ  
O- open clodes princ  
L- Liskov substitution princ  
I- Interface segregation pric  
D- Dependency inversion princ  
Bez tego kodu nie można testować; wiele się będie powtarzać, wydłużymy czas implementacji; zajmuje więcej czasu na implementacje i może stworzyć nowe bugi  
  
**Single Responsibility Priniciple:**Klasa powinna mieć tylko jeden powod do zmiany  
-każda klasa i moduł skupia się na jednym zadaniu na raz  
-wszystko w klasie powinno być w calu tego jednego zadania  
-klasy są mniejsze ale czystsze   
-kod jest mniej podzielony   
**Open closed principle:**-otwarte dla rozwijania, zamknięte dla modyfikowania  
-powinniśmy rozwijać klasę ale bez zmiany specyfikacji interfejsu (np. przy poliformiźmie tak zmieniąc klasę dziedziczące żeby nie musieć przerabiać interfejsu)  
**Lisov substitution principle:**-powinniśmy używać dowolnego childa polimorficznie bez martwienia się którego używamy   
-argumenty i zwracane wartości childrenów są zgodne ze sobą i z parentem   
**Interface segregation principle:**-lepiej mieć wiele jednozadaniowych interfejsów niż jeden wielozadaniowy  
-będziej customalizowalne   
**Dependency inversion principle:**-komponent wyższy nie może polegać na niższym,a oba powinny na interfejsie