Low Level Design - 101

Logistics

- assignment / homework deadline: no such thing. Strongly recommend that you complete the assignment + homework before the next lecture.
- for any batch, instructors will keep changing - hpowever we will make sure that a given topic is taught by the same instructor
- HLD assignment
    - not going to be graded
    - you to get a feel for what HLD questions look like
- Are the assignments linked to placement assistance?
    - yes & no
- for any dashboard/other issues, please email at support@scaler.com

---

What and Why of LLD

- what
    - closer look at the application you're creating
    - closest possible: individual lines of code
    - 80k lines of code - enjoy!
    - Programming Paradigms
    - SOLID principles
    - Design Patterns
    - ER-Diagrams
    - Database Schemas
    - MVC
    - Clean Architecture
- why
    - LLD allows us to make our code better
    - 10-15% of a dev time is spent writing code
    - Maintainance
        - testing
        - debugging

        - refactoring
    - Read, Review and Research
        - looking at other people's code
            - PR
            - KTs
            - understanding what they've done to learn from it
        - looking at your own code
        - writing code is easier than reading code
    - meetings
    - canteen
    - netflix, ...

Aim

Code should be

1. Maintainable
    - Testable
    - Debuggable
    - Refactorable
2. Readable
    - transparent/obvious code/boring
    - self-explanatory

```
// declare an array of size 10
int ar[10];

for(int i = 0; i < 10; i++) {
    // run a loop from i = 0 to i = 10
    ar[i] = ar[i] * ar[i];
    // multiple each value by itself and store in array
}

for(int i = 0; i < 10; i++) {
    // run a loop from i = 0 to i = 10
    print(ar[i]);
    // get the ith element and print it
```

```
}


// declare an array of size 10
int ar[10];


// square each element
for(int i = 0; i < 10; i++) {
    ar[i] = ar[i] * ar[i];
}

// print
for(int i = 0; i < 10; i++) {
    print(ar[i]);
}


void squareElemnents(int ar[]) {
    for(int i = 0; i < ar.length; i++)
        ar[i] *= ar[i];
}
void printArray(int ar[]){
    // ...
}


int ar[10];
squareElemnents(ar);
printArray(ar);
```

3. Extensible
    - requirements change all the time
    - the code you write should
        - anticipate future requirement changes and structure itself in a manner that incorporating those changes is simpler

---

How on earth do we achieve these 3 points?

- Programming paradigms - Object Oriented Programming (OOP)

- Guidelines / Principles that others have figured out - SOLID principles
- Some standard - widely accepted & battle-tested solutions to common problems
    - Design Patterns
- How to break down a vague problem and make it executable
- around 1.5 months of LLD classes

Agenda for today

- do a shallow dive into OOP

Are we going to learn LLD using Java?

- yes, we will follow Java
    1. Java is the de-facto standard for OOP
    2. Java is highly in demand
- whatever we learn here will be language agnostic

Programming Paradigms

- Object Oriented
- Procedural
- Functional
- Declarative
- Imperative
- Data Driven
- Reactive / event driven
- Asperct Oriented

Most modern programming language are multi-paradigm in nature

---

The usual way we write code - Procedural

Top-to-Bottom

```
void calculator() {
    do {
        print('Gimme 2 numbers');
        int a, b = // get from input
        print('Gimme an operator');
        char operator = // get from input
        if(operator == '+') {
            print(a + b);
        } else if(operator == '*') {
            print(a * b);
        }
        print('Do you wanna do another calculation?')
        char choice = // input
    } while(choice == 'y');
}
```

Object Oriented Programming

- tries to mimic the waqy we solve problems in real life
- how to tackle a big problem?
    - break it down into smaller pieces
    - solve the smaller pieces
    - combine the solutions into the larger solution
- how do we break a problem down?
    - Concepts / Ideas
    - build a repository of these concepts
    - we might combine some concepts to create yet other concepts
- Concept
    - Smartphone is a concept
        - is it real? is it a physical thing?

            - no - it is a thought that you have
        - OS, Apps, Features, Call, Camera, Feel, Touch Screen, Security, ...
        - Attributes/Features of the concept “Smartphone”
        - Behaviros of the concept “Smartphone”

- Smartphone
    - attributes
        - size
        - design
        - color
        - screen size
        - resolution
        - battery life
        - build - plastic / metal
        - camera quality
        - operating system
    - behavior
        - send & recieve calls
        - browse the internet
        - install and use apps
        - take photos/videos/audios
        - use the gps to navigate
        - listen to music

- Battery
    - attributes
        - type - Li-ion, Hydrogen, Gravity, Lead-acid
        - size
        - capacity
        - the number of cycles it can sustain
        - weight
        - output voltage and current
        - charagable
        - time to charge from 10 - 100%
        - manufacturer
        - warranty
        - number of charging ports
    - behavior
        - get charged

        - charge other things - it will itself get discharged
        - heat up
        - blow up
        - degrade over time
        - plugged into a device
        - consume physical space

- Relationships b/w concepts: Smartphone has-a/uses-a battery

---

Concept
- attributes / physical properpties
- behavior / functionality
- relationships with other concepts
+ has-a / uses-a
* Bird sits on a Branch
* Bird has-a Leg
* Smartphone has-a touchscreen
* association (always bi-directional)
- composition (ownership - directional)
- aggregation (no ownership)
+ is-a
* Bird is-an Organism
* Bird is-a LivingThing
* Smartphone is-aa ElectronicDevice

Objects
- physical/real things
- touch and feel and hold
- they consume space
- each object can be related to one or more concepts
+ smartphone
+ electronic device
+ communication device
+ camera
+ brand

Object Oriented Programming

- we think of programs/solutions in terms of concepts and objects
- concept <-> class

```
// concept / idea / blueprints
class Smartphone {

    // member variables <-> attributes

    int height, size;
    Color color;

    // has-a relationships <-> member variables
    Battery battery;
    Touchscreen screen;

    // behavior
    // methods (not the same as functions)
    // things that you can do with smartphone
    void browseInternet() {}

    void makePhoneCall() {}

    // things that smartphone can do
    void onRecievePhonecall() {}

    void shutdown() {}

    void updateFirmware() {}
}

class Battery {

}

class Touchscreen {

}


// ---------------------------
// mapping an object to just 1 concept
// A concept will have many objects corresponding to it

Smartphone phone1 = new Smartphone();
// instantiating a class by calling the constructor / manufactured an objec
```

```
// object is an instance of a concept




Smartphone phone1 = new Smartphone();
//                      ^^^^^^^^^^
//                      Class/concept
//                  ^^
//                  constructor
//          ^^^^^^
//          variable/reference/name
// ^^^^^^^
// Class/Data Type
// Object exists in the memory
// variable phone1 is a name/alias/pointer that we are using to refer to th

// what is the object?
// lives in the memory
// consumes physical space (space in RAM)
// comes into existance only when the code is being actually run


// constructor in Java is a method that has the same name as the class
// constructor doesn't have a return type
class Calculator {
    int value;

    public Calculator() {
        this.value = 10;
        // this is the constructor
        // this is infact an "object-initializer"
        // this is the first thing that gets executed after you've created
    }
}



# the type() function is the actual constructor in Python

class Calculator:
    def __new__(cls) -> Calculator:
        obj = super().__new__()
        return obj

    def __init__(self):
        pass
```

- When the initalizer is being executed the object has already been constructed
- the initalizer can use the object
- In the constructor the object must manually be created

Method vs Function

- Method is simply a function that is tied to an object

10.25
back by 10.35

```
class Calculator {
    static int initialValue = 0;

    int lastResult;

    public Calculator() {
        lastResult = initialValue;
    }

    // this function can only execute if it is provided the content of some
    public void add(int value) {
        // this function can only be executed with some context
        // context is the value of lastResult
        // this is a variable referring to the "cuurent" object on which th
        this.lastResult += value;
        print(this.lastResult);
    }

    public void substract(int value) {
        lastResult -= value;
        print(lastResult);
    }

    public void multiply(int value) {
        lastResult *= value;
        print(lastResult);
```

```
    }

    public void divide(int value) {
        if(value == 0) {
            throw new Exception("Division by 0 is not allowed!");
        }
        lastResult /= value;
        print(lastResult);
    }

    public void setInitialValue(int value) {
        initialValue = value;
    }
}

// object has internal state
// we mean the current values of the attributes of that object

class Client {
    void main() {
        Calculator c1 = new Calculator();
        print(c1.lastResult); // 0
        c1.add(10);           // 10
        c1.multiply(20);      // 200
        c1.substract(50);     // 150
        c1.divide(3);         // 50
        c1.setInitialValue(500);

        // state of different objects is maintained separately

        Calculator c2 = new Calculator();
        print(c2.lastResult); // 500 (because it took the value from the "s
        c2.add(10);           //
        c2.multiply(20);      //
        c2.substract(50);     //
        c2.divide(3);         //

        // how do we share state between objects?
        // if we want the objects of the same class to share some variable
        // we can declrare that variable with the "static" modifier
    }
}

// VSCode
// Microsoft, Github, Open Source, [Github Co-Pilot]
```

Syntactic Sugar

- a language feature that is not strictly necessary - but just makes our lives a little simpler

---

Is-A Relationship

Inheritance

- When I say that A Sparrow is a Bird, what do I really mean?
  - Sparrow is a concept
  - Bird is a concept
  - All the attributes, behvaior, and relationships that the concept Bird has, the concept Sparrow also has all those ... along with some extra stuff
- Can I say?
  - Bird is living
  - Sparrow is Bird
  - Sparrow is non-living?
  - NO - this is incorrect!

Various ways of defining inheritance

if ChildConcept is-a ParentConcept
then

1. Everything (attribute, behavior & relation) that ParentConcept has, ChildConcept also has
2. Every object of ChildConcept is also an object of ParentConcept
3. The set of objects of ParentConcepts contains/is-a-superset of the set of objects of ChildConcept
4. A ChildConcept object can be used wherever a ParentConcept object is expected, but not vice-versa (Runtime Polymorphism)

> ### Example
>
> You can give a Sparrow object to a BirdKeeper, because everything that they will do with a Bird, a Sparrow also supports it
> But you can't give a generic Bird to a SparrowKeeper, because the SparrowKeeper might expect the Sparrow to fly, but not all Birds can fly!

```
class Bird {
    int height, weight;

    void feed() { print("peck at the ground"); }
    void sleep() { print("snore adorably"); }
}

class Sparrow extends Bird {

}
```

```
class Bird:
    height: int
    weight: int

    def feed(self):
        ...

    def sleep(self):
        ...

class Sparrow(Bird):
    ...
```

Inheritance vs Composition

- should we model concepts using is-a relationship or using has-a relationship
- composition is usually preffered, because it offers certain benefits, and in simpler to maintain

- SOLID
- Design Patterns
- Database Schema

Problem

> Model the concepts of Child, Father, Son, Daughter, GrandParent, Mother, Parent using OOP

```
class GrandParent extends Parent {
    List<Child> children;
    List<Child> grandChildren;

    void playWithGrandChildren() {}
}

class Parent extends Child {
    // parent is-a child
    // parent has-a parent

    List<Child> children;

    void drive() {}
}


class Child {
    Parent mom, dad;  // child has-a parent

    int age;
    String name;
    int weight;
    String gender;

    void sleep() {}
    void eat() {}
}


// there is a relationship b/w a parent and a child
// what kind of relationship is it?
// ans: both
```

```
// every child has-a parent // YES.
// every child is-a parent // NO.
// every parent is-a child // YES.
// every parent has-a child // YES. (imagine a parent who doesn't have a ch


// if A is-a B, then `class A extends B {}`


// has-a relationship are usually bi-directional
// a child has-a a parent
// a parent has-a child

// is-a relationships are Directional
// a Sparrow is-a Bird
// a Bird is-a Sparrow   // wrong


// if A can do everything B can do
// A is the sub-class where as B is the parent-class
// A extends B
```

Object-Relational impedance mismatch

- Most Relational databases have no concept of inheritance - they only support composition
- how to convert is-a relationship into a has-a relationship

DSA - 3 months
CS fundas - 1 month

LLD

Homework vs Assignment

- live lecture: solve several problems

- once you kwow the approach you should be able to fluently convert your thoughts into code
- assignment have the same problems that were discussed during the lecture
    - so you can practice implementation, debugging, managing edge-cases
- also want to learn “how” to think
    - homework are novel problem from the same topic, but problems we haven’t discussed in the lecture
    - gives you to opportunity to think for yourselves, and then code

---

School - by the Alumni

Hints, Solution Approach, Correct “Official” Solution, Best solution by other students for each language
Video explanation for each Homework question

+91 7351769231