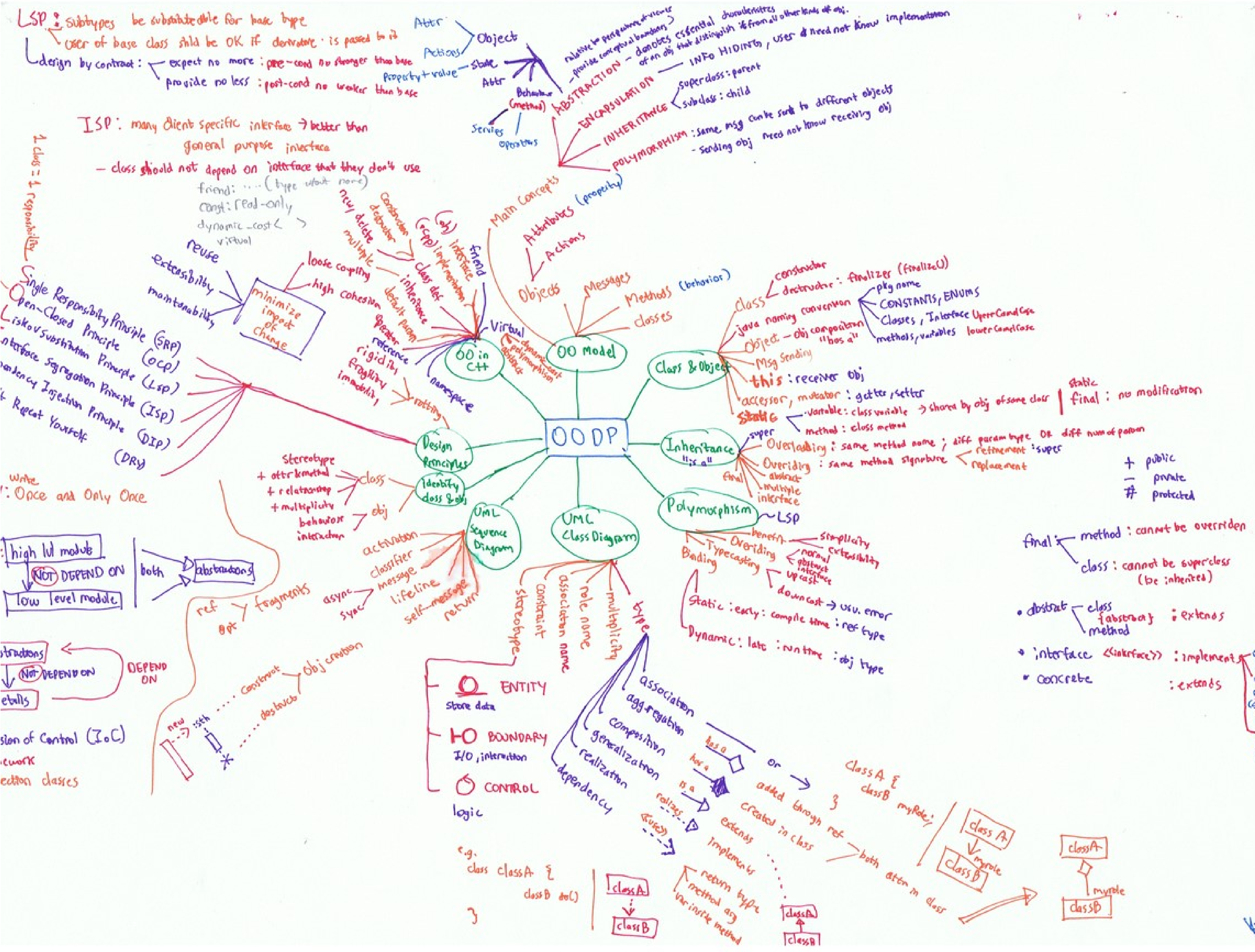
OODP
LSP: subtypes be substitutable for base type
user of base class shld be OK if derivative is passed to it
design by contract: expect no more: pre-cond no stronger than base
provide no less: post-cond no weaker than base
ISP: many client specific interface → better than general purpose interface
- class should not depend on interface that they don't use
friend: ... (type about none)
const: read-only
dynamic_cast< >
virtual
1 class = 1 responsibility
Single Responsibility Principle (SRP)
Open-Closed Principle (OCP)
Liskov Substitution Principle (LSP)
Interface Segregation Principle (ISP)
Dependency Injection Principle (DIP)
Don't Repeat Yourself (DRY)
Write Once and Only Once
reuse
extensibility
maintainability
minimize impact of change
loose coupling
high cohesion
rigidity
fragility
immobility
rotting
Design Principles
Object
Attr
Actions
State
Property + value
Behaviour (method)
Services
Operations
ABSTRACTION - denotes essential characteristics of an obj that distinguish it from all other kinds of obj.
ENCAPSULATION — INFO HIDING, user need not know implementation
INHERITANCE: superclass: parent; subclass: child
POLYMORPHISM: same msg can be sent to different objects - sending obj need not know receiving obj
Main Concepts
Attributes (property)
Actions
Objects
Messages
Methods (behavior)
classes
OO Model
OO in C++
Virtual
dynamic cast
polymorphism
abstracts
namespace
friend
interface
implementation
class def
inheritance
default param
operator
reference
multiple
new, delete
constructor, destructor
Class & Object
class: constructor; destructor: finalizer (finalize())
Java naming convention: pkg name; CONSTANTS, ENUMS; Classes, Interface UpperCamelCase; methods, variables lowerCamelCase
Object - obj composition "has a"
Msg Sending
this: receiver obj
accessor, mutator: getter, setter
static: variable: class variable → shared by obj of same class; method: class method
static final: no modification
Inheritance "is a"
super
Overloading: same method name; diff param type or diff num of param
Overriding: same method signature: refinement: super; replacement
final
abstract
multiple
interface
+ public
- private
protected
Polymorphism
LSP
benefit: simplicity, extensibility
Overriding: normal, abstract, interface
Typecasting: upcast, downcast → usu. error
Binding
Static: early: compile time: ref type
Dynamic: late: run time: obj type
final: method: cannot be overriden
class: cannot be superclass (be inherited)
abstract: class {abstract} method: extends
interface <<interface>>: implements
concrete: extends
Identify class & obj
class: stereotype, + attr & method, + relationship, + multiplicity
obj: behaviour, interaction
UML Sequence Diagram
activation
classifier
message: async, sync
lifeline
self-message
return
fragments: ref, opt
obj creation: new, constructor, destructor
UML Class Diagram
stereotype
constraint
association name
role name
multiplicity
type
ENTITY: store data
BOUNDARY: I/O, interaction
CONTROL: logic
association
aggregation: has a
composition: has a
generalization: is a
realization: realizes
dependency: <<uses>>
or
added through ref
created in class
extends
implements
both attr in class
return type, method arg, var inside method
e.g. class classA { classB do() }
class A { classB myRole; }
classA
classB
myrole
high lvl module
NOT DEPEND ON
low level module
both
abstractions
DEPEND ON
details
Inversion of Control (IoC)
framework
injection classes