A in function dosomething, there should not emerge any problem & all processes should work smoothly as previously. Example! class Video Player public function play (Afile) 11 play the video class Avivideoplayer extends Videoplayer public function play (still). if (pathinfo(file, PATHINFO_EXTENSION!== 'ovi') throw new Exception; //violates the LSP Here highlighted code is responsible to violate the LSP. As VideoPlayer class play method has same execution & neturn type of all types of vid explayers but it's child class Arikideoplayer play method thow exception for non-avi video playess. to be be the state of the state of -126-

Jon 1941 Do 106/2017 Example: 10 102 2011 11 100 oder: borneil spill interface Lesson Repository Interface * Fetch all records : ... 2 4. Orefum array - 1 11 11 11 11 11 11 199 bublic function getAU(); class FileLessonRepository implements. LessonRep ositonyInterface · bublic function: getAll() on all a to se 11 return through filesystem return []; 10 -112 (3) 1.00 (10) (10) (10) class DbLessonRepository implements LessonReposi tonyInterface public function getAille low risks with sold and return Lesson: all ; //violates the LSP sheturn Lesson: all () = to Array(); function too (lesson Repository Interface) Alesson) Cofference can use both class object i'e. either FileLessonRupository or DolessonRepository.

106/2017 Mere highlighted code in Ablesson Repository class widates the LSP. As first child class Filelesson Repository return array which is ok but second child class blesson Repository of Lesson Repository Interface returne Callection class object which is wrong. All child classes Of same interface must return same type of output. Principle 4. Interface Segregation: A client should never be forced to implement an interface that it does not use. clients should not be forced to depend on metho ds they do not use. Example: mother 2015, appoint on the oil interface Manageable Interface Distriction public function benanaged (); interface Workable Interface ion work(); - ? noise. mit silland interface sleepablemterface : 12 100 1000 Sublic function slupe; Enclosed the Comment of work is a series A not with The darkens and the work work the 20 for the all a thort

more than a most of the soul Data(201) class Humanworker implements workable Interface, sleepable Interface, Manageable Interface bublic function work () 11 implement work method public function sleep() return human sleeping; to any state of the public function bepranaged (). Sthis - work (); had made adding and sthis - sleepel; down and, board of income object to a more a transfer class Andraid Worker implements workable Interface ManageableInterface Dublic function work() Consider mires Total - 12 neturn landroid working!; public function temanaged () WHAT BALLEY TO THE THE WAS THE ad. Ithus - work (); - in or to me and out of all and 2, motions of the control to water of the constant tout a worker - - - - 29 -

decouple-STAMM - STANT ARTT 106/201) of the special series and the special security .. class Captain are the second of the cape of bublic function manage (manageable Interface) · swaker) sworker + bepranaged(); 5. Dependency Inversion Raincible: Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module; but they should depend on abstractions. This principle allows for decoupling, following example seems like the best way to explain this principle. class Password Reminder private & do Connection; bublic function -- construct (MySQLConnection & old Connection) Sthis-dolonnection = &dolonnection; Bublic function topological First the Mysollonnection is the low level modifile while the Password Reminderic high level, but according to the definition of Din S.O.L. I.D. which states that 'Depend on Abstraction not on concreti -120-