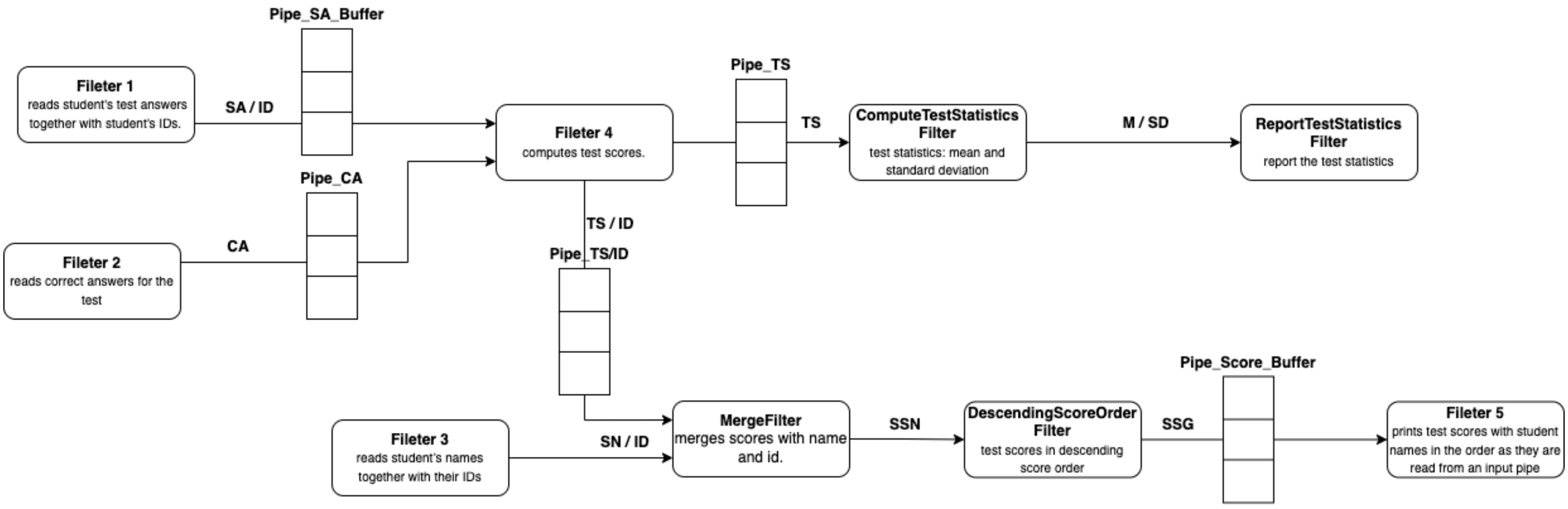
Total : 100/100

PROBLEM #1

Part A


Pipe_SA_Buffer
Fileter 1
reads student's test answers together with student's IDs.
SA / ID
Fileter 4
computes test scores.
Pipe_TS
TS
ComputeTestStatistics Filter
test statistics: mean and standard deviation
M / SD
ReportTestStatistics Filter
report the test statistics
Pipe_CA
CA
Fileter 2
reads correct answers for the test
TS / ID
Pipe_TS/ID
Pipe_Score_Buffer
Fileter 3
reads student's names together with their IDs
SN / ID
MergeFilter
merges scores with name and id.
SSN
DescendingScoreOrder Filter
test scores in descending score order
SSG
Fileter 5
prints test scores with student names in the order as they are read from an input pipe

Pipes:

SA : student's test answers together with student's IDs

CA : correct answers for the test

TS : student test scores

SN : student’s names together with their IDs

SSN : student names, ids and test scores

SSG : student names ,ids and test scores sorted in a descending grade order

M : mean

SD : standard deviation

Filters:

ComputeTestStatisticsFilter : this filter computes test statistics: mean and standard deviation

ReportTestStatisticsFilter : this filter reports test statistics.

MergeFilter : this filter merges scores with name and id.

DescendingScoreOrderFilte: this filter descending score order with student names and their IDs.

Filter1 : this filter reads student's test answers together with student's IDs.

Filter2 : this filter reads correct answers for the test.

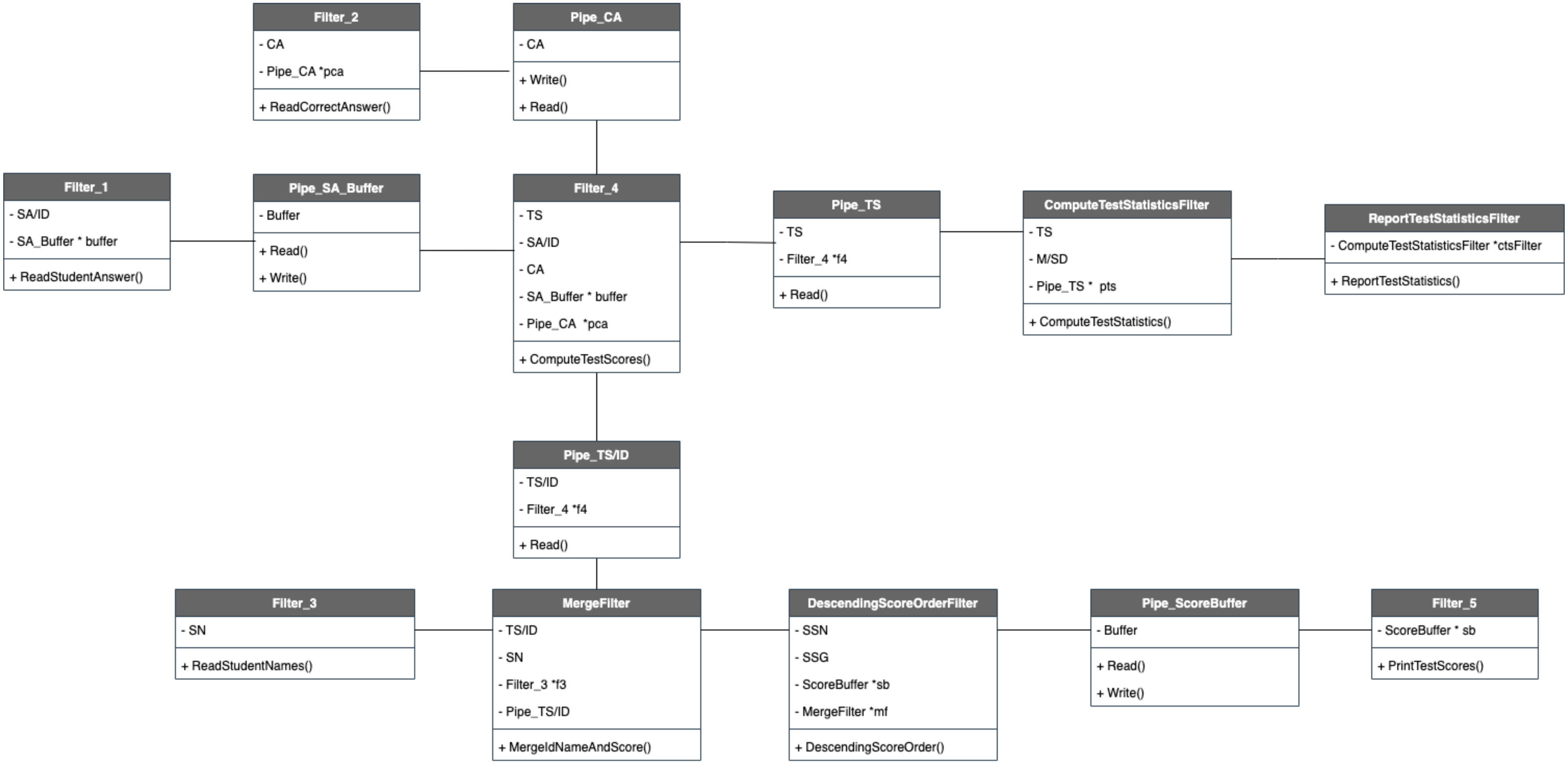
Filter3 : this filter reads student’s names together with their IDs.

Filter4 : this filter computes test scores.

Filter5 : this filter prints test scores with student names in the order as they are read from an input pipe.

Part B

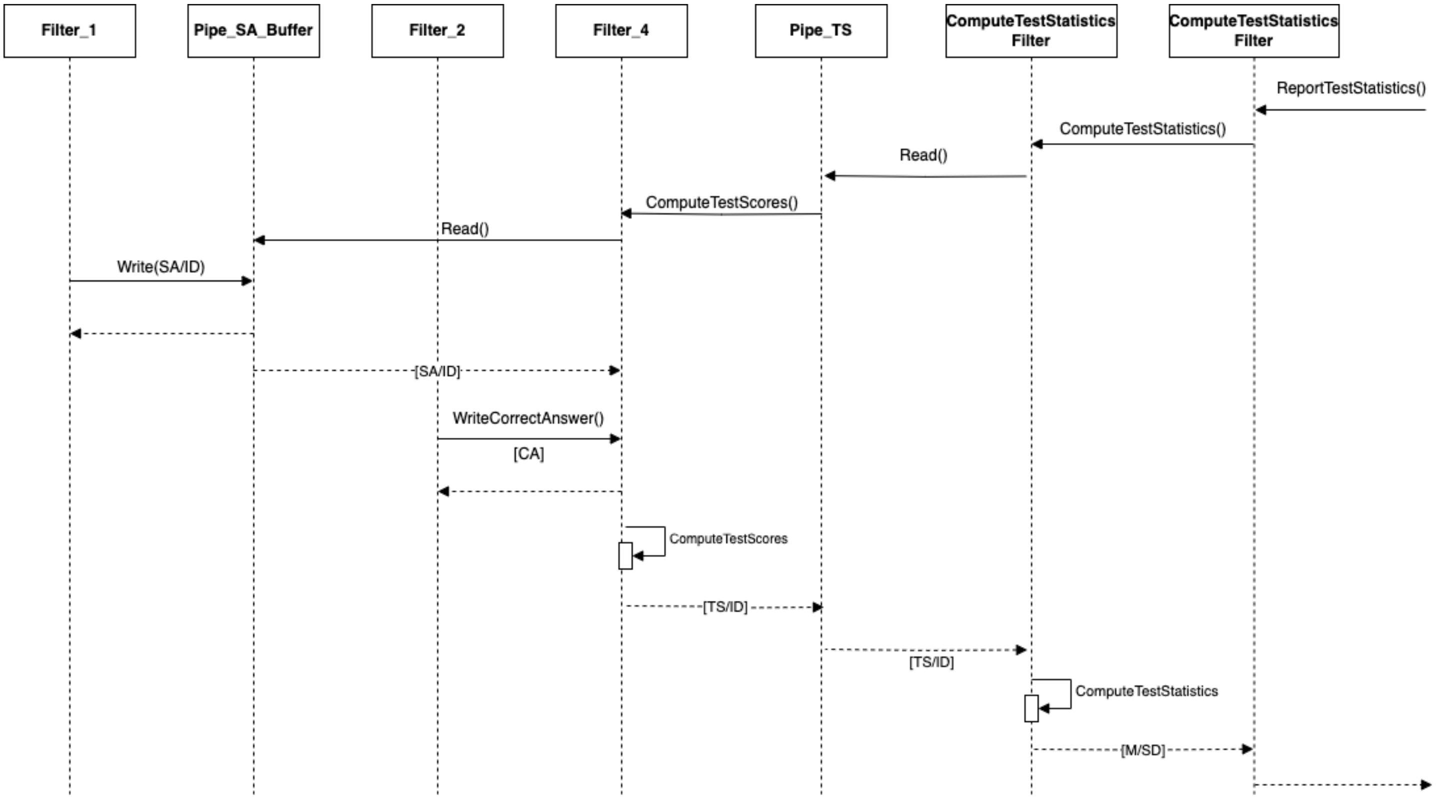
UML


Filter_2
- CA
- Pipe_CA *pca
+ ReadCorrectAnswer()
Pipe_CA
- CA
+ Write()
+ Read()
Filter_1
- SA/ID
- SA_Buffer * buffer
+ ReadStudentAnswer()
Pipe_SA_Buffer
- Buffer
+ Read()
+ Write()
Filter_4
- TS
- SA/ID
- CA
- SA_Buffer * buffer
- Pipe_CA *pca
+ ComputeTestScores()
Pipe_TS
- TS
- Filter_4 *f4
+ Read()
ComputeTestStatisticsFilter
- TS
- M/SD
- Pipe_TS * pts
+ ComputeTestStatistics()
ReportTestStatisticsFilter
- ComputeTestStatisticsFilter *ctsFilter
+ ReportTestStatistics()
Pipe_TS/ID
- TS/ID
- Filter_4 *f4
+ Read()
Filter_3
- SN
+ ReadStudentNames()
MergeFilter
- TS/ID
- SN
- Filter_3 *f3
- Pipe_TS/ID
+ MergeIdNameAndScore()
DescendingScoreOrderFilter
- SSN
- SSG
- ScoreBuffer *sb
- MergeFilter *mf
+ DescendingScoreOrder()
Pipe_ScoreBuffer
- Buffer
+ Read()
+ Write()
Filter_5
- ScoreBuffer * sb
+ PrintTestScores()

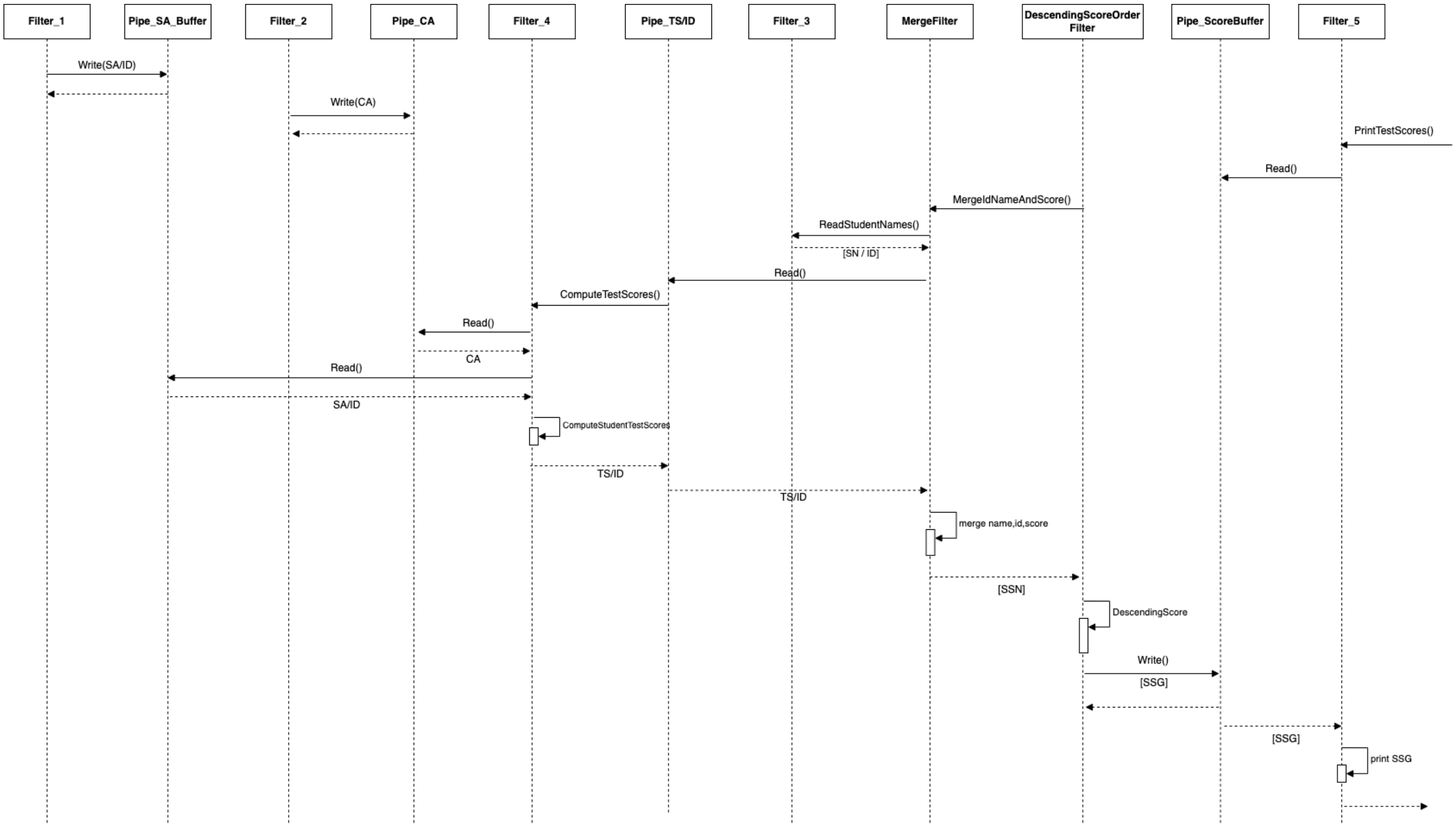
pseudo-code

```
class Filter_1 {
    Pipe_SA_Buffer buffer;
    ReadStudentAnswer(){
        while(true) {
        // Read student test answers in to SA.
        SA/ID = read student's test answers together with student's IDs.
        buffer.write(SA/ID)
        }
    }
}

class Pipe_SA_Buffer {
    buffer
    // SA means student's test answers.
    SA/ID Read() {
        if(buffer.isEmpty()) {
            wiat()//Waiting for data(SA) to be written to the buffer.
            return buffer.pop() //Data(SA) writing completed, end waiting, pop 
data(SA)
        }else {
            //The buffer is not empty, pop data(SA)
            return buffer.pop()
        }
    }
    Write(SA/ID) {
        buffer.push(SA)
    }
}

class Filter_2 {
    Pipe_CA pca;
    // CA means correct answers
    ReadCorrectAnswer() {
        //reads correct answers for the test in to CA.
        CA = reads correct answers
        pca.Write(CA)
    }

}

class Filter_3 {
    // SN means student's names
    SN/ID ReadStudentNames() {
        SN/ID= reads student's names together with their ids
        return SN/ID
    }
}

class Filter_4 {
    SA_Buffer buffer;
    Pipe_CA pca;

    TS/ID ComputeTestScores() {
       SA/ID = buffer.read()
       if(SA/ID == null) {
            wait(); //Waiting for SA/ID to be written
        }

       CA = pca.read()
       if(CA == null) {
            wait(); //Waiting for CA to be written
        }

        TS/ID = calculate student test score . using SA and CA
        return TS/ID

    }
}

class MergeFilter {
    Filter_3 f3;
    Pipe_TS/ID pts;

    SSN MergeIdNameAndScore() {
        SN/ID = f3.ReadStudentNames()
        TS/ID = pts.read()
        SSN = merge SN,TS,ID
        return SSN
    }
}

class DescendingScoreOrderFilter {
    ScoreBuffer sb;
    MergeFilter mf;
    // SSG means descending score order
    DescendingScoreOrder() {
        while(true) {
            SSN = mf.MergeIdNameAndScore()
            SSG = test scores in descending score order with SSN;
            // write SSG to ScoreBuffer
            sb.write(SSG)
        }
    }
}

class Pipe_ScoreBuffer {
    buffer

    DS Read() {
        if(buffer.isEmpty()) {
            wiat()//waiting for data(SSG) to be written to the buffer.
            return buffer.pop() //data(SSG) writing completed, end waiting, pop
data(SSG)
        }else {
            //The buffer is not empty, pop data(SSG)
            return buffer.pop()
        }
    }

    Write(SSG) {
        buffer.push(SSG)
    }
}

class Filter_5 {
    Pipe_ScoreBuffer sb;

    PrintTestScores() {
        while(true) {
            SSG = sb.read()
            prints(SSG) //prints test scores with student names in the order
        }
    }
}

class ComputeTestStatisticsFilter {
    Pipe_TS pts;
    // M/SD means mean and standard deviation
    M/SD ComputeTestStatistics() {
        TS = f4.pts
        M/SD = compute test statistics: mean and standard deviation
        return M/SD
    }
}

class ReportTestStatisticsFilter {
    ComputeTestStatisticsFilter ctsf;

    ReportTestStatistics() {
        M/SD = ctsf.ComputeTestStatistics()
        report(M/SD) //Report test statistics
    }
}


class Pipe_CA{
    CA

    Write(CA) {
        this.CA = CA
    }

    Read() {
        if(CA == null) {
           wait()//Waiting for CA to be written
        }
    }

}

class Pipe_TS{
    Filter_4 f4;

    Read() {
        f4.read()
    }
}

class Pipe_TS/ID{
   Filter_4 f4;

    Read() {
        f4.read()
    }

}
```

sequence diagram(Report test statistics)



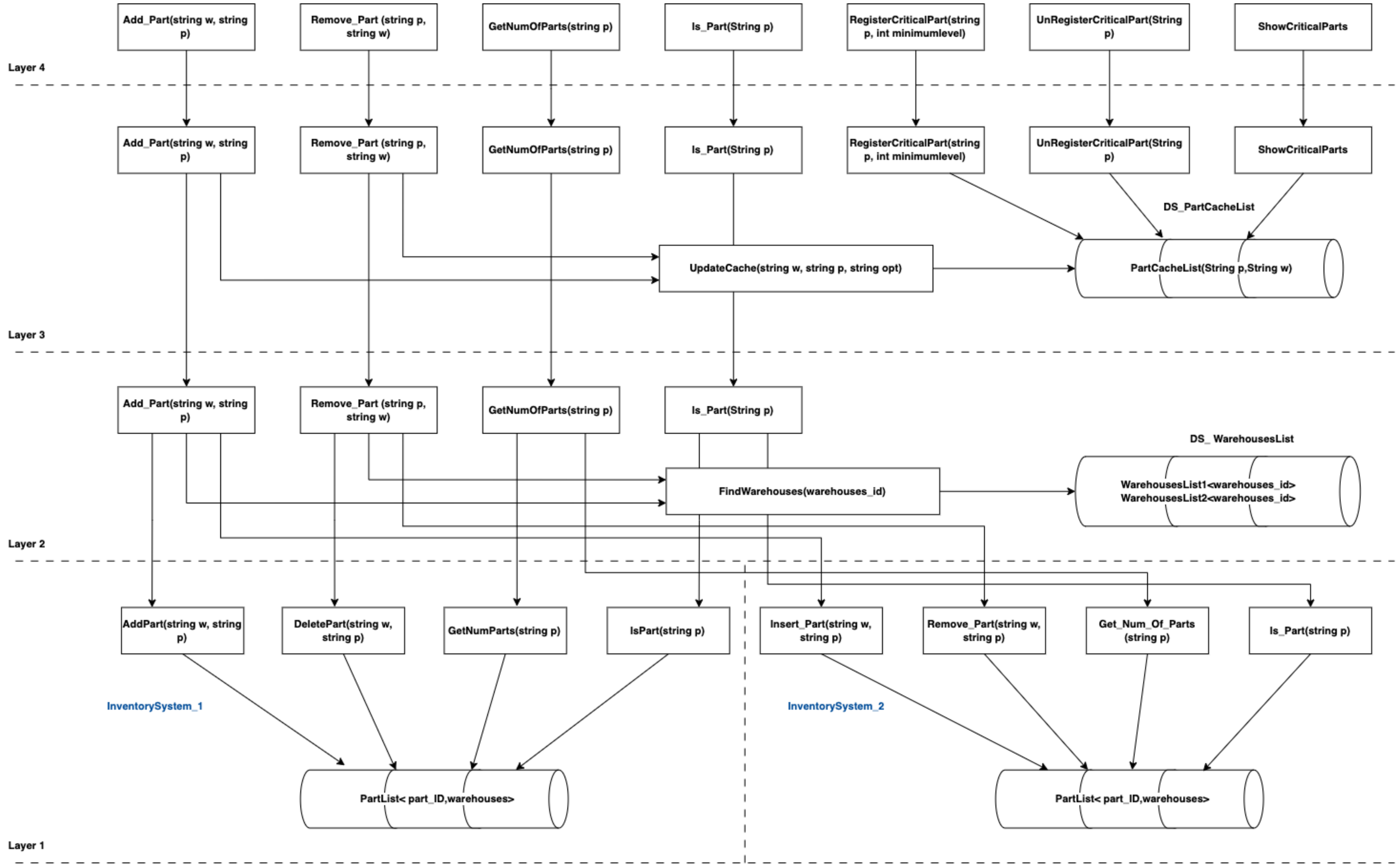
sequence diagram(Report test scores)



Problem#2 : 30/30

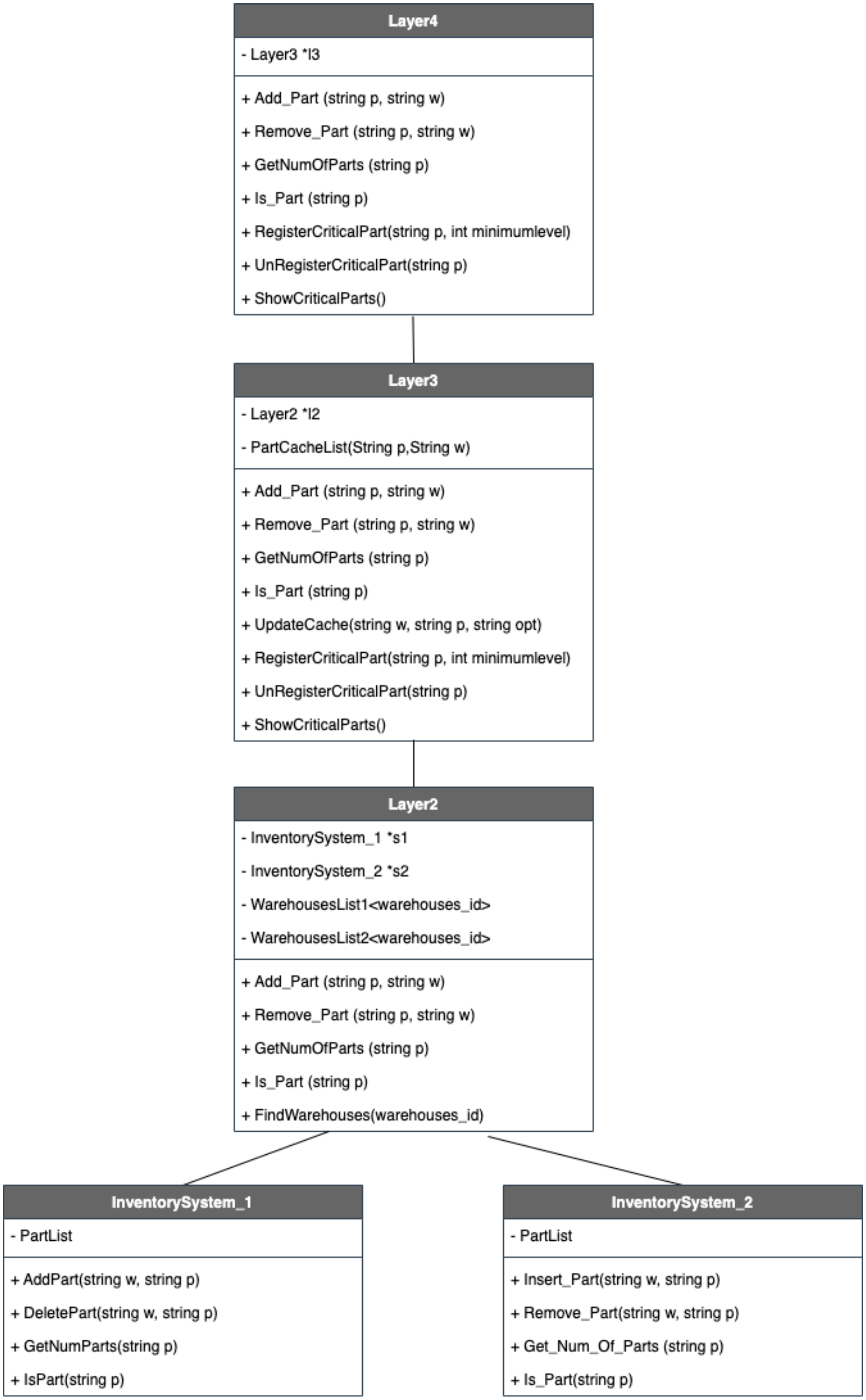
PROBLEM #2

Layer diagram



Layer 4
Add_Part(string w, string p)
Remove_Part (string p, string w)
GetNumOfParts(string p)
Is_Part(String p)
RegisterCriticalPart(string p, int minimumlevel)
UnRegisterCriticalPart(String p)
ShowCriticalParts
Layer 3
Add_Part(string w, string p)
Remove_Part (string p, string w)
GetNumOfParts(string p)
Is_Part(String p)
RegisterCriticalPart(string p, int minimumlevel)
UnRegisterCriticalPart(String p)
ShowCriticalParts
DS_PartCacheList
UpdateCache(string w, string p, string opt)
PartCacheList(String p,String w)
Layer 2
Add_Part(string w, string p)
Remove_Part (string p, string w)
GetNumOfParts(string p)
Is_Part(String p)
DS_ WarehousesList
FindWarehouses(warehouses_id)
WarehousesList1<warehouses_id>
WarehousesList2<warehouses_id>
Layer 1
AddPart(string w, string p)
DeletePart(string w, string p)
GetNumParts(string p)
IsPart(string p)
Insert_Part(string w, string p)
Remove_Part(string w, string p)
Get_Num_Of_Parts (string p)
Is_Part(string p)
InventorySystem_1
InventorySystem_2
PartList< part_ID,warehouses>
PartList< part_ID,warehouses>

UML



Layer4

- Layer3 *l3

+ Add_Part (string p, string w)
+ Remove_Part (string p, string w)
+ GetNumOfParts (string p)
+ Is_Part (string p)
+ RegisterCriticalPart(string p, int minimumlevel)
+ UnRegisterCriticalPart(string p)
+ ShowCriticalParts()

Layer3

- Layer2 *l2
- PartCacheList(String p,String w)

+ Add_Part (string p, string w)
+ Remove_Part (string p, string w)
+ GetNumOfParts (string p)
+ Is_Part (string p)
+ UpdateCache(string w, string p, string opt)
+ RegisterCriticalPart(string p, int minimumlevel)
+ UnRegisterCriticalPart(string p)
+ ShowCriticalParts()

Layer2

- InventorySystem_1 *s1
- InventorySystem_2 *s2
- WarehousesList1<warehouses_id>
- WarehousesList2<warehouses_id>

+ Add_Part (string p, string w)
+ Remove_Part (string p, string w)
+ GetNumOfParts (string p)
+ Is_Part (string p)
+ FindWarehouses(warehouses_id)

InventorySystem_1

- PartList

+ AddPart(string w, string p)
+ DeletePart(string w, string p)
+ GetNumParts(string p)
+ IsPart(string p)

InventorySystem_2

- PartList

+ Insert_Part(string w, string p)
+ Remove_Part(string w, string p)
+ Get_Num_Of_Parts (string p)
+ Is_Part(string p)

pseudo-code

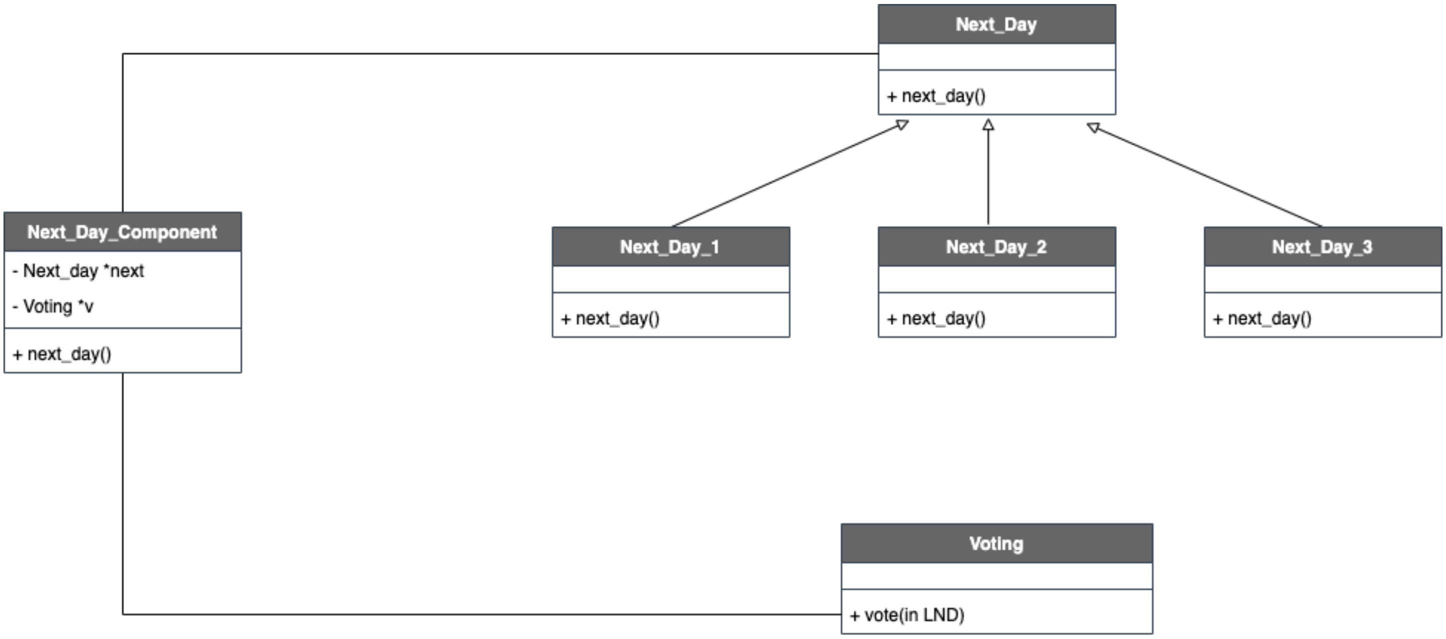
```
class Layer4 {
    Layer l3;

    void Add_Part (string p, string w) {
        l3.Add_Part(p,w)
    }

    void Remove_Part (string p, string w) {
        l3.Remove_Part(p,w)
    }

    int GetNumOfParts (string p) {
        return l3.GetNumOfParts(p)
    }

    int Is_Part (string p) {
        return l3.Is_Part(p)
    }

    void RegisterCriticalPart(string p, int minimumlevel) {
        l3.RegisterCriticalPart(p,minimumlevel)
    }

    void UnRegisterCriticalPart(string p) {
        l3.UnRegisterCriticalPart(p)
    }

    void ShowCriticalParts() {
        l3.ShowCriticalParts()
    }

}

class L3 {
    L2 l2;
    PartCacheList cache;

    void Add_Part (string p, string w) {
        l2.Add_Part(p,w)
        updateCache(p,w,"add")
    }

    void Remove_Part (string p, string w) {
        l3.Remove_Part(p,w)
        updateCache(p,w,"remove")
    }

    int GetNumOfParts (string p) {
        return l2.GetNumOfParts(p)
    }

    int Is_Part (string p) {
        return l2.Is_Part(p)
    }
```

```
   void RegisterCriticalPart(string p, int minimumlevel) {
        if(l2.isPart(p)) {
            insert into p and minimumlevel to PartCacheList
        }
    }

    void UnRegisterCriticalPart(string p) {
        remove p from PartCacheList
    }

    void ShowCriticalParts() {
        if(number of parts of a critical part < minimum) {
            display critical parts from cache.
        }
    }

    updateCache(string p, string w,string opt) {
        if(opt == 'add') {
            add part to PartCacheList
        } else if(opt == "remove") {
            remove part from PartCacheList
        }

        if(number of parts of a critical part < minimum) {
            add part to cache
        }
    }
}
class layer2 {
    WarehousesList1<warehouses_id>
    WarehousesList2<warehouses_id>
    InventorySystem_1 s1;
    InventorySystem_2 s2;

    void Add_Part (string p, string w) {
        if(FindWarehouses(w) == "InventorySystem_1") {
            s1.AddPart( w,  p)
        } else if( FindWarehouses(w) == "InventorySystem_2") {
            s2.Insert_Part( w,  p)
        }

    }

    void Remove_Part (string p, string w) {
        if(FindWarehouses(w) == "InventorySystem_1") {
            s1.DeletePart( w,  p)
        } else if( FindWarehouses(w) == "InventorySystem_2") {
            s2.Remove_Part( w,  p)
        }
    }

    int GetNumOfParts (string p) {
       //Return the sum of the parts counts from two warehouses, that is, the total
number of parts.
        return s1.GetNumParts(p) + s2.GetNumParts(p)
    }

    int Is_Part (string p) {
       //Any existence of this part returns true
        return s1.IsPart(p) || s2.Is_Part(p)
    }

    String FindWarehouses(warehouses_id) {
        if(WarehousesList1.contains(warehouses_id)) {
            return "InventorySystem_1"
        }else if(WarehousesList2.contains(warehouses_id)) {
            return "InventorySystem_2"
        }else {
            return "none"
        }
    }
}
class InventorySystem_1 {
    PartList p;
    void AddPart (string w, string p) {
        p.add(w,p)
    }
    void DeletePart (string w, string p) {
        p.remove(w,p)
    }

    int GetNumParts (string p) {
        return p.getParts(p)
    }
    int IsPart (string p) {
        return p.contains(p)
    }
}
class InventorySystem_2 {
    PartList p;
    void Insert_Part (string w, string p) {
        p.add(w,p)
    }
    void Remove_Part (string w, string p) {
        p.remove(w,p)
    }

    int Get_Num_Of_Parts (string p) {
        return p.getParts(p)
    }
    int Is_Part (string p) {
        return p.contains(p)
    }
}
```

PROBLEM #3

N-Version architecture

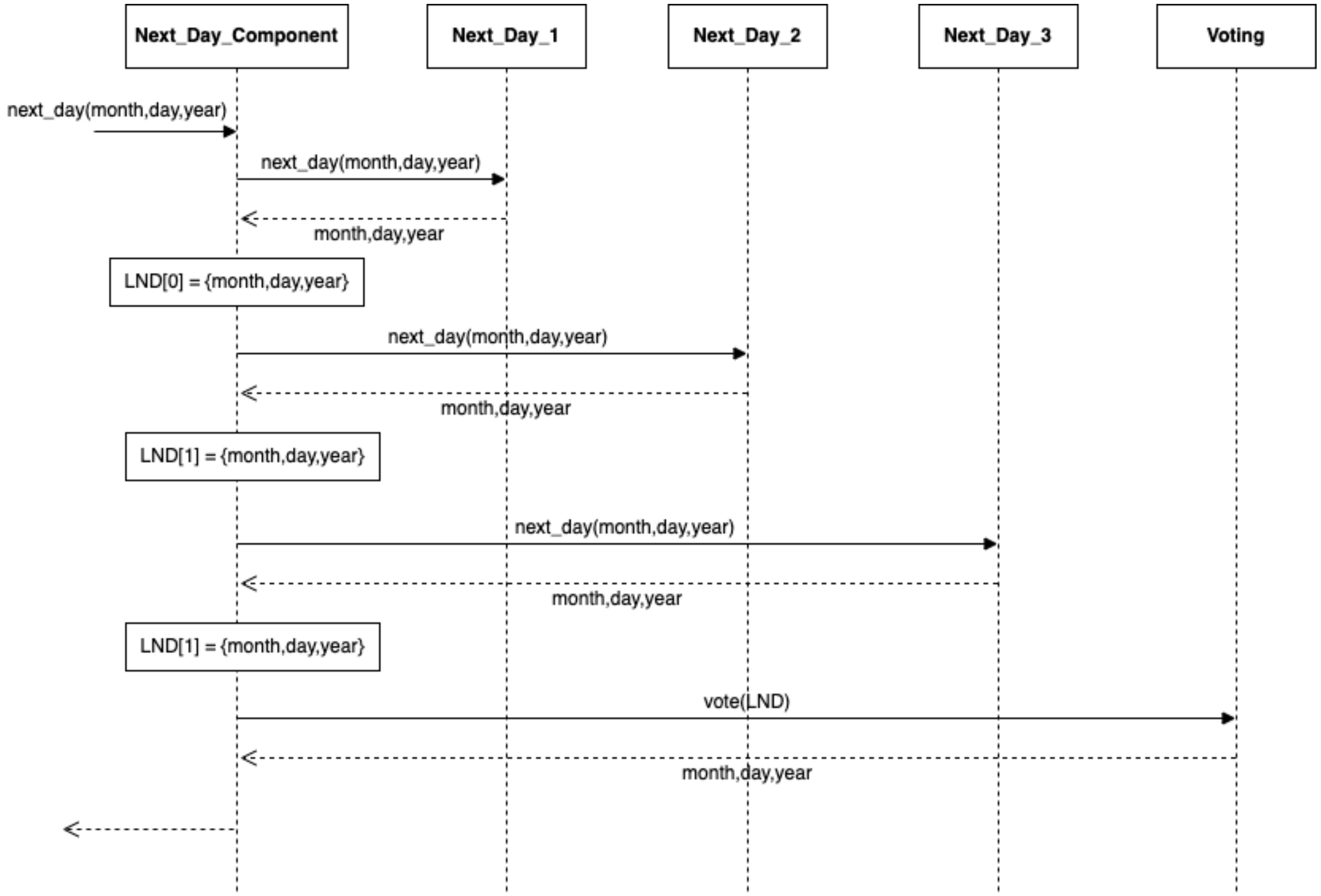
UML



pseudo-code

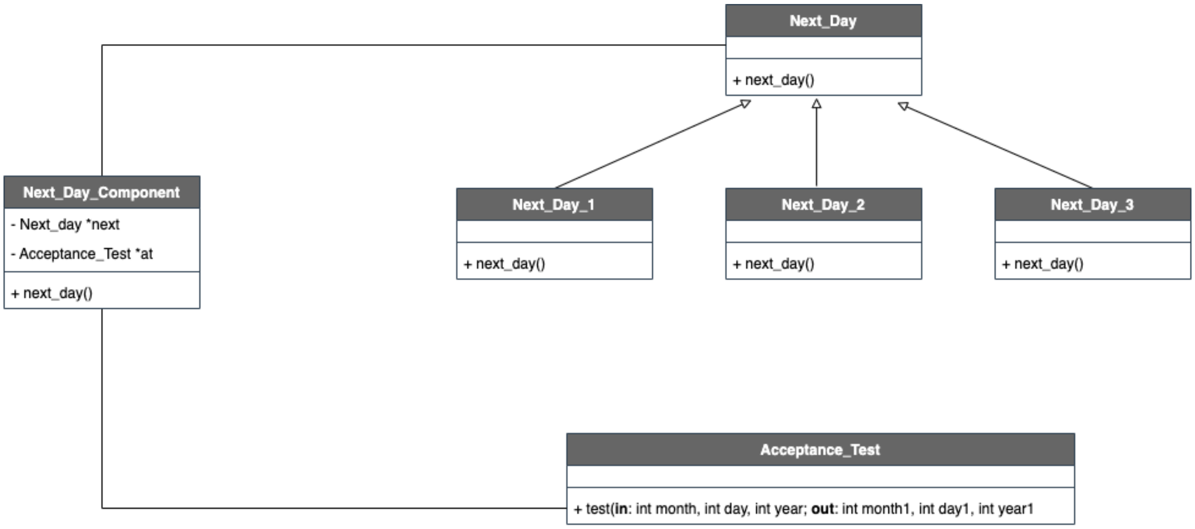
```
class Next_Day_Component {
    Next_Day next;
    Voteing v;

    next_day(int month, int day, int year) {
        LND[] //An array containing 3 sets of year, month, and day, similar to
[{yyyy, mm, dd}, {yyyy, mm, dd}, {yyyy, mm, dd}]
        Next_Day[]
        Next_Day[0] = new Next_Day_1()
        Next_Day[1] = new Next_Day_2()
        Next_Day[2] = new Next_Day_3()

        for(int i=0; i<Nexy_Day.length; i++) {//iterate  arrays

            y,m,d = Next_Day[i].nexy_day(month,day,year);
            LND[i] = {y,m,d};
        }
        v.vote(LND)
    }
}
class Voting {
    {year,month,day} vote(LND) {
        if(LND[0] == LND[1]) {
            return LND[0]

        } else if(LND[1] == LND[2]) {
            return LND[1]

        } else if(LND[0] == LND[2]) {
            return LND[2]
        }

        int n = Random(0,2)//Randomly assign a number from 0 to 2
        return LND[n] //randomly select one from LND and return
    }
}
```

sequence diagram



Recovery-Block architecture

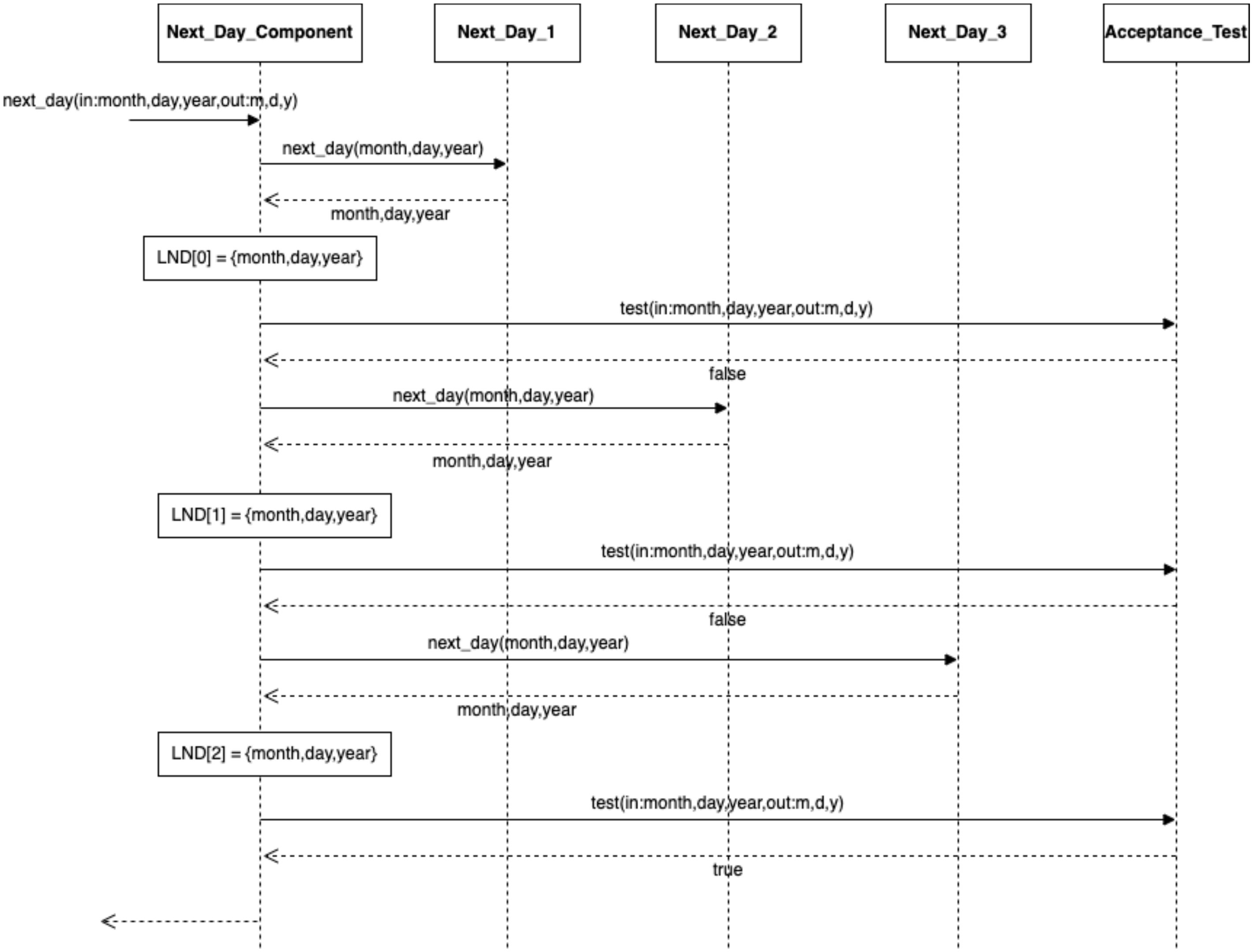
UML



pseudo-code

```
class Next_Day_Component {
    Next_Day next;
    Acceptance_Test at;

    next_day(int month, int day, int year) {
        LND[] //An array containing 3 sets of year, month, and day, similar to [{yyyy, mm, dd}, {yyyy, mm, dd}, {yyyy, mm, dd}]
        Next_Day[]
        Next_Day[0] = new Next_Day_1()
        Next_Day[1] = new Next_Day_2()
        Next_Day[2] = new Next_Day_3()

        y,m,d = Next_Day[0].next_day(month,day,year)
        LND[0] = {y,m,d};
        boolean result = at.test(month,day,year,m,d,y)
        if(result) {
            return;//exit
        }

        y,m,d = Next_Day[1].next_day(month,day,year)
        LND[1] = {y,m,d};
        boolean result = at.test(month,day,year,m,d,y)
        if(result) {
            return;//exit
        }

        y,m,d = Next_Day[2].next_day(month,day,year)
        LND[2] = {y,m,d};
        boolean result = at.test(month,day,year,m,d,y)
        if(result) {
            return;//exit
        }
        //all tests are false
        int n = Random(0,2)//Randomly assign a number from 0 to 2
        LND[n] //randomly select one from LND and return
    }
}
public class Acceptance_Test {

    /**
     * Validates that the result date is indeed the next day of the input date.
     * This method integrates checking for leap years and the calculation of days in the month.
     *
     * true if the result date correctly represents the next day; false otherwise.
     */
    public boolean test(in:int month, int day, int year, out:int Month, int Day, int Year) {
        // Check for the transition to a new year (December 31st to January 1st).
        if (month == 12 && day == 31) {
            return Month == 1 && Day == 1 && Year == year + 1;
        }

        // Check if it's the end of the month, which would require a month transition.
        // It needs to account for both the month changing and the year changing if
it's December.
        if (day == getDaysInMonth(month, year)) {
            return Month == (month % 12) + 1 && Day == 1 && Year == (month == 12 ?
year + 1 : year);
        }

        // Check for a regular day increment (e.g., middle of the month).
        // The result should be the same month, the next day, and the same year.
        return Month == month && Day == day + 1 && Year == year;
    }

    /**
     * Determines the number of days in a specified month, taking leap years into
account.
     *
     *  month: The month number (1 for January, 2 for February, etc.).
     *  year:  The year, used to check for leap years in February.
     * return The number of days in the month.
     */
    private int getDaysInMonth(int month, int year) {
        // April, June, September, and November have 30 days.
        if (month == 4 || month == 6 || month == 9 || month == 11) {
            return 30;
        }

        // February's days depend on whether it's a leap year or not.
        if (month == 2) {
            // Check for a leap year: divisible by 4, not 100 unless also divisible
by 400.
            if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {
                return 29;
            }
            // Not a leap year so February has 28 days.
            return 28;
        }

        // All other months have 31 days.
        return 31;
    }
}
```

sequence diagram


Next_Day_Component
Next_Day_1
Next_Day_2
Next_Day_3
Acceptance_Test
next_day(in:month,day,year,out:m,d,y)
next_day(month,day,year)
month,day,year
LND[0] = {month,day,year}
test(in:month,day,year,out:m,d,y)
false
next_day(month,day,year)
month,day,year
LND[1] = {month,day,year}
test(in:month,day,year,out:m,d,y)
false
next_day(month,day,year)
month,day,year
LND[2] = {month,day,year}
test(in:month,day,year,out:m,d,y)
true