Class Relationships:

1-Association Relation: class A uses class B(no ownership)ex: member read book
2-Aggregation Relation :(weak-has a): library has # of books (no ownership)
one object (the whole) has parts ,but doesn't control life-time of theses parts ,parts can exist independently .
3-Composition Relation(strong "owns-a")car owns 4 wheels, car owns one motor.
one object (whole) owns its parts completely ,if the whole dies the parts must die with it.

4-Inheritance-->"is-a"

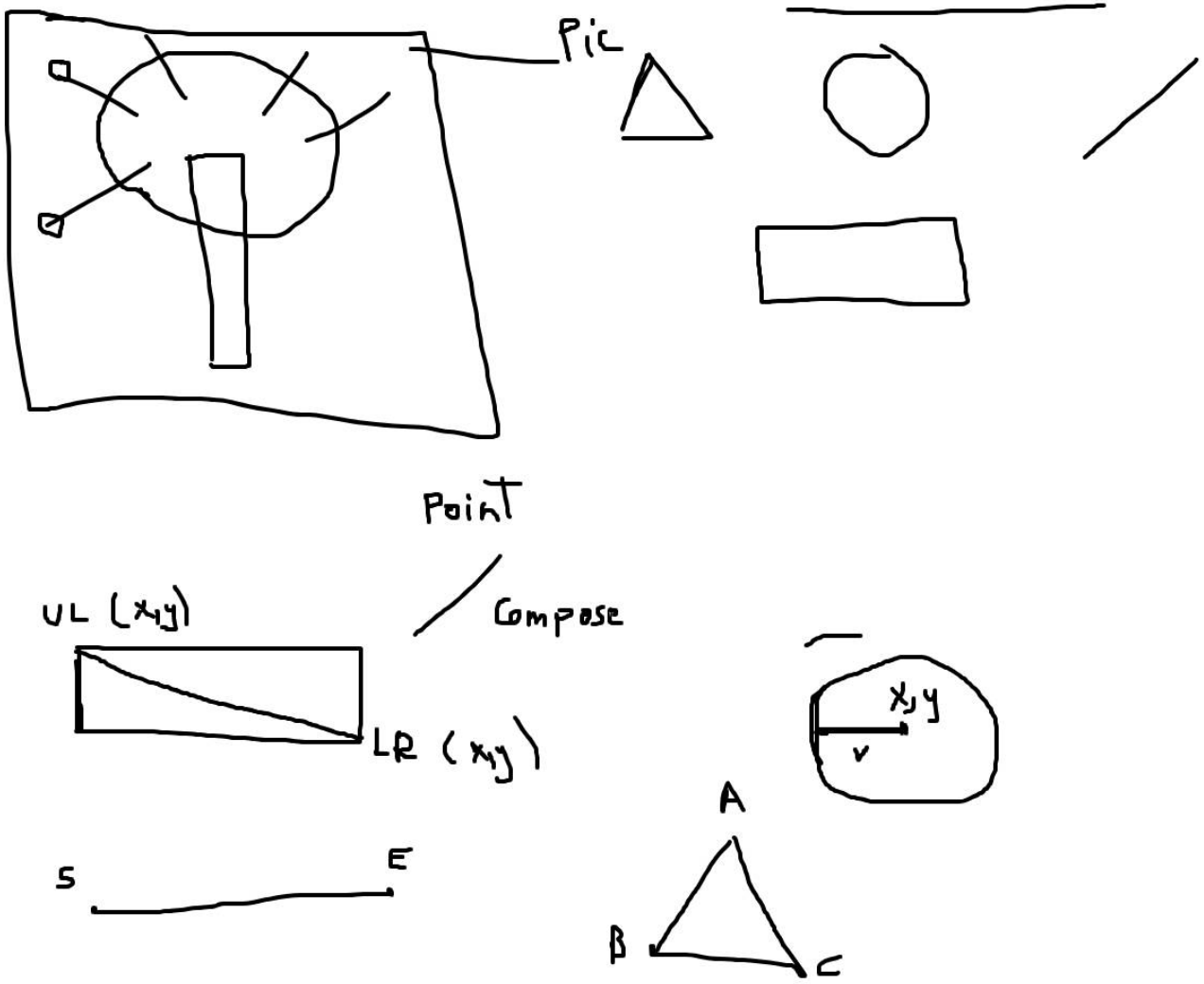
UML --> notations

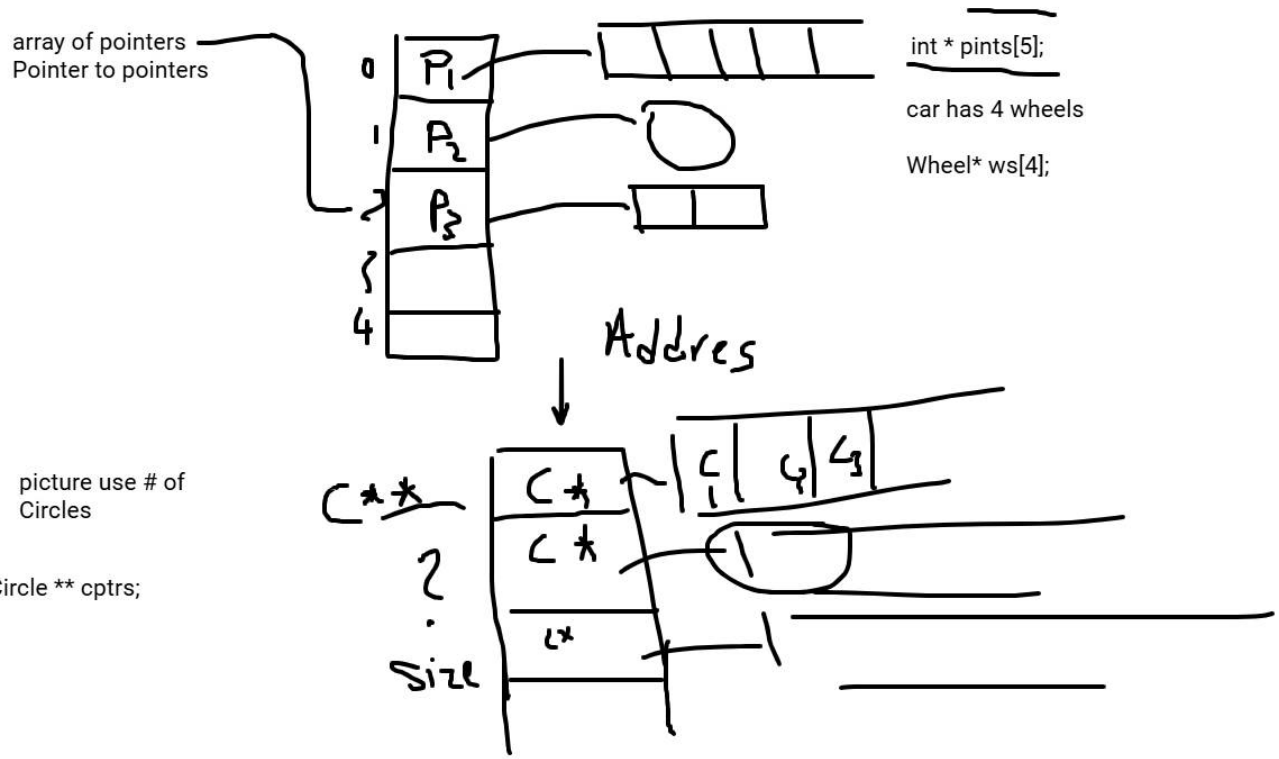
Association:
Member -------->(1---1-*) book
(*---0-*)
(1---0,1,2,3) Implement --->use pointer
Aggregation
Library <|—Books Implement ---->pointers

Composition
Book owns an Author
Book <|—— Author implement ---<use object ,dynamic allocated object (whole create new object ,and destruct this object)

```
#include <iostream>
#include<vector>
using namespace std;
class Author
{
   string name;
   string nationality;
public:
   Author(string n,string nat):name(n),nationality(nat) {}
   string getName(){return name;}
   string getNationality(){return nationality;}
};
//composition between Book and Author
class Book
{
   string title;
   int publish_year;
   //Author author;
   Author* author;
public :
   Book(string t,int y,string an,string
anat):title(t),publish_year(y)//,author(an,anat)
   { author=new Author(an,anat);}//dynamic allocation
   string getTitle(){return title;}
   int  getPubYear(){return publish_year;}
   string getAuthorName(){return author->getName();}
   string getAuthorNationality(){return author->getNationality();}
   ~Book(){delete author;}//must delete author when deleting book
};
//Aggregation Book & Library
class Library
{
   //vector of pointers ,Array of pointers dynamic ,pointer to pointers
   vector<Book*> books;
public:
   void addBooks(Book* b)
   {
      books.push_back(b);
   }
   void listBooks()
   {
         cout<<"Library books"<<endl;
         for(auto bb:books)
            cout<< bb->getTitle()<<"authored by: "<<bb-
>getAuthorName()<<"in
:"<<bb
->getPubYear()<<endl;

   }
};
/Association Member & book
class Member
{
   string name;
   int id;
public:
   Member()=default;
   Member(string n,int i):name(n),id(i){}
   void borrowBook(Book &b)
   {
      cout<<"Member:"<<name<<" borrowed
"<<b.getTitle()<<endl;
   }
};

int main()
{
   cout << "Hello Library!" << endl;

   Book b1("1948 ",1949,"George","British");
   Book b2("Farm Animal
",1955,"George","British");
   Book b3("Tales of 2 Cities
",1940,"Gee","Bri");
   //library
   Library lib;
   lib.addBooks(&b1);
   lib.addBooks(&b2);
   lib.addBooks(&b3);

   lib.listBooks();

   Member m("Ahmed",55);
   Member m2("sherry",66);
   m.borrowBook(b1);
   m2.borrowBook(b2);
   return 0;
}
```

Ex:
le's draw a picture that uses number of geometric shapes to create its drawing ,Design UML then implement classes.




array of pointers
Pointer to pointers
0
1
2
3
4
int * pints[5];
car has 4 wheels
Wheel* ws[4];
Addres
picture use # of
Circles
Circle ** cptrs;
size

```
#include "SimpleGraphics.h"
#include <iostream>
using namespace std;

#include <vector>

//----------------Shape classes using Raw Pointers-------------------


// ---------- Point ----------
class Point {
   int x{}, y{};
public:

   Point(int a, int b) : x(a), y(b) {}
   int getX()  { return x; }
   int getY()  { return y; }
};

// ---------- Circle ----------
class Circle {
   Point center;       // Composition (owns)
   int radius;
public:
   Circle(int x, int y, int r):center(x,y),radius(r) {
      //center = new Point(x, y;
      //radius = r;
   }

   ~Circle() {
      //delete center;
   }

   void draw()  {
      drawCircle(center.getX(), center.getY(), radius);
   }
};
      class Rect {
         Point ul, lr;      // Composition
      public:
         Rect(int x1,int y1,int x2,int y2):ul(x1,y1),lr(x2,y2) {
            //ul = new Point(x1, y1);
            //lr = new Point(x2, y2);
         }

         ~Rect() {
            //delete ul;
            //delete lr;
         }

         void draw()  {
            drawRect(ul.getX(), ul.getY(), lr.getX(), lr.getY());
         }
      };
class Picture {
   Circle** circles;   int cNum{};
   Rect**   rects;     int rNum{};
   Line**   lines;     int lNum{};
   Triangle** tris;    int tNum{};

public:
   Picture()
      : circles(nullptr), rects(nullptr),
        lines(nullptr), tris(nullptr) {}

   void setCircles(int n, Circle** arr) { cNum=n; circles=arr; }
   void setRects(int n, Rect** arr)     { rNum=n; rects=arr; }
   void setLines(int n, Line** arr)     { lNum=n; lines=arr; }
   void setTris(int n, Triangle** arr)  { tNum=n; tris=arr; }

   void paint() {
      for (int i=0; i<cNum; i++) circles[i]->draw();
      for (int i=0; i<rNum; i++) rects[i]->draw();
      for (int i=0; i<lNum; i++) lines[i]->draw();
      for (int i=0; i<tNum; i++) tris[i]->draw();
   }
};
```

1-Demonstrate the Example of Library & Books Author & Book--->using Raw pointer
---->static object inside Book,Dynamic (add copy constr and =
operator to avoid any runtime error)
using vector instead of Dynamic pointer in library

2-Shapes Example with Picture --->Dynamic & Static(object)
Main --- ask user to enter sizes of shapes to use in his picture ** Dynamic
Add copy ctr or + if needed
add Ellipse class and show how to draw

```
initScreen();   // clear ASCII screen

    Picture pic;

    // Add shapes (students focus on relationships)


    // Draw everything
    pic.paint();

    // Show screen
    renderScreen();
```