Liskov Substitution Principle

If S is a subtype of T then objects of type T can be replaced with objects of type S without altering the desirable properties of T.

Class variants

* Interface
* Type
* Implementation Class

Operations Preconditions & Postconditions

An operation is a service that is requested from an object which affects its behaviour.

Pre conditions are on non local variables that are taken by a method and must be true

Post conditions are on non local variables after execution of a method

Class Invariants

Relationship between non local variables that a function does not change

Rule 1. class invariants must be unchanged from the base classes and subclasses and must be forced as a constraint upon the subclass where parameters are forced to be similar to type T

Rule 2. Pre and post conditions must be satisfied and asserted as true always from the substited class types.

The Extended class does not do anything which violates the LSP because it should have some function that can replace the base class.

If there are no variables placed into the class speed settings then o will be returned which is erroneous and will cause the distance speed to divide by zero.

The adjustspeed function should be able to do something. Also the time to go speed should be divided over the variables and if not have a function to check if the vehicle is moving otherwise return a statement sayinh that the vehicle is stationary rather than dividing by zero.

For the sequence diagram there is a general thread of execution to be followed. The customer withdraws cash function is sent to the teller which then activates the get baclance function and looks up the bank account id. The balance is returned and the account is debited with the cash now being sent out to the customer object.

Ordinary State Machines issue some behaviour in a transitional mode with events and actions and activities

Protocol STate Machines have rules which govern communication between the various states. The machine user declares what operations can happen in without having to say anything about the behaviour

Flexibility

* Adding further functionality without having to change existing code

Reusability

* Methods are reusable in other classes

Efficiency

* Design for flexibility and reusability
* Implement correctness and robust systems

Correctness

* Goal is to satisfy desgin requirements for the application in all areas of a program
* Using maths to provide evidence that functionality is fully covered.#

Robustness

* Handle anomalous situations. More robust programs handle alot of of abnormalities
* Control faulty input and developer errors.
* Fail-safe attitude

Use Cases

Monitor Patient

Create File report

Create Patient Record

View Patient History

Cockburn Style

Use case Name

Goal

Scope

Preconditions

Post conditions

Primary Actor

Secondary Actors

Description

Extensions

Variants

Class Responsibility Collaboration Cards

Class Name

List of Responsibilities Collaborators

OO Principles

* Encapsulation
* Abstraction
* Inheritance
* Polymorphism

SOLID Principles of OO Design

* Single Responsibility Principle
* Open/Closed Principle
* Liskov Substitution Principle
* Interface Segregation Principle
* Dependency Inversion Principle