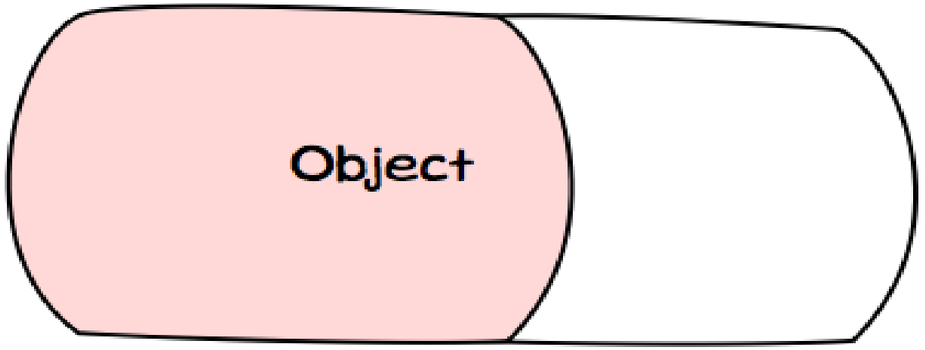
Object Collaborations

by Andrew Cain and Willem van Straten

SWINBURNE UNIVERSITY OF TECHNOLOGY

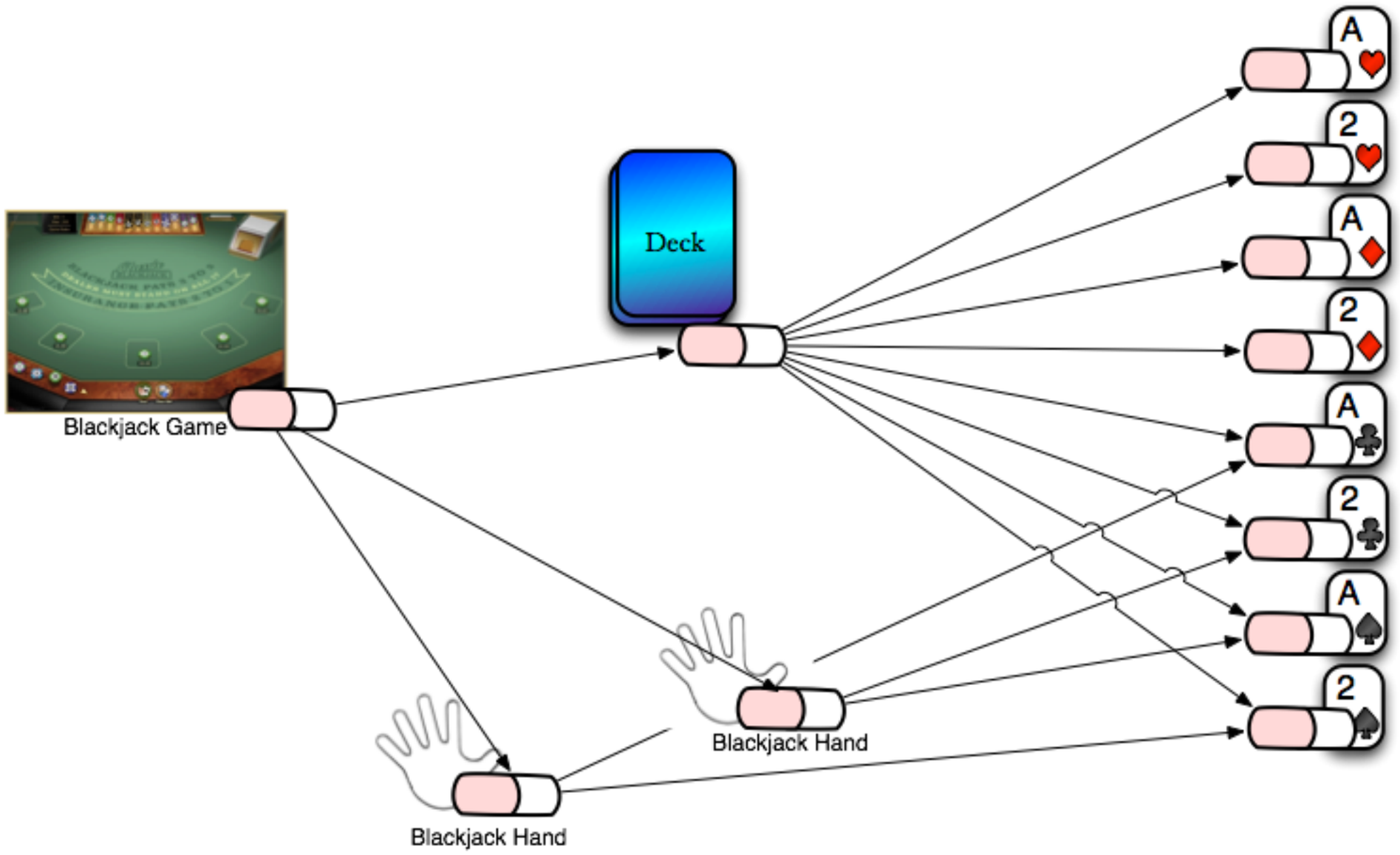
Object oriented programs are designed around the idea of objects



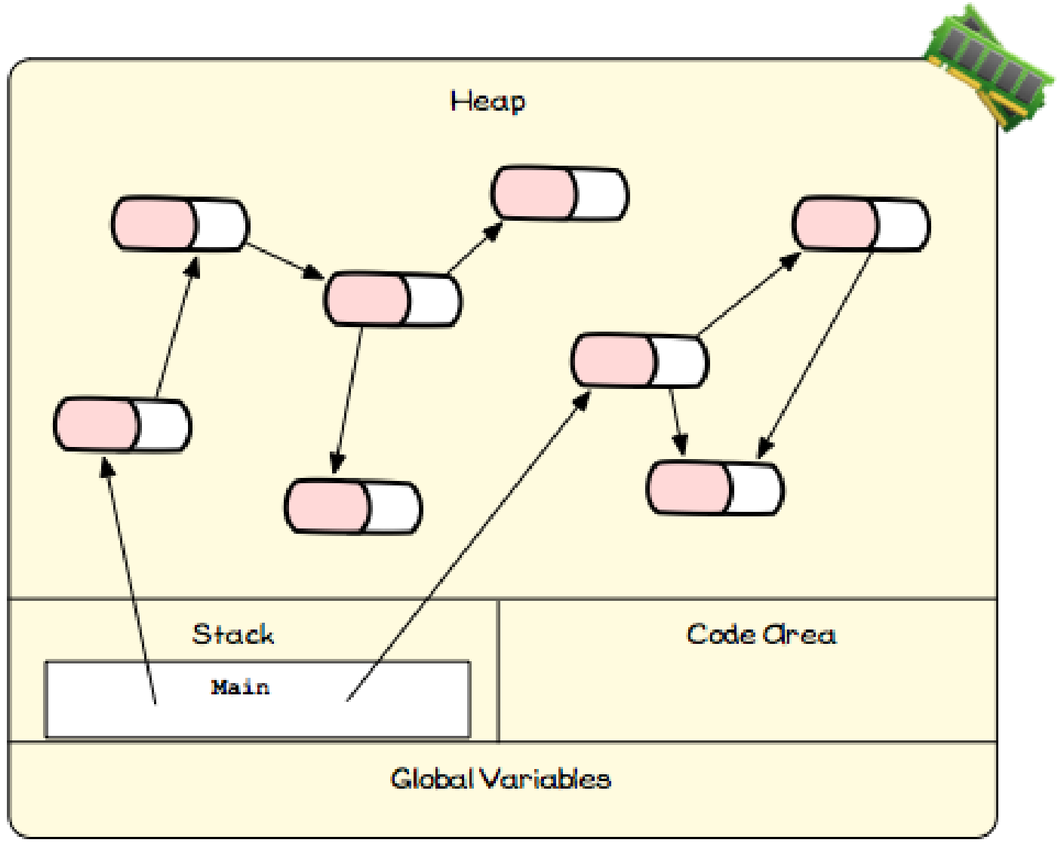
Objects in programming attempts to mimic real-world objects

Objects have attributes (*data*) and behaviour (*functions or methods*)

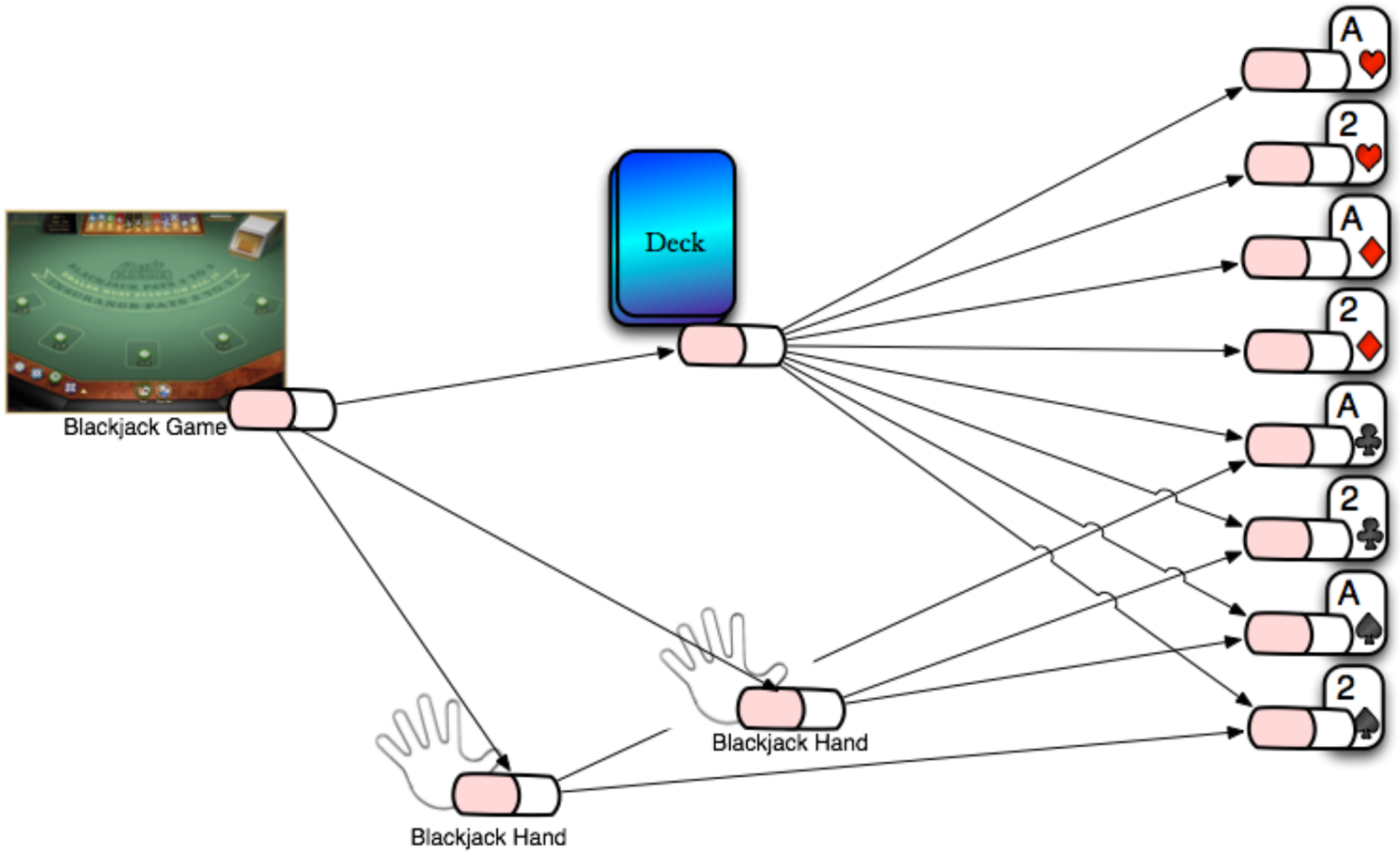
Developers use objects to create components for use in their software



Object oriented programs usually contain many objects of different kinds



To work effectively objects will need to interact to achieve program goals



Objects collaboration

- Objects collaboration: when two or more objects interact.
- Example: one object sending one message to another object (through passing parameters). Or dozens of objects exchanging messages requesting information or actions
- Entire application is like a single massive collaboration among the objects, creating a network of relationships.

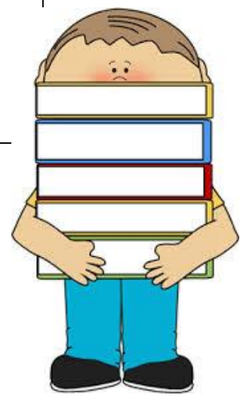
At runtime objects exist on the heap !!

Let's delve deeper

Computer's memory is organized into 3 segments

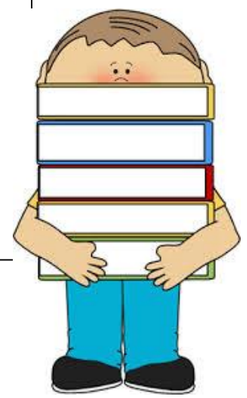
Text (code) segment

- compiled codes of the program itself resides



Stack segment

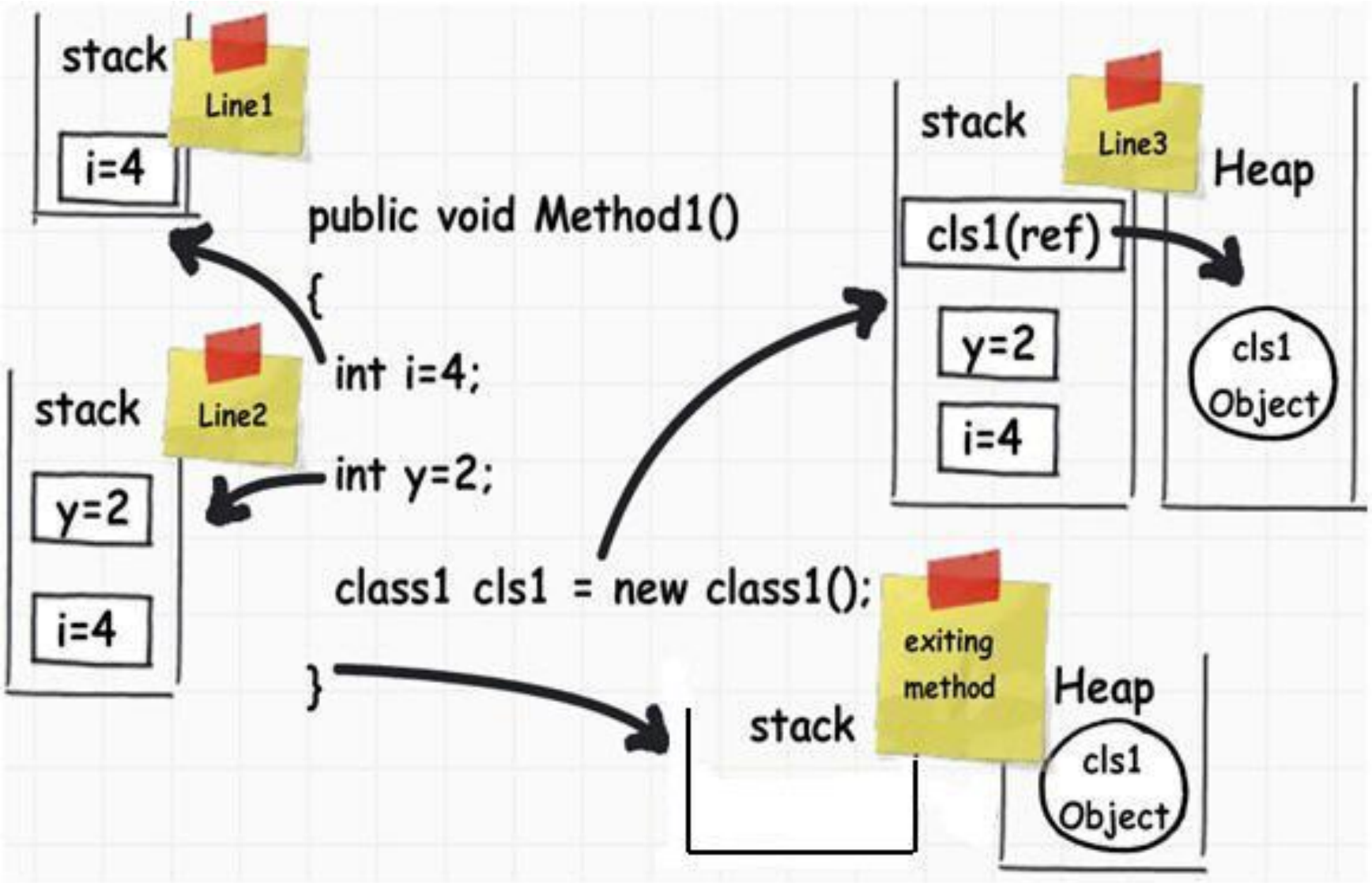
- Stores all the variables that are declared and initialized before runtime
- Uses Last In First Out (LIFO) method



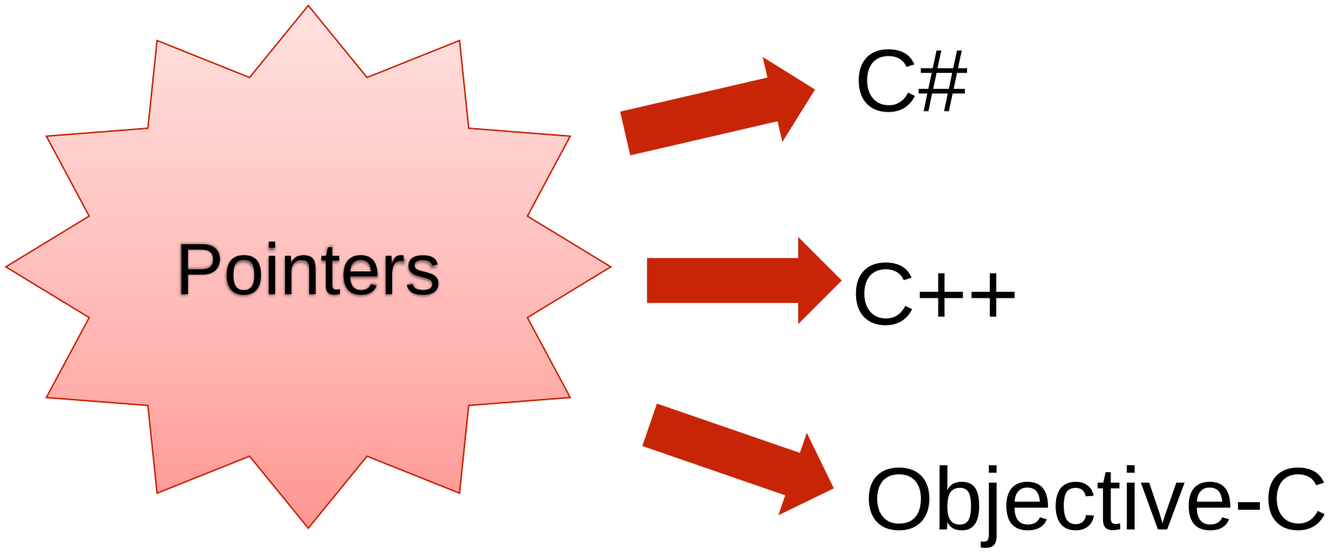
Heap segment

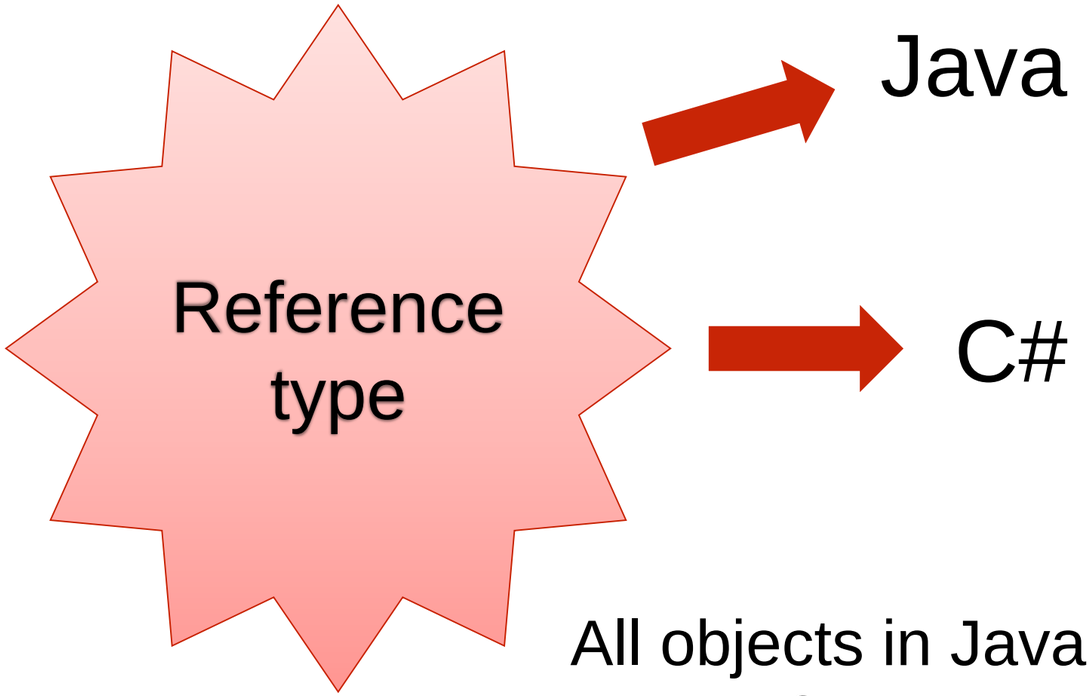
- Stores objects created at run time

Stack & heap


stack
Line1
i=4
public void Method1()
{
int i=4;
int y=2;
class1 cls1 = new class1();
}
stack
Line2
y=2
i=4
stack
Line3
cls1(ref)
y=2
i=4
Heap
cls1
Object
stack
exiting
method
Heap
cls1
Object

Languages use some form of pointer to refer to objects...

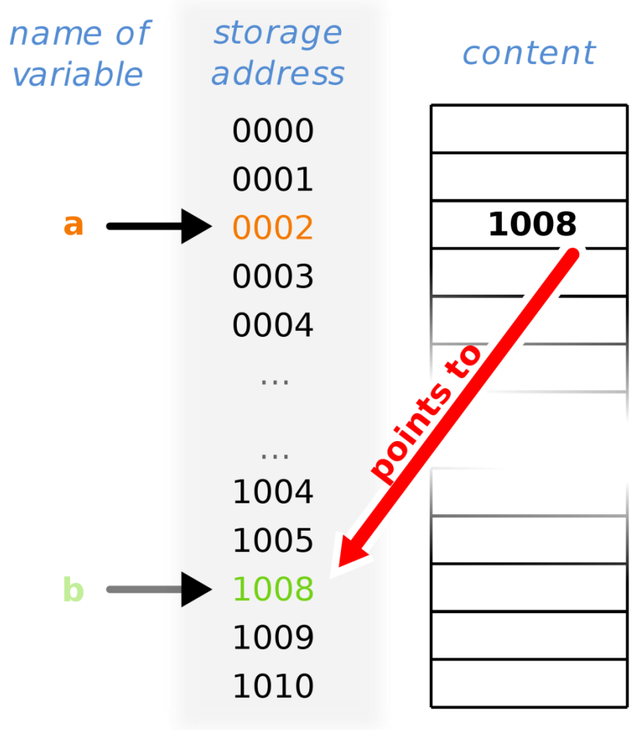




All objects in Java are references

What is a pointer ?

A pointer is a variable whose value is the address of another variable



How are pointers used?

C#

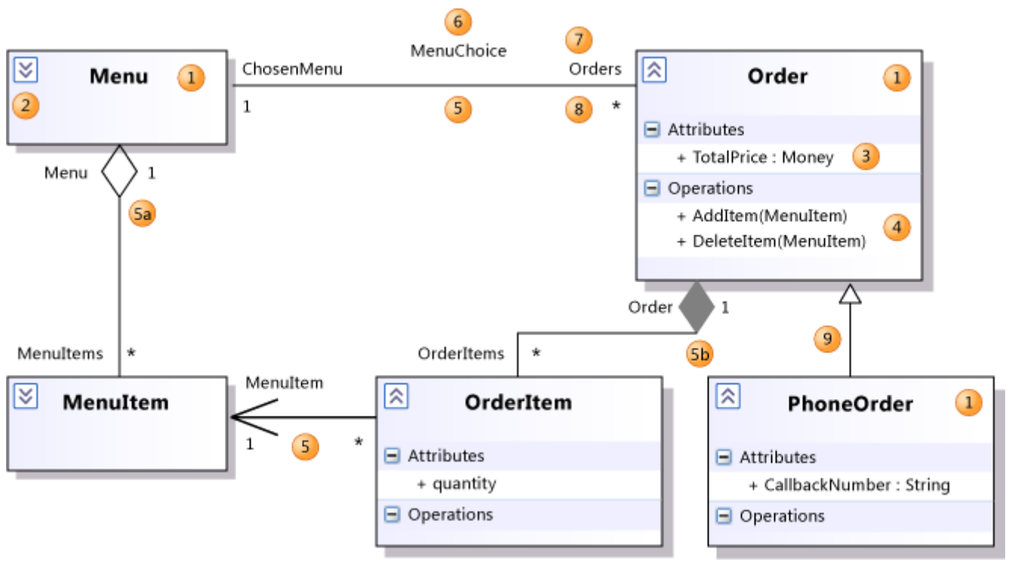
```
public void Method()
{
    int x = 10;
    int *ptr = &x;

    // Displays the memory address
    Console.WriteLine((int)ptr) ;

    // Displays the value at the memory address
    Console.WriteLine(*ptr);
}
```

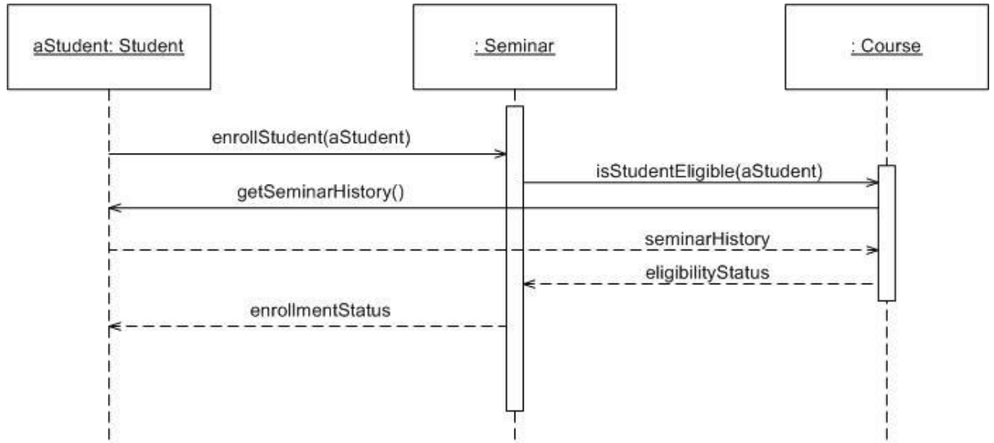
Java

```
class Obj {
    public int value;
}

public class pointers() {
    public static void main(String[] args)
     {
       Obj  x;  // Allocate the pointers x

       x = new Obj();   // Allocate an Obj pointee
                        // and set x to point to it

       x.value = 42;   // Dereference x to store 42 in its pointee


    }
}
```

Design interactions between the objects in your solution

- using UML diagrams and notations





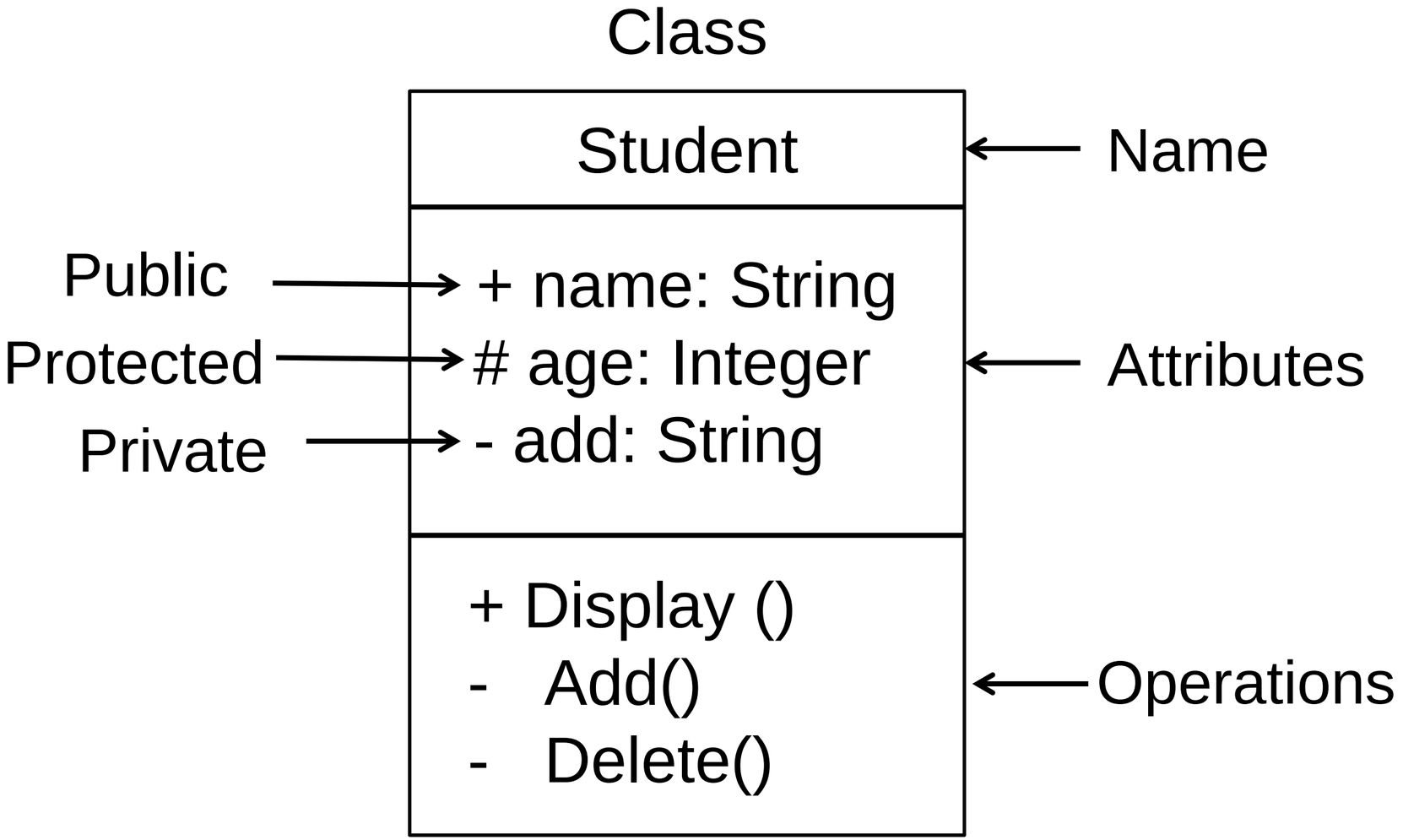
Unified Modelling Language (UML)

- UML is a way of describing a software program (object-oriented) using a collection of diagrams

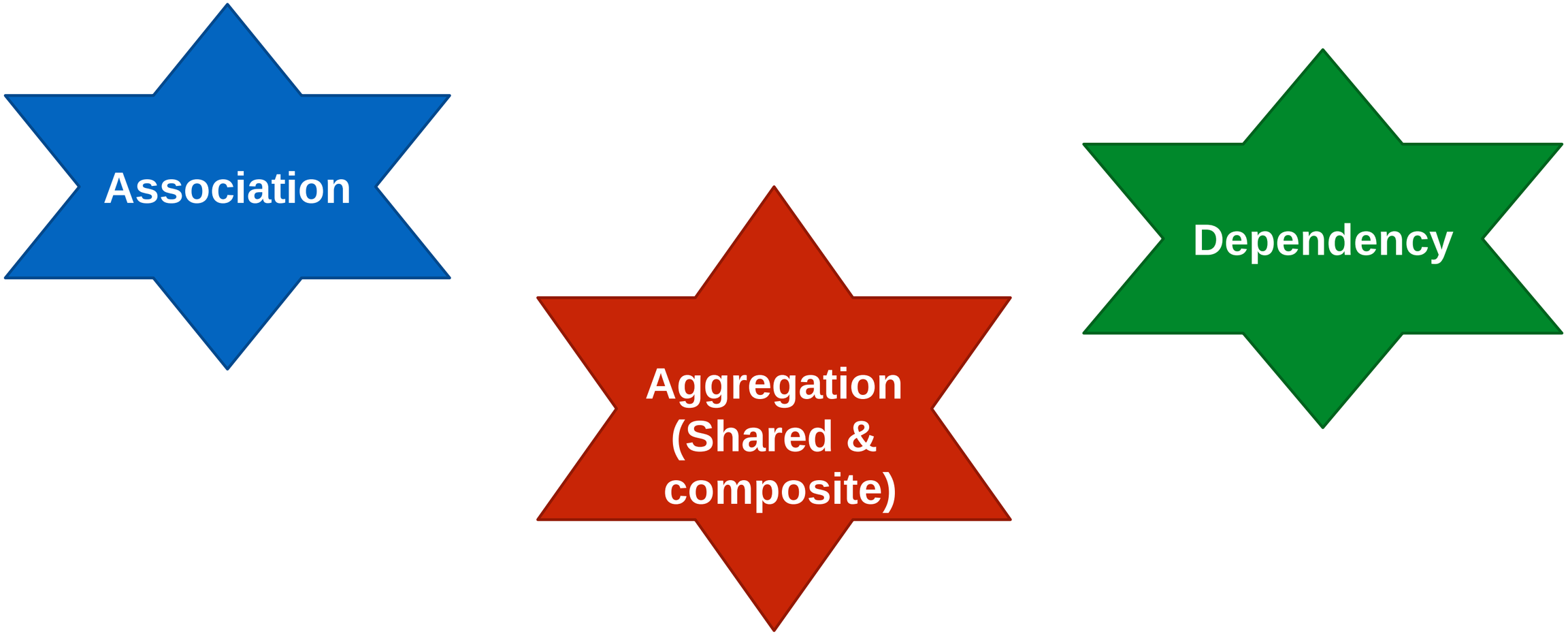
- In this unit, we are focusing only on interaction diagram & class diagram

- Describes the structure of a system by showing the system's classes their attributes, operations (or methods), and the relationships among objects

Class diagram notations

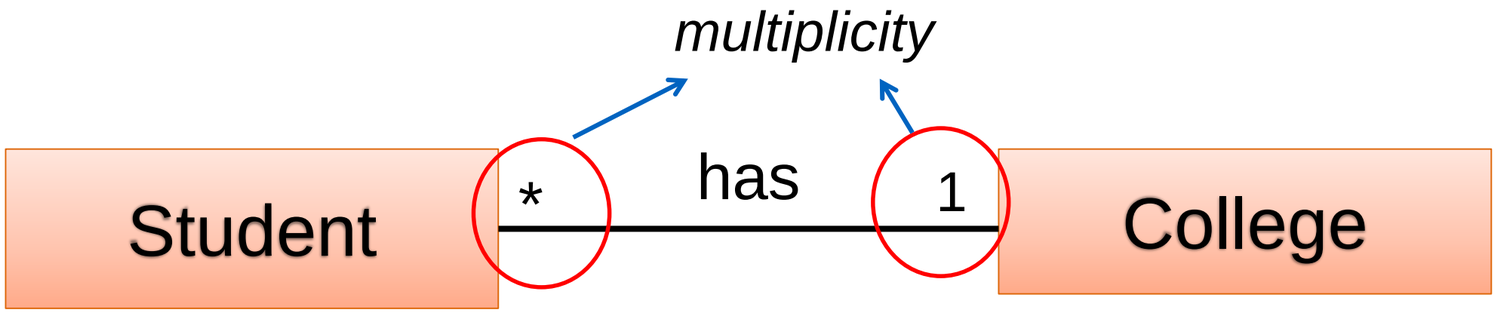


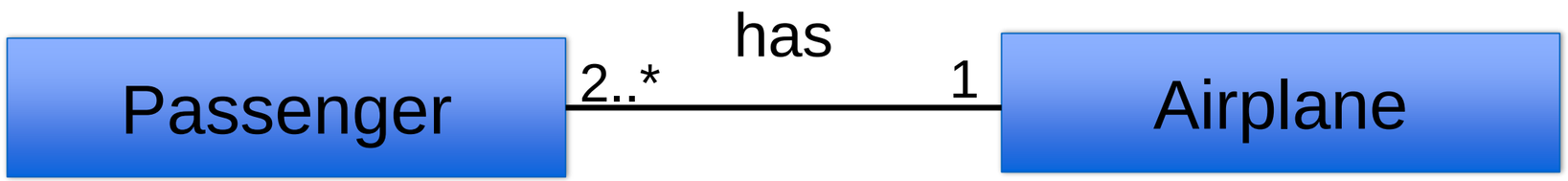
Three main kinds of relationships are used in this unit



Association

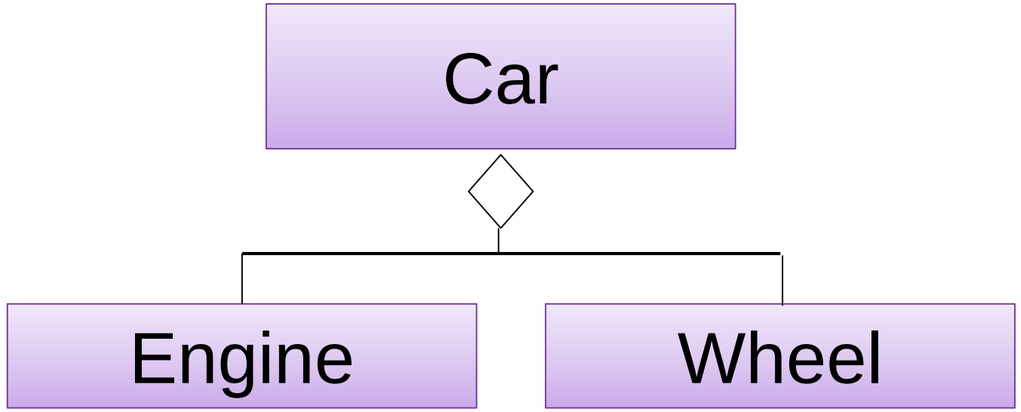
When two classes are connected to each other in any way, an association relation is established

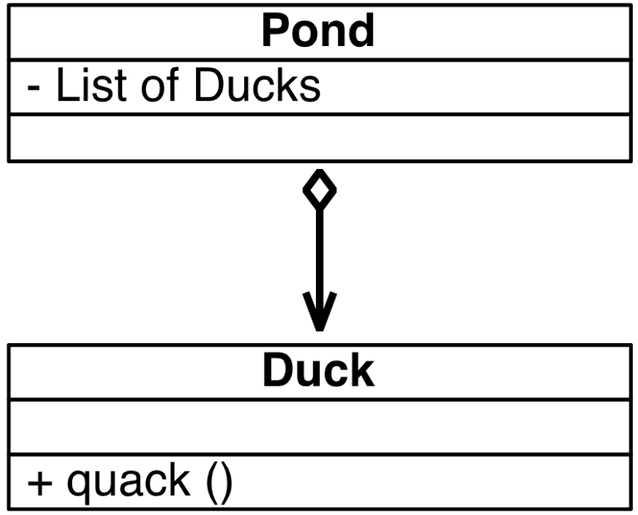




Shared aggregation

- Represents “is part of” relationship
- More specific than association relationship
- Consists of aggregated or built as a collection of classes
- **Weak dependency** - The child class is not automatically destroyed when the parent class is destroyed





Shared aggregation in C#


```
public class Address
{
. . .
}

public class Employee
{
    private Address address;

    public Employee(Address address)
    {

        this.address = address;

    }
    …
}
```

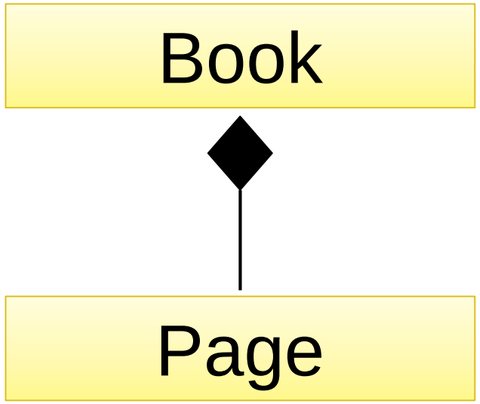
Person would then be used as follows:

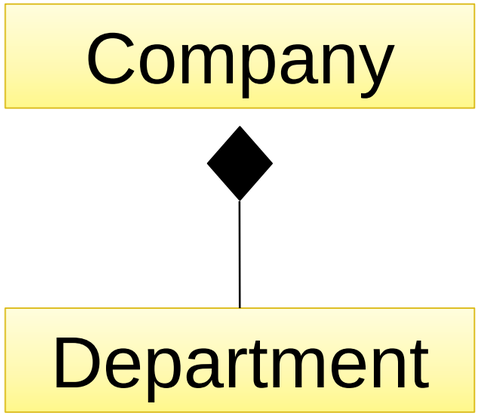
```
Address address = new Address();
Employee emp = new Employee(address);
```

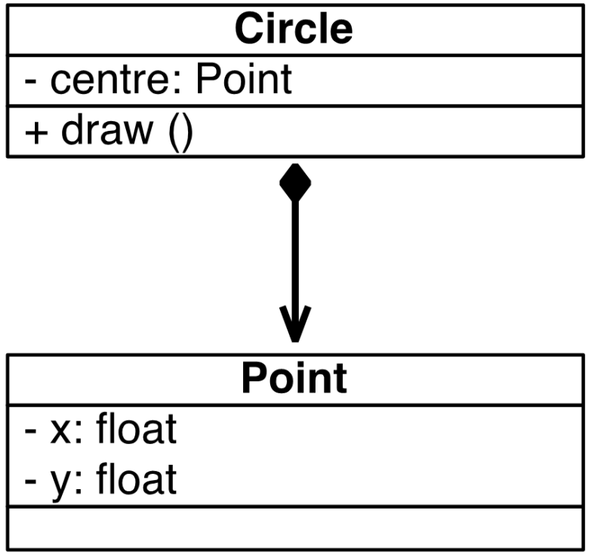
If Employee is destroyed, the Address still exists.

Composition

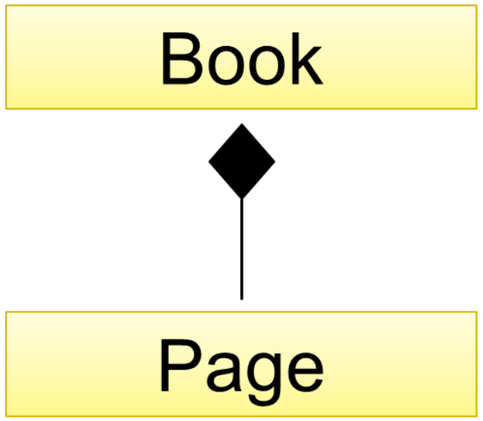
- Described as “ is entirely made of ”
- Represents a **strong dependency** - child class's instance lifecycle is dependent on the parent class's instance lifecycle - the child class live and die with its parent







Composition in C#

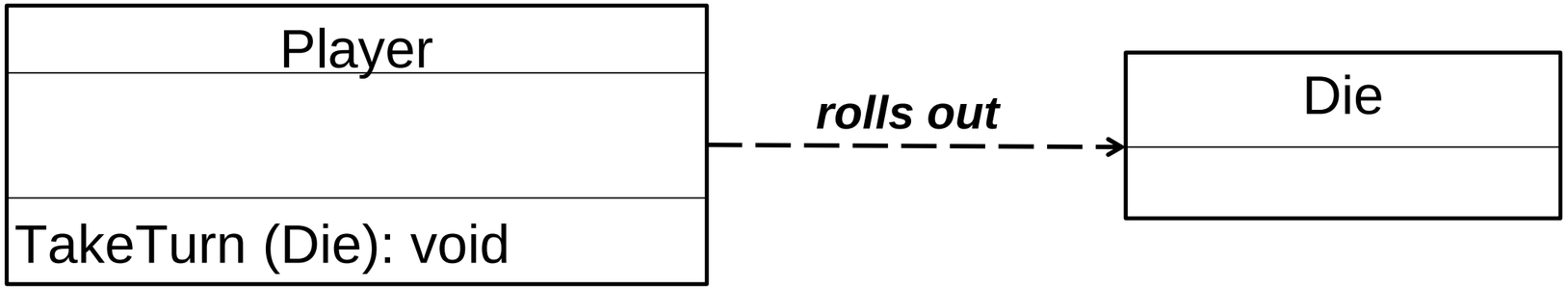



```
public class Page
{
 . . .
}

public class Book
{
    Page pg= new Page();
    .......
}
```

When Book is destroyed, Page is destroyed as well

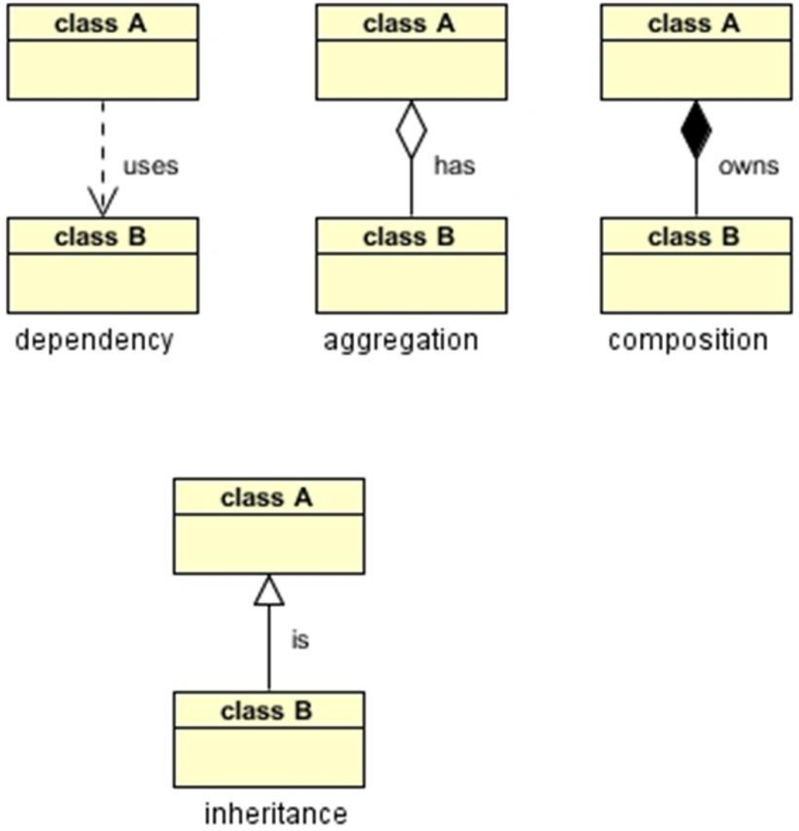
Dependency

- Weakest form of relationship
- Forms temporary relationship
- An object of one class might use an object of another class
  - Example: An object could be passed in as a parameter



- The Player class has a *TakeTurn* method which uses the Die object.
- The Player class pass in Die as a parameter into the method that will ask the die to roll.

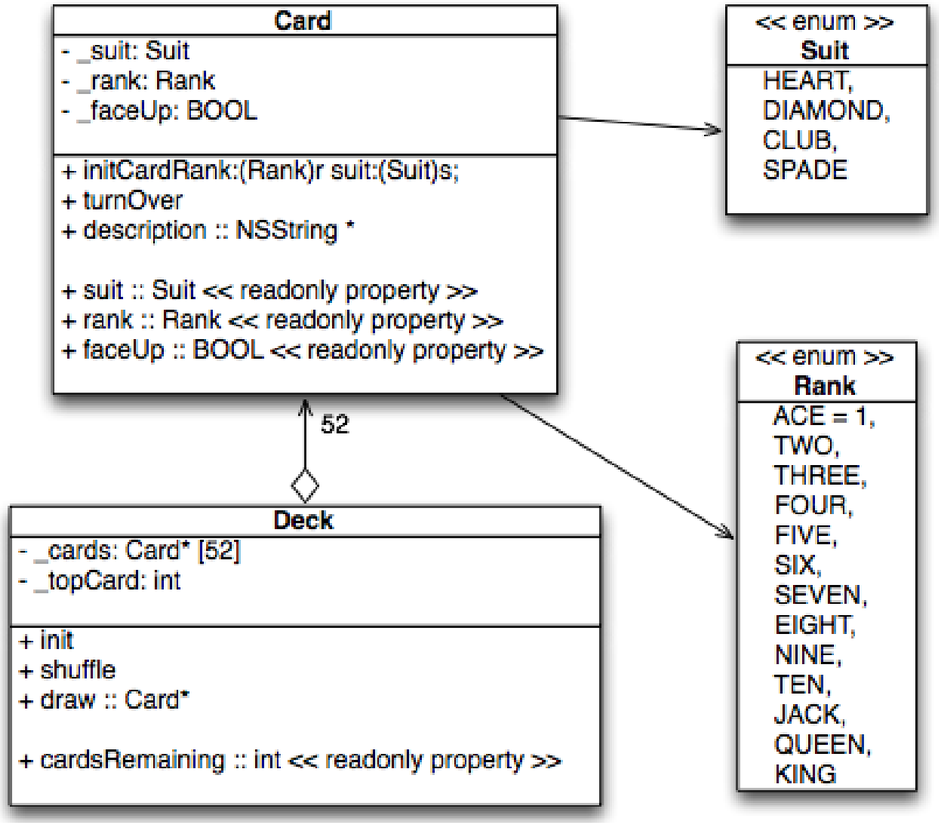
Summary of relationships….



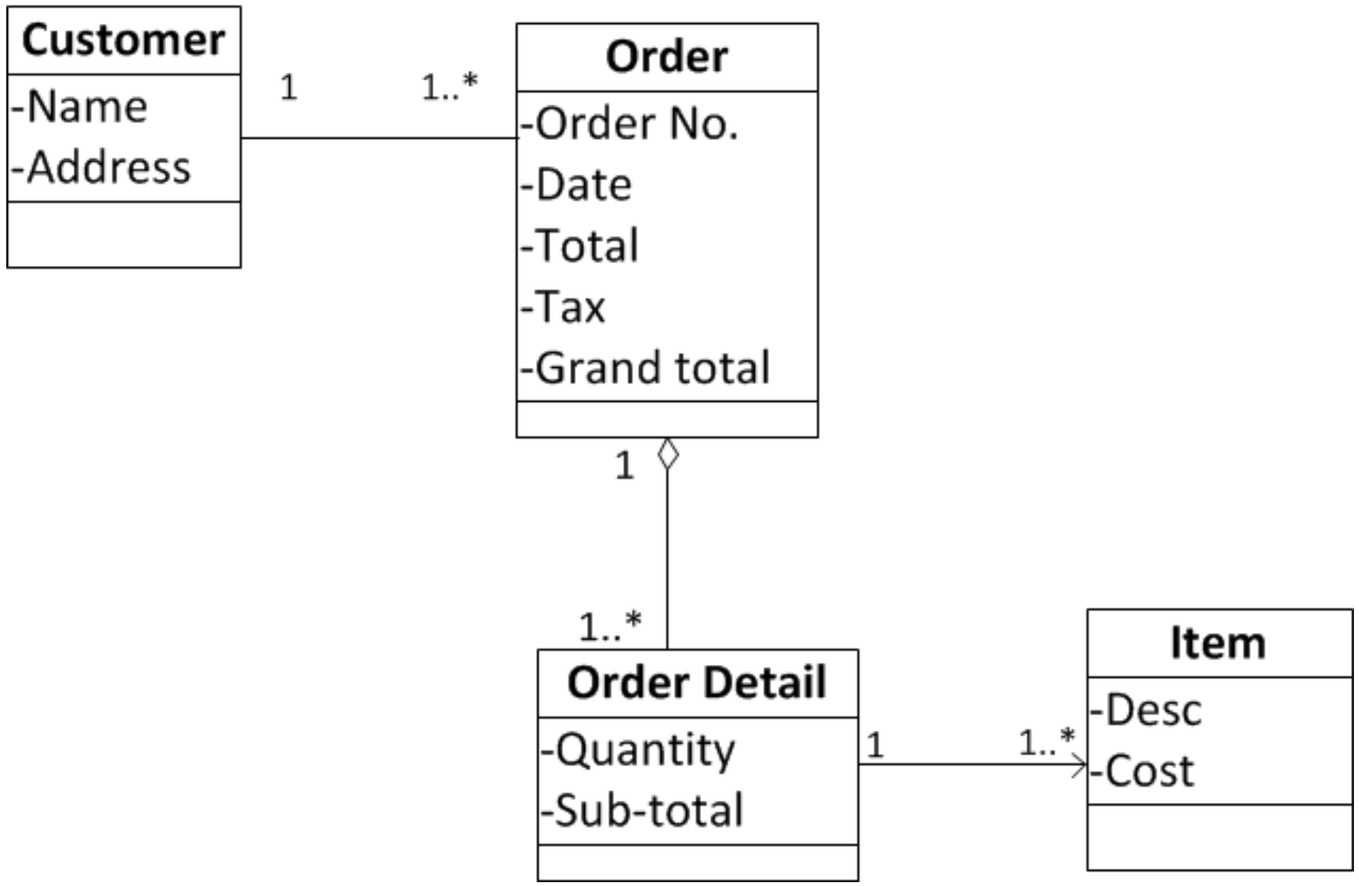
The relationships shown are read as follows:

- Dependency : class A uses class B
- Aggregation : class A has a class B
- Composition : class A owns a class B
- Inheritance : class B is a Class A (or class A is extended by class B)

Communicate the static structure of your program using a Class diagram



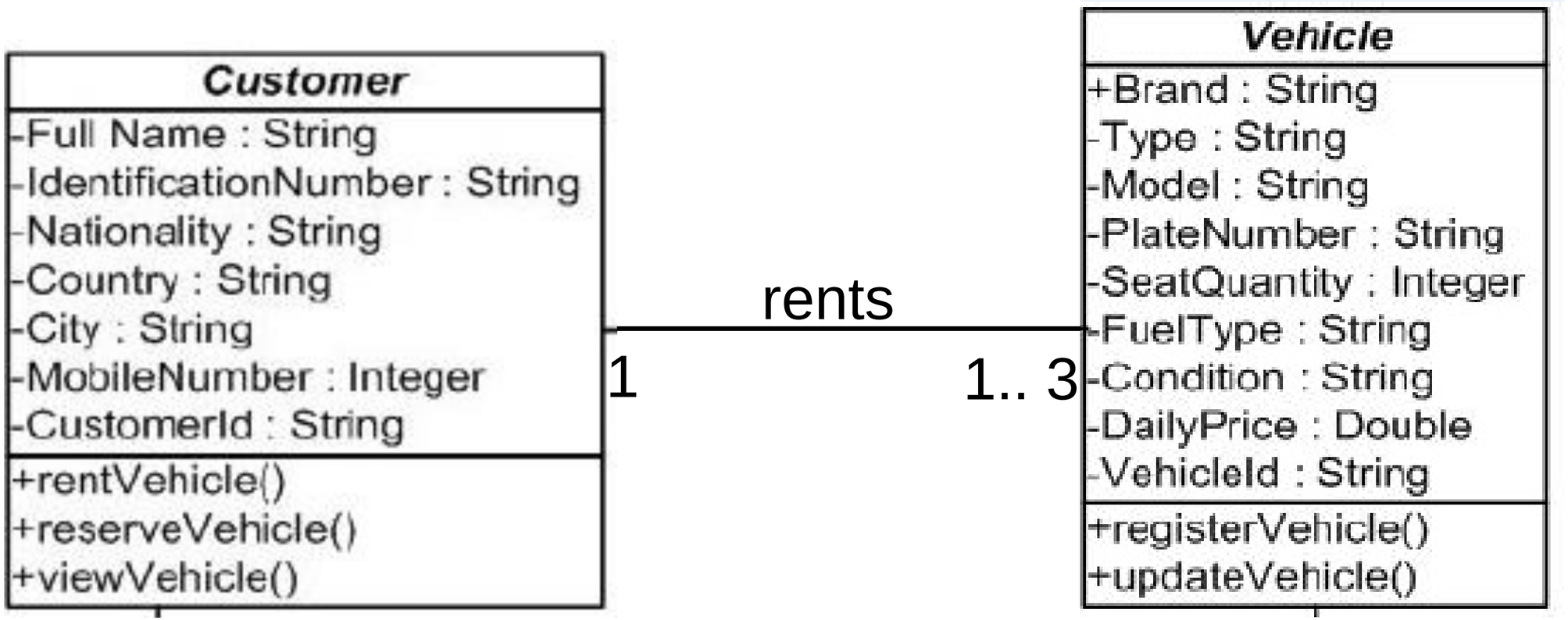
Example of a simple class diagram



Activity 1

Draw a simple class diagram for a car rental system. The customer may rent a maximum of 3 cars at a time. Each car will have different rental cost. Identify the classes needed, specify the relationship and the attributes…

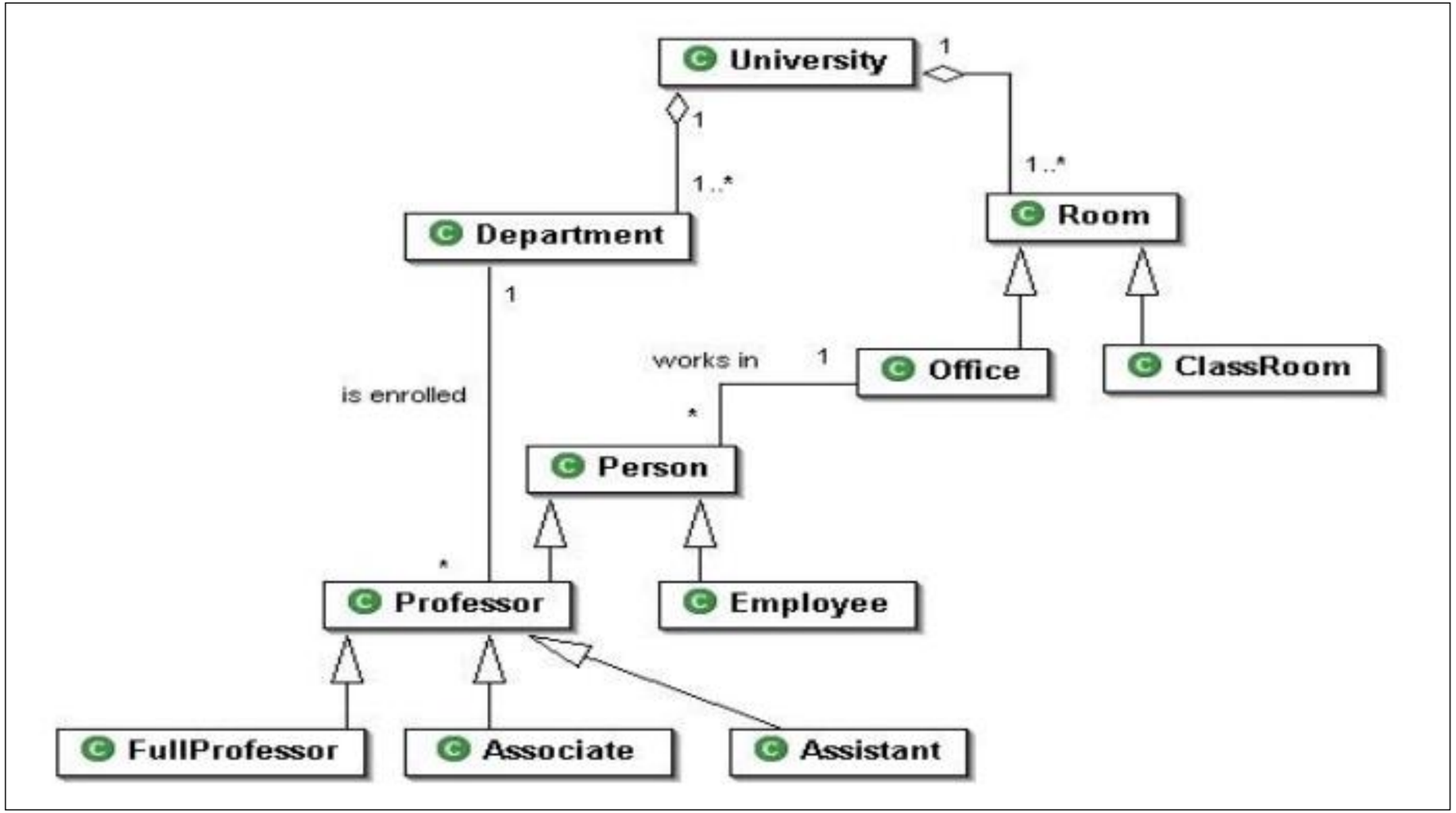
Sample Solution



Let’s have a 2nd try on class diagram!

In a university there are different classrooms, offices and departments. A department has a name and it contains many offices. A person working at the university has a unique ID and can be a professor or an employee. A professor can be a full, associate or assistant professor and he/she is enrolled in one department. Offices and classrooms have a number ID, and a classroom has a number of seats. Every employee works in an office.

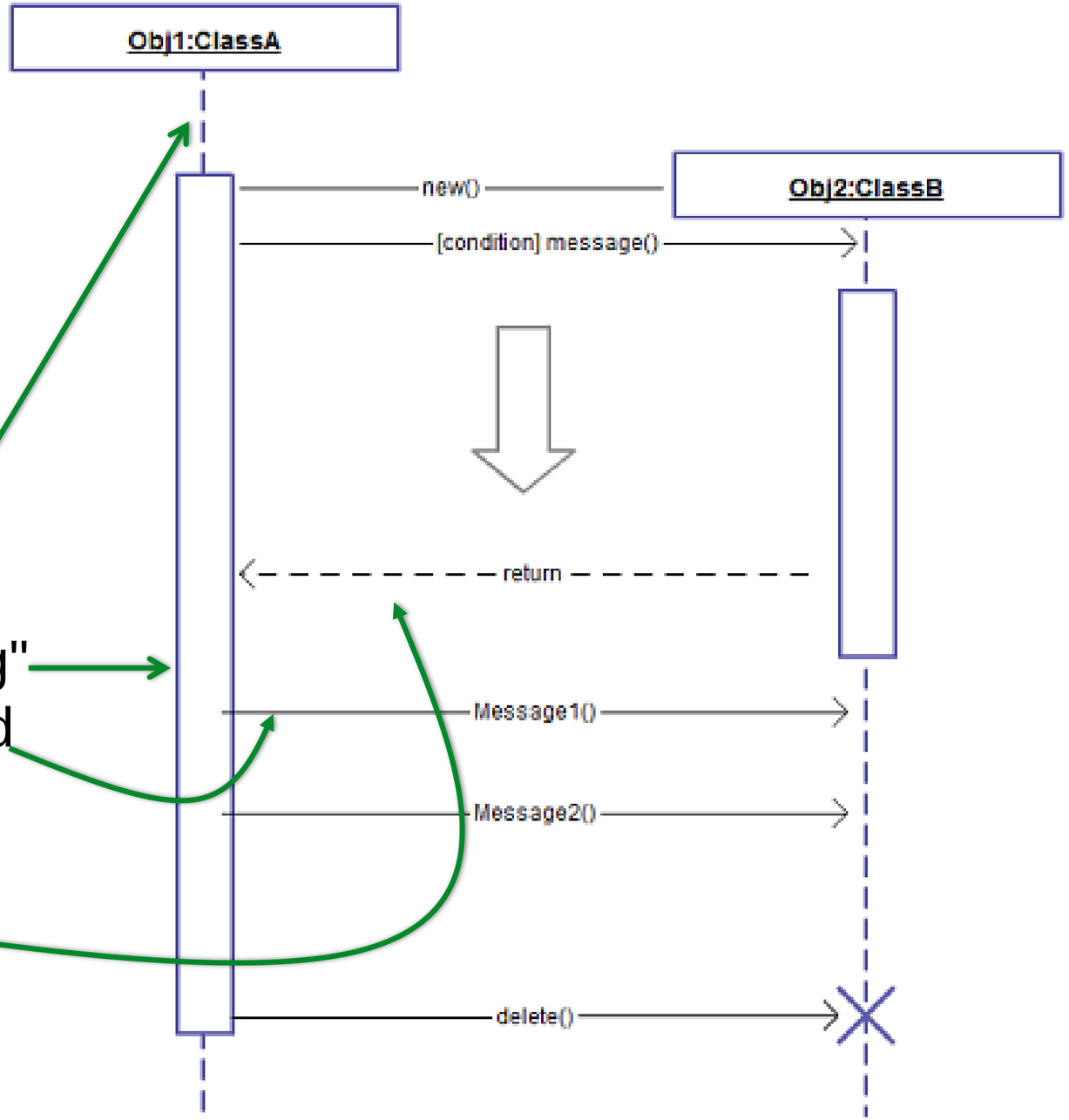
Sample solution



Resource: http://softeng.polito.it/tongji/SE/ex/exercises-w2.pdf

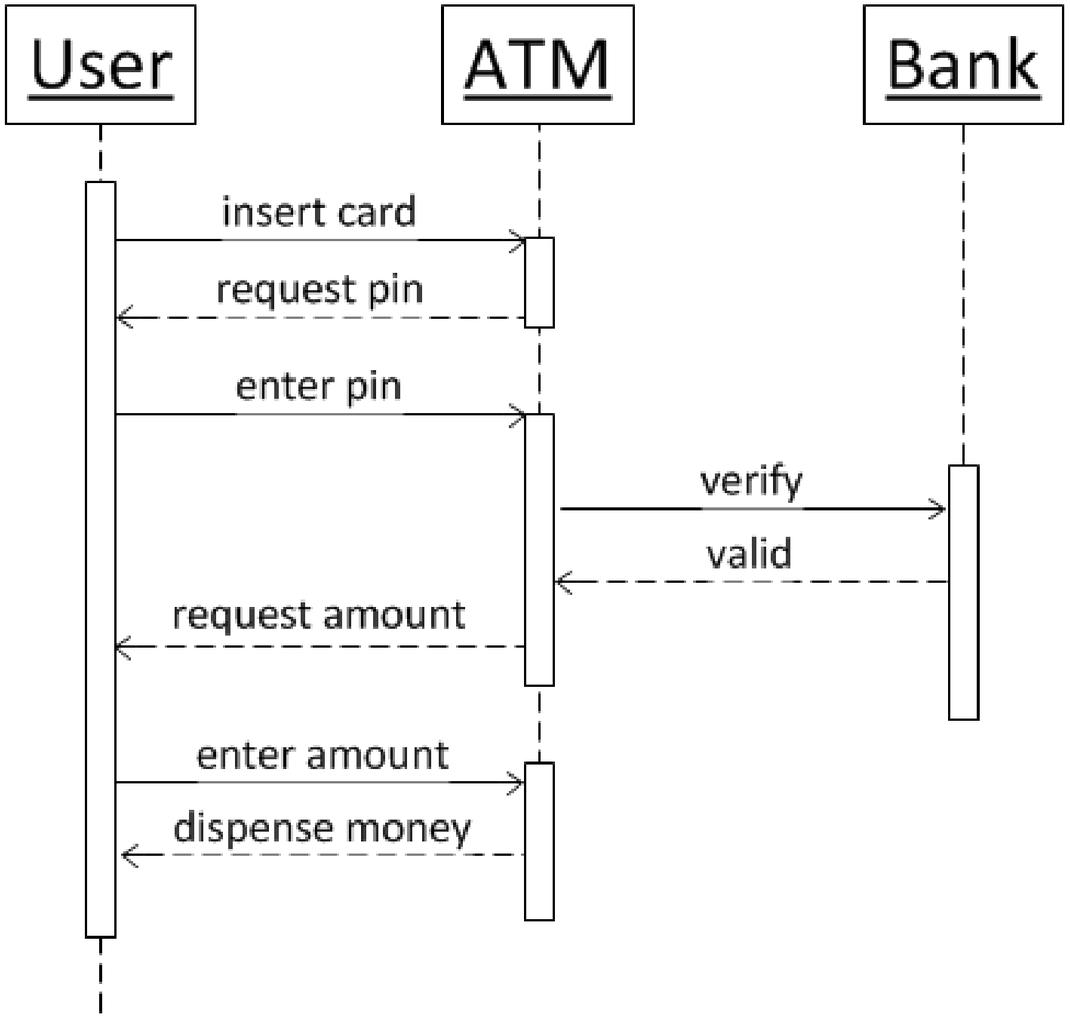
Communicate interactions using Sequence diagram

- life lines show where objects live
- activation boxes show objects "doing"
- Calls are arrows – show data passed
- Dashed arrows show return values

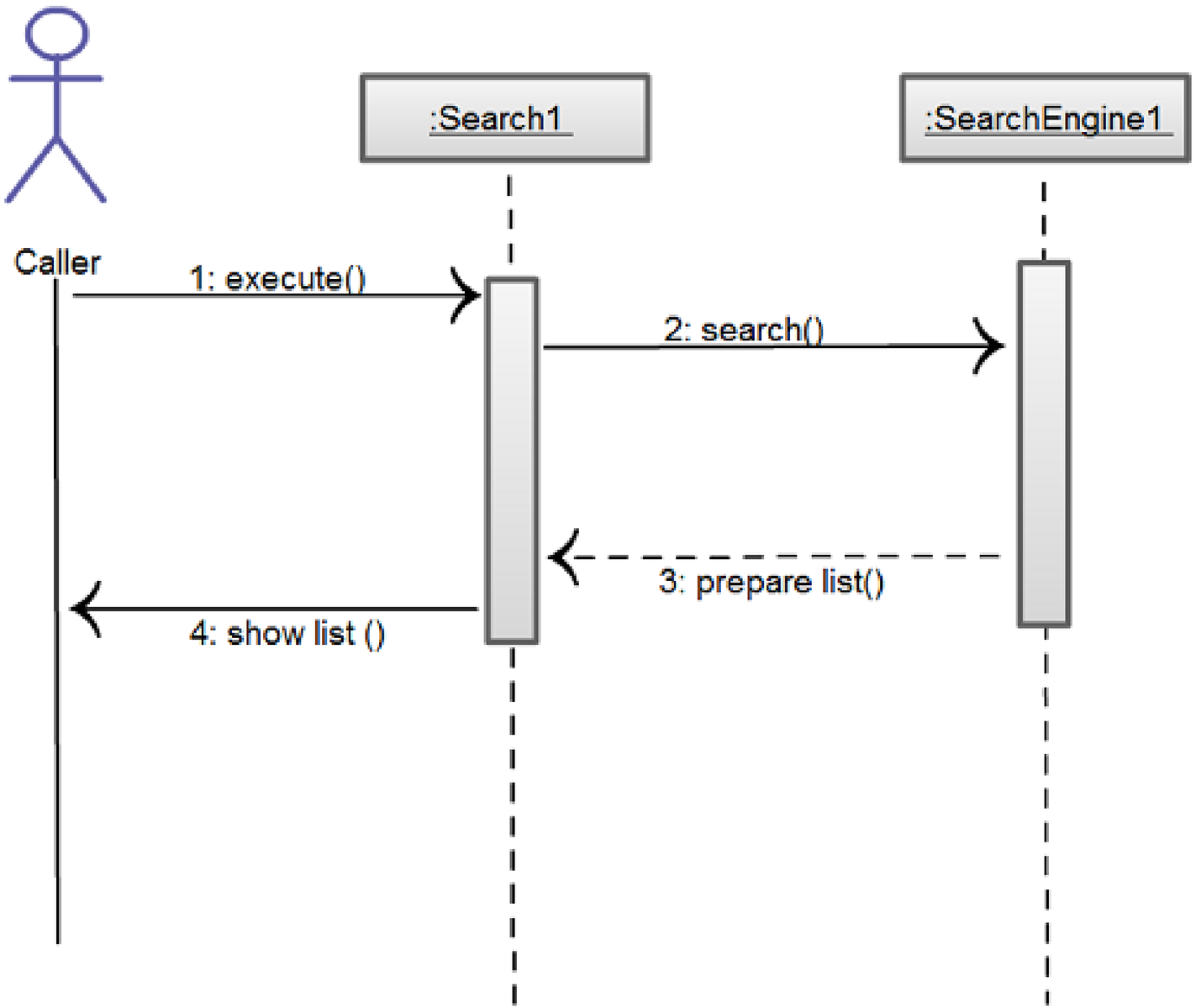


Example of Sequence Diagram for newbies

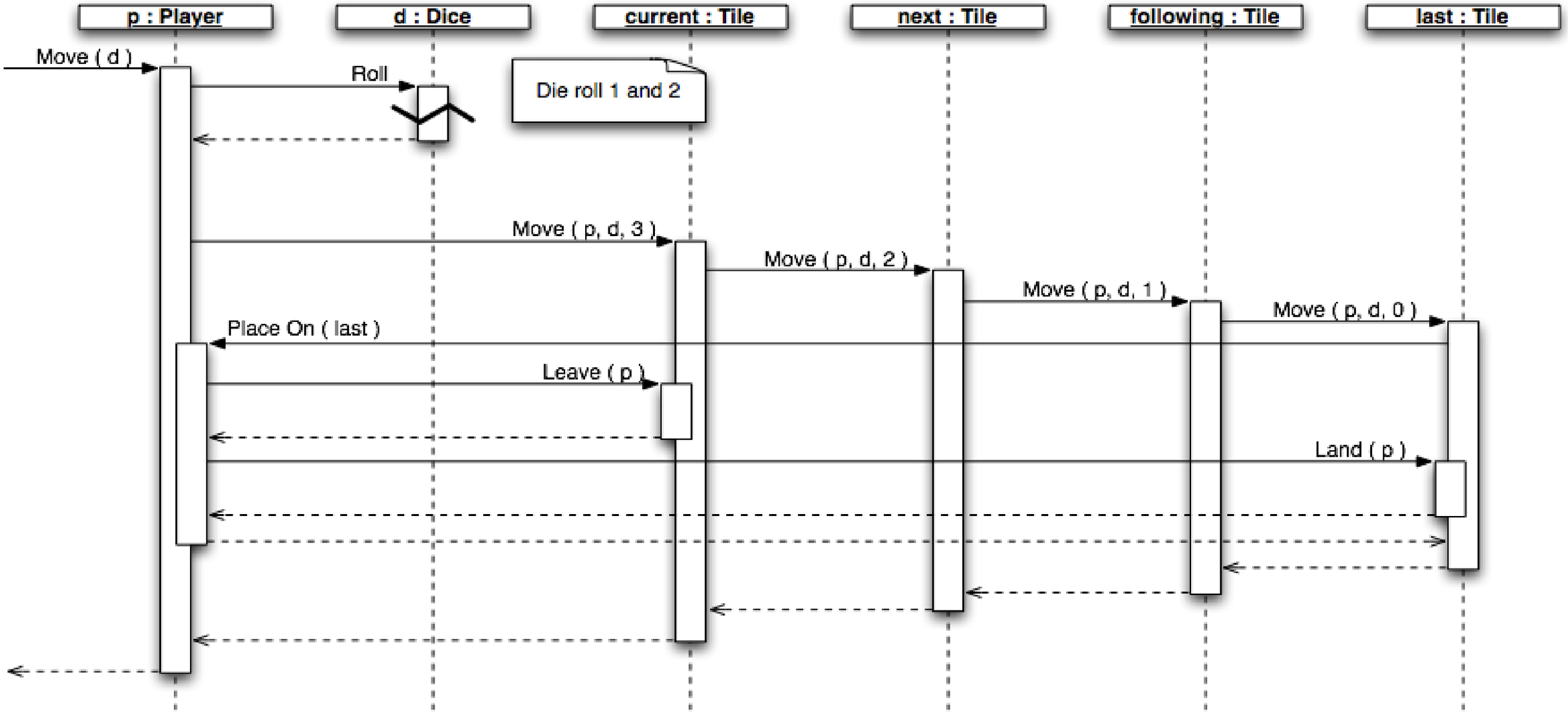
ATM withdrawal sequence diagram for newbies



Example of sequence diagram (1)



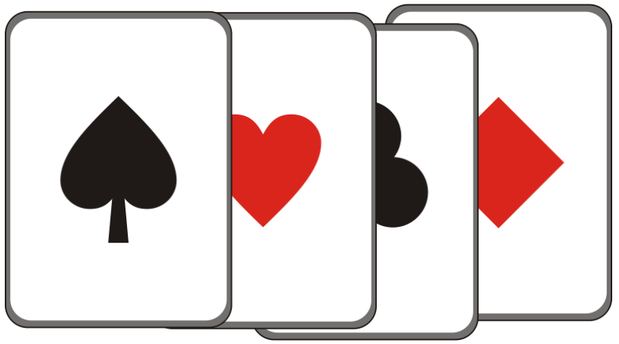
Example of sequence diagram (2)



C# collections

- specialized classes for data storage (string, integer,…, objects) and retrieval
- enhancement to arrays
- lists are dynamic, arrays have fixed size (a List is implicitly resizable)
- provides methods such as add, insert, remove, search
- To use it, you need to add `System.Collections.Generic` namespace

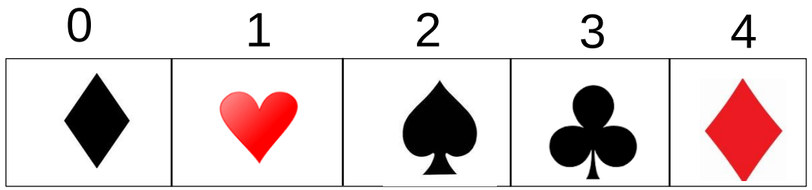
Example of List



objects

store

List of suits



Using list to store primitive data type

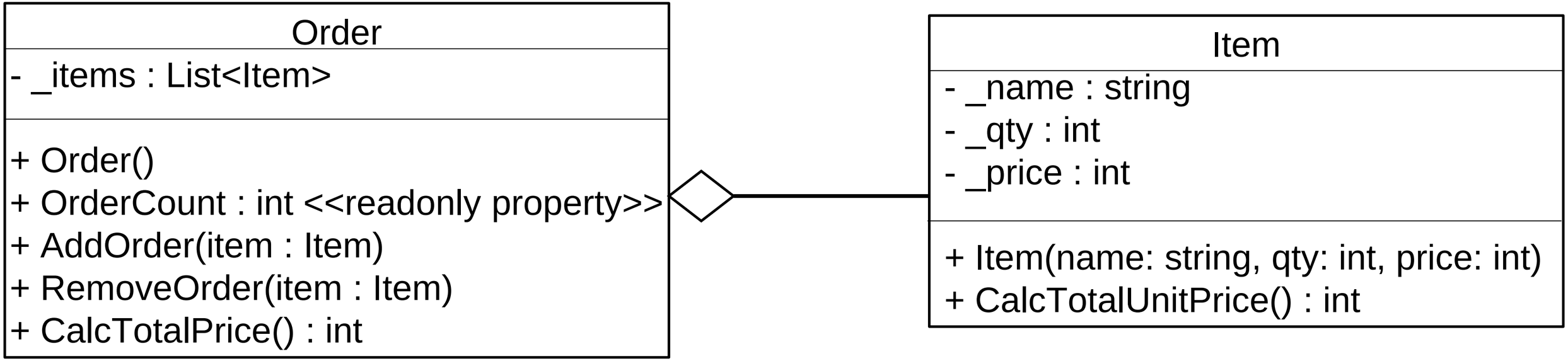
```
//creating the list
List<string> language = new List<string>();

//add into the list
language.Add("Java");
language.Add("C#");
language.Add("C");

//remove from the list
language.Remove("C#");
```

Using list to store objects

```
public class Customer
{
    private int ID;
    private string Name;

    //constructor for Customer
    public Customer(int id, string name)
    {
        ID = id;
        Name = name;
    }
}
```

```
class MainClass
{
    public static void Main(string[] args)
    {
        //creating Customer objects passing in ID, name
        Customer cust1=new Customer(1,"John Doe");
        Customer cust2= new Customer (2, "Mary Jane");

        //creating List to store customer objects
        List <Customer> clist = new List<Customer>();

        clist.Add(cust1);
        clist.Add(cust2);
    }
}
```

Let’s try out a simple example!



- ***Create the classes given and develop the unit tests to test their functionality***
- ***Implement a main program to run the classes implemented***

Next: C# Indexer

An indexer is a special kind of property that allows the caller to access your object using an index like an array.

```
Access_modifier return_type this[int index]{
    get{ // return the value specified by index }
    set{ // set the value specified by index }
}
```

Let's have a look on the implementation of indexer on the Order example!

This Week’s Tasks

Pass Task 9 - Shape Drawer

** Pass Task 10 - The Bank Customers (Assessed Task)

** compulsory tasks