Case Study Review-II

Military Personnel Management System

S.no	Roll No	Name
1	CB.EN.U4CSE20605	Ashwath V A
2	CB.EN.U4CSE20606	Ayyagari Sai Anish
3	CB.EN.U4CSE20611	Yashasvi Chowta
4	CB.EN.U4CSE20630	Kishore V

Contents

I. Problem Statement

The military power of a country is very essential to protect the country from external and non-native threats which ensures the safety of the given country's citizens. So, it is very essential for the military to function in an efficient manner. This is solely possible when there is a chain of command. This implies that there are officers in the military who hold different ranks which indicate their position in the chain of command. In such situations where there is a clear chain of command, to ensure that everyone is following orders of a superior officer, to ensure minimum error in the decision making/implementation process.

The overall strength of the armed forces varies depending on a range of factors. There are too many applicants for the armed forces every year. Out of these applicants, only the worthy of the bunch are to selected and given the job. Each recruited soldier needs to be allotted with a starting level, posting location and many more. This newly recruited soldier needs to be added to a list which mentions all the military personnel by the order of superiority. This list of personnel is not a constant list as many soldiers are recruited and many soldiers move up the ladder of ranks after few years of service and are promoted to a new rank which would expand their current list of perks/allowances, hike their current salary etc. The previously mentioned list of soldiers would also have to be changed as the officer's rank changes. Thus, the list of military personnel is a list that needs constant updating.

The personnel list might be also changed in the situation where the soldier is unable to provide further service to the armed forced. This may happen when the soldier willingly retires from the force or is incapable of active service due to severe injury or is killed in action. In this case these soldiers must be removed from the existing personnel list and need to be moved to a special list which would make them eligible to receive pension.

As previously mentioned, there is a chain of command that is present in the armed forces. With increase in the rank, the set of tasks and responsibilities of the soldier expands and it would only be fair if they are compensated differently due to increased responsibilities and rewards like allowances. The task of calculation of salaries of each officer based on the ranks and granting the allowance money for each soldier is a task that is prone to errors which needs to be avoided.

Awarding a person is an honorary attachment to the awardee which acknowledges the exceptional performance of the awardee in a particular task. The armed forces are an organization consisting of many people providing such exceptional service for which they are under-compensated. Thus, these officers are awarded with some of the prestigious awards is a well-organized ceremony grazed by the heads of the armed forces and the

country. The awards usually are coupled with a cash prize which needs to be credited into the bank account of the awardee.

For the proper functioning of any organization, the basic requirement is that the management of the organization needs to verify the records of personnel, transactions etc. on a regular basis. In huge personnel organizations like the army, navy and the air force, these records would contain a lot of data and would be humanly impossible to maintain and verify the correctness of these records manually

Thus, having a physical bound ledger consisting of all the records, lists, payments, etc. would be impossible to maintain and update on a regular basis. Thus, it is will never remain error free or up to date on a regular basis. Thus, there is a need of an efficient system which would be able to do these tasks and keep it error free.

II. Actor – task-list

The various levels in the Indian military are:

Sepoy - Seaman – Aircraftsman ----------> Level 1

Major – Lieutenant commander – Squadron leader ----------> Level 2

Brigadier – Commodore – Air Commodore ----------> Level 3

General – Admiral – Air Chief Marshal ----------> Level 4

Actors	**task**
Officer	Recruitment of new soldiers Promotion of officers Payment of salary Allowance payment Retirement of military personnel Discharge of officers (Medical/KIA) Awards for meritorious service
System	Recruitment of new soldiers Promotion of officers

	Payment of salary Allowance payment Retirement of military personnel Discharge of officers (Medical/KIA) Awards for meritorious service Generation of reports
Senior officer	Recruitment of new soldiers Promotion of officers Allowance payment Retirement of military personnel Awards for meritorious service Generation of reports
Manager	Retirement of military personnel Payment of salary Allowance payment Discharge of officers (Medical/KIA)
payment	Payment of salary Allowance payment Awards for meritorious service
Personnel	Recruitment of new soldiers
Record	Recruitment of new soldiers Generation of reports
Promotion record	Promotion of officers Generation of reports
Retirement Record	Retirement of military personnel Discharge of officers (Medical/KIA) Generation of reports
Award record	Awards for meritorious service Generation of reports
Transaction record	Payment of salary Allowance payment

	Generation of reports
Veteran	Retirement of military personnel Payment of salary Discharge of officers (Medical/KIA)
Award	Awards for meritorious service
Allowance	Allowance payment
Bank Account	Payment of salary Allowance payment Awards for meritorious service

Tasks:

1. Recruitment of officers
2. Promotion of officers
3. Payment of salary
4. Refunding allowances
5. Managing retirement of officers
6. Managing records of officer who are KIA (killed in action)/severely injured.
7. Giving prize money for awards
8. Generating reports

III. Use case document

1. Use Case Name: Recruitment of new soldiers

Description: The applicant enrolling to the armed forces is required to submit the basic details of the applicant, scores of the prerequired exams and physical tests are mentioned and the senior officer responsible for recruitment reviews the application and decides if the person should be selected or not and the necessary action is taken based on the decision of the recruiting officer.

Actors:

PrimaryActors:
Applicant
Recruitment Officer

Secondary Actors:
System

Trigger:

External event: The applicant submits their application to the armed forces

Preconditions:

The applicant should be a citizen of the country and should have appeared in the prerequisite examinations

Flow:

Basic Flow:

1. The applicant submits their application.
2. The recruiting officer reviews the application of the applicant and decides if the applicant is eligible for the job that the applicant is seeking.
3. If the Applicant is deemed worthy of the job, then the officer recruits the applicant.
4. The applicant receives the confirmation

5. The essential details of the applicant are added to the list of soldiers ordered by the ranks of the various soldiers
6. The system records the changes made into the record.

Exceptions:

1. If the applicant is deemed unfit for the job, the recruiting officer provides the news of rejection to the applicant.

Alternatives: NIL

Level: User-goal

Postcondition:

The applicant receives information about the status of their employment

Stakeholders:

Applicant

Department of the armed forces

Identifying nouns, conceptual classes and attributes:

Nouns: Applicant, personnel, Senior officer, system, officer, record, Recruits, soldiers

Noun as conceptual class: personnel, Senior officer, system, officer, record

Attributes

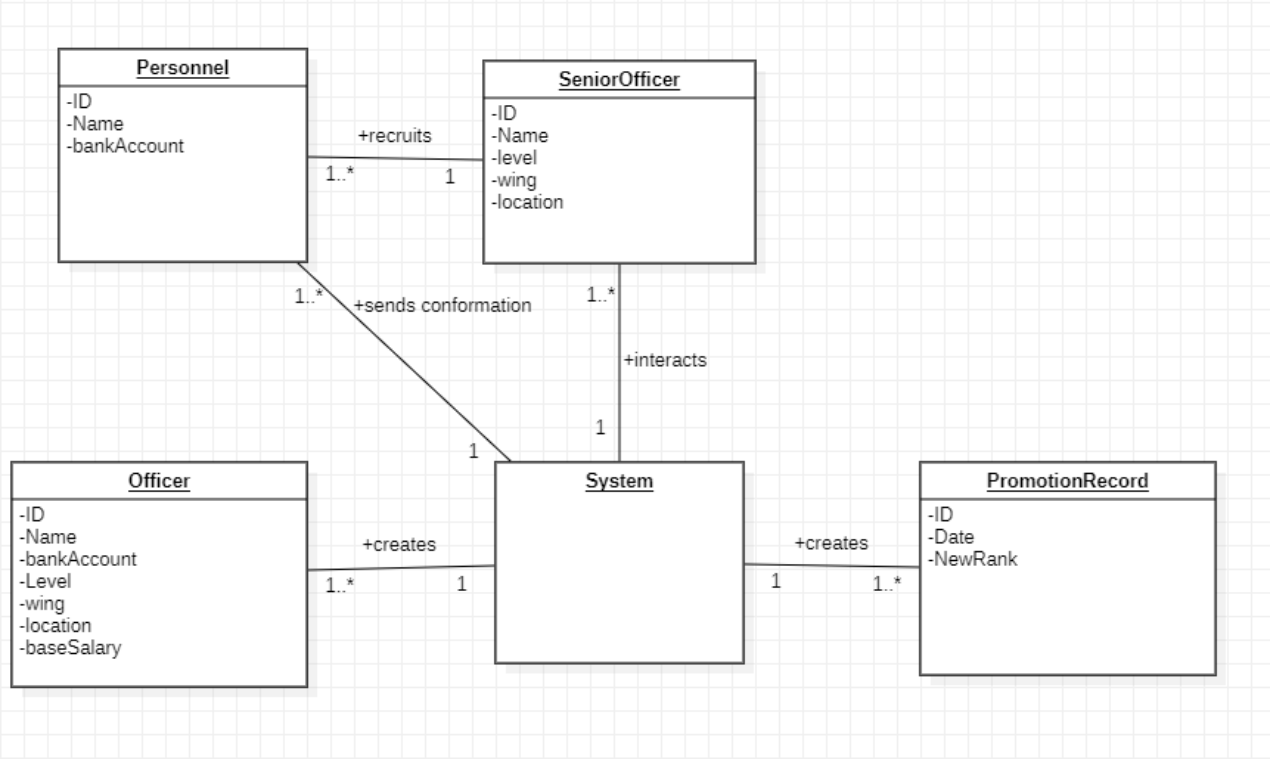
Personnel - id, name, bank account, date of birth

Senior officer – id, name, wing, level, location

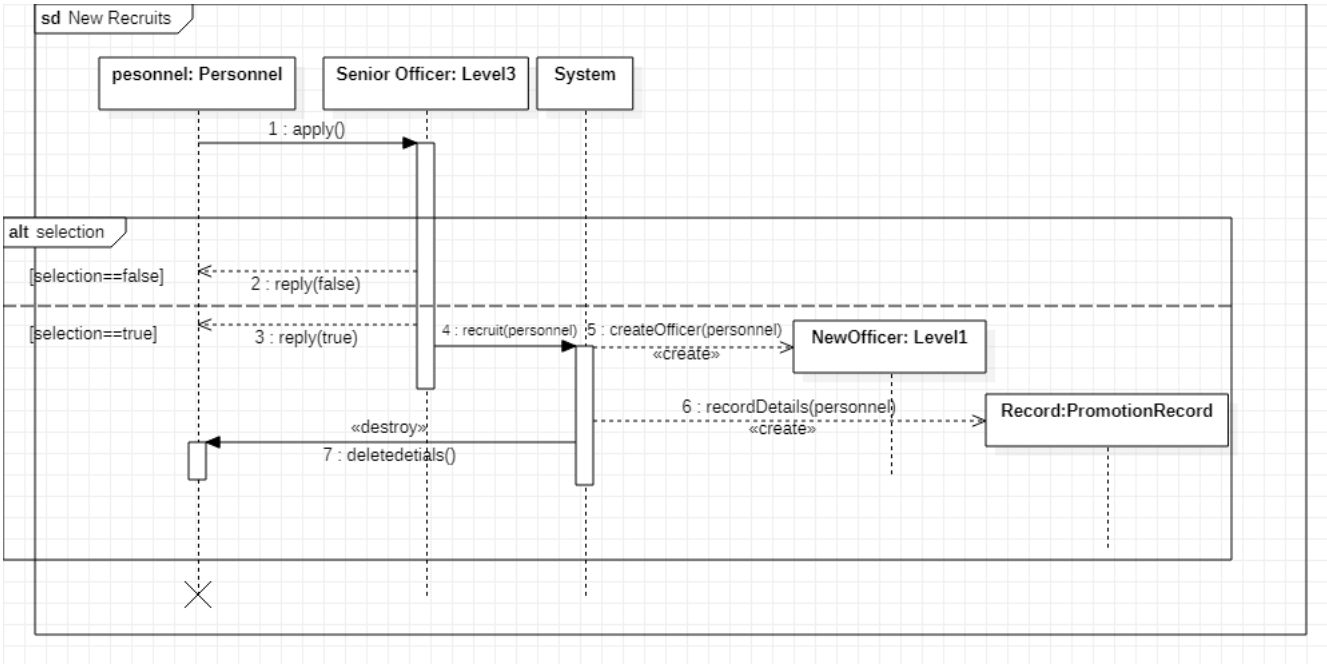
Level 1 officer- id, name, bank account, wing, level, location, base salary

Record – id, date, rank

Object Model:



Time sequence diagram:



2. Use Case Name: Promotion of officers

Description:

The Senior officer (level 3 or 4) has a chance of promoting a soldier. The senior officer finds a suitable candidate worthy of a well-deserved promotion based on the services provided by the soldier. The Senior officer then logs into the system and removes the officer from their existing position and assigns a new rank to the soldier.

Actors:
Primary Actors:
existing officer
Senior officer
System

Secondary Actors:
Senior Officer

Trigger:

External event: The Senior officer submits a recommendation on the applicant's behalf

Preconditions:

The applicant should be an existing officer in the armed forces

Flow:

Basic Flow:

1. The Level 3/4 officer (senior officer) logs into the system.
2. The senior officer selects the soldier suited for a promotion
3. The senior officer removes the officer's current rank and replaces it with a new rank
1. The officer is informed about their promotion
2. The changes are recorded into the record by the system

Exceptions:

When Level 3 or 4 senior officer tries to promote personnel from other wings. The senior officer is informed by the system that they are restricted from promoting soldiers from another wing.

Alternatives: NIL

Level: User goal

Postcondition:

The officer is either promoted to a new title or stays at the same level

Stakeholders:

Officer

Department of defense

Identifying nouns, conceptual classes and attributes:

Nouns: Senior officer, system, officer, record, soldiers

Nouns as conceptual class: Senior officer, system, officer, promotion record

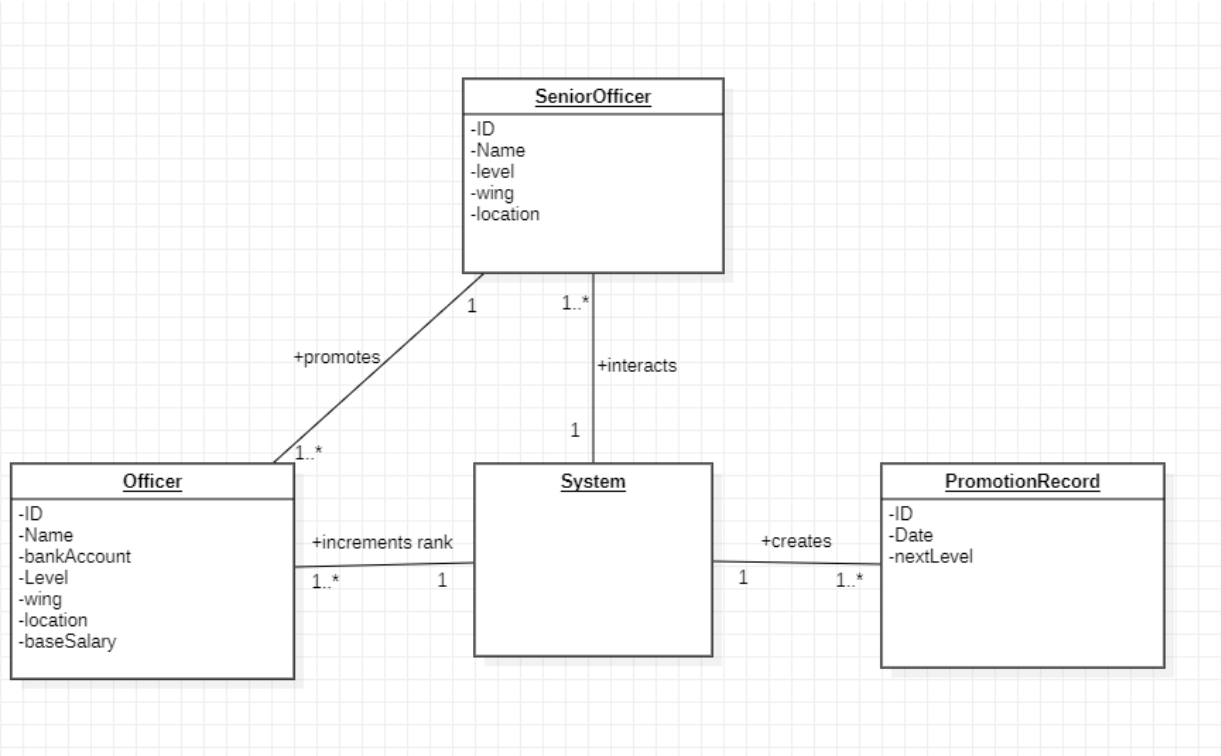
Attributes:

Senior officer – id, name, wing, level, location

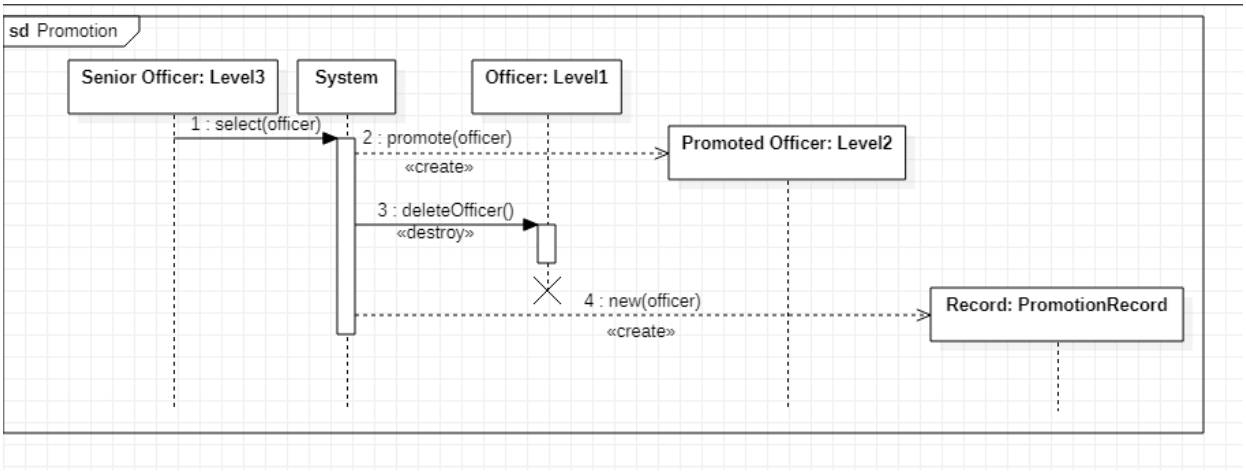
officer- id, name, bank account, wing, level, location, base salary

Promotion record – id, date, next level

Object Model:



Time sequence diagram



3. Use Case Name: Payment of salary

Description: This use case begins every month during payment of salary. A manager will generate the salary of the military personnel using the system. The system will calculate the basic salary by level of the solider. Payment will be generated and a transaction would be created with an id. For the retired officer pension amount will be transferred to their account. After all the transactions, the system will create transaction record.

Actors:

Primary Actors:
System
Officer

Secondary Actors:
Manager

Trigger: First of the month

Preconditions: He/she should be working in air force or navy or army

Flow:

Basic Flow:

1. Every First of the month, manager initiates salary payment process.
2. According to the officer level/ veteran, System will consider a base salary
3. The amount will be credited officers/veteran bank account.
4. The system records this transaction

Exceptions:

1. If transaction stops in the middle of transferring, the system maintains the atomicity and continues with the previous officer.
2. If the bank server does not respond, payment is carried out after some time.

Alternatives:

The manager transfers the basic salary to the officer's bank account and this transaction is recorded in the system.

Level: Kite level

Postcondition: After transferring the amount everything needs to be recorded in the system and a message is sent to the officer that payment is successful.

Stakeholders:
Department of defense
Officers

Identifying nouns, conceptual classes and attributes:

Nouns: Manager, payment, officer, veteran, system, transaction, bank account, transaction record, soldier, army, navy, air force

Nouns as conceptual class: Manager, payment, officer, veteran, system, bank account, transaction record

Attributes:

Manager – Id, name

Payment – id, account no, amount, type

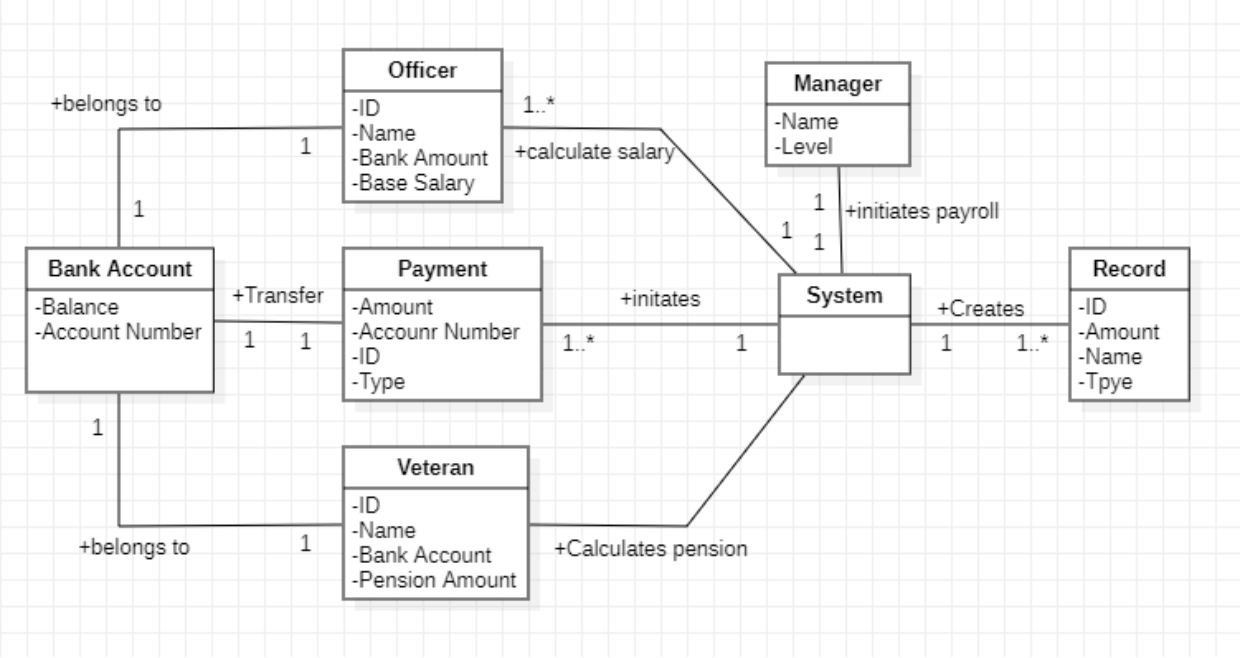
Bank account – account number, balance

officer- id, name, bank account, wing, level, location, base salary

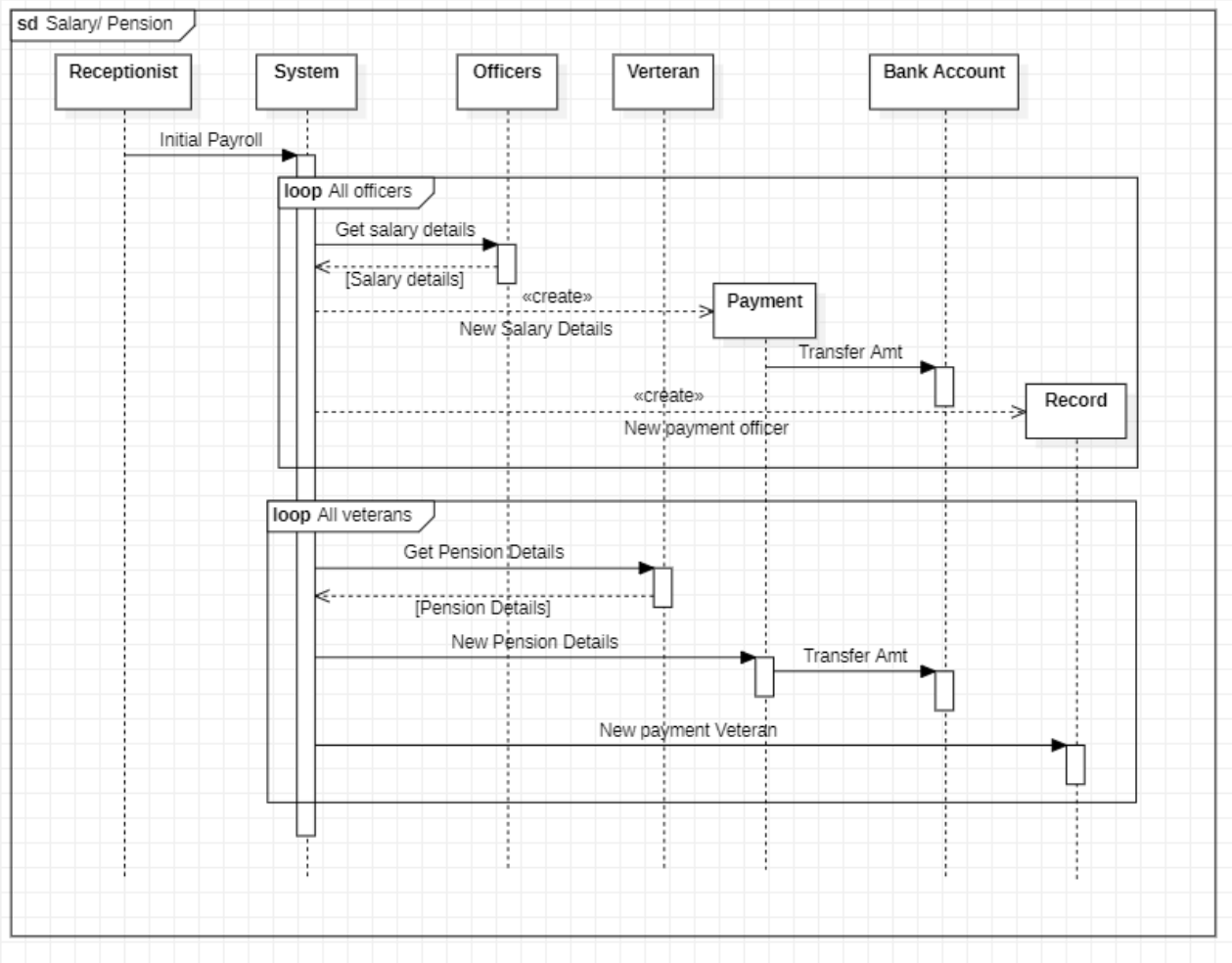
Veteran – id, name, pension amount, bank amount

Transaction record – bank account number, amount, id, date

Object Model:



Time sequence diagram:



4. Use case name: Allowance payment

Description:

Allowances payment process is a part of the military management system which handles the processing of allowances for the military personnel. According to the 7th Pay commission the military personnel are awarded allowances according to the level in which they are in the pay scale. Allowances are mainly categorized into two, one where the personnel receive the allowances implicitly as part of their salary and the other where they submit the necessary bills and claims the allowances. This allowance payment process deals with latter part. The concerned personnel would submit the necessary bills to the manager who validates and authorizes the allowance payment process.

The allowances are awarded according to the ranks of the personnel

S.no	Category	Level	Allowance	Amount
1	Army	L2	Classification allowance	225 pm
			Rum allowance	360 pm
		L3	Composite personal maintenance allowance (CPMA)	90 pm
			Dress allowance	20000 pa
			Extra allowance	2% of basic pay
		L4	Language allowance	2025 pm
			Post graduate allowance	2250 pm
2	Navy	L2	Diving allowance	1800 pm
			Dip Money	3500 pm
			Diving Attendant allowance	700 pm($1/5^{th}$ of Dip money)
		L3	Island special duty allowance	16% of basic pay
			Sea going allowance	10500 pm
		L4	Submarine Duty Allowance	5300 pm
			Submarine technical allowance	1000 pm
3	Airforce	L2	Air worthiness certificate allowance	338 pm
			High altitude allowance	5300 pm
			Flying allowance	17300 pm
		L3	Composite personal maintenance allowance (CPMA)	20000 pa
			Dress allowance	20000 pa
		L4	Flying allowance	25000 pm

Actors:

Primary actors:

Officer

Secondary actors:

Manager

System

Trigger:

The trigger for this use case is an external event which is started when the military personnel submit the necessary details to the manager to claim the allowance.

Precondition:

Officer must be of level 2 or higher.

Flow:

Basic flow:

1. The allowances are awarded after the military personnel submit the necessary documents. The allowances are classified according to the ranks of the personnel
2. The military personnel provide the necessary document to the manager who then checks whether the produced documents are for valid allowances as mentioned above.
3. If the officer is eligible for the allowances the manager authorizes the transaction of funds from the defense reserve to recipient.

Alternate:

Nil

Exception:

If the provided documents are insufficient are not for the above-mentioned category the receptionist conveys the same to the officer who is either asked to provide some more necessary documents or is told that there does not exist any category like for which they are applying.

Level:

User goal

Postcondition:

The military personnel receive the allowance

Stakeholders:

department of the defense

Officer

Identifying nouns, conceptual classes and attributes:

Nouns: Senior officer, officer, manager, system, air force, navy, army, personnel, allowance, transaction, transaction record, payment

Nouns as conceptual class: bank account, manager, system, transaction record, payment, allowance

Attributes:

Manager – Id, name

Payment – id, account no, amount, type

Bank account – account number, balance

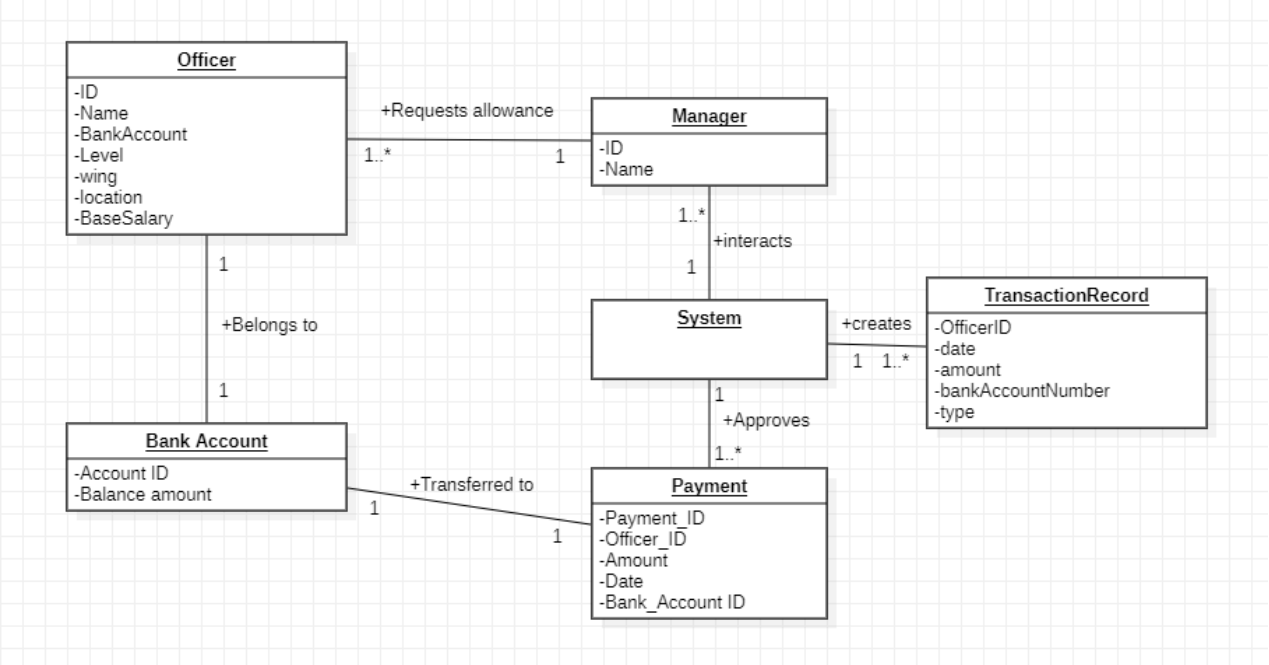
officer- id, name, bank account, wing, level, location, base salary

Senior officer – id, name, wing, level, location

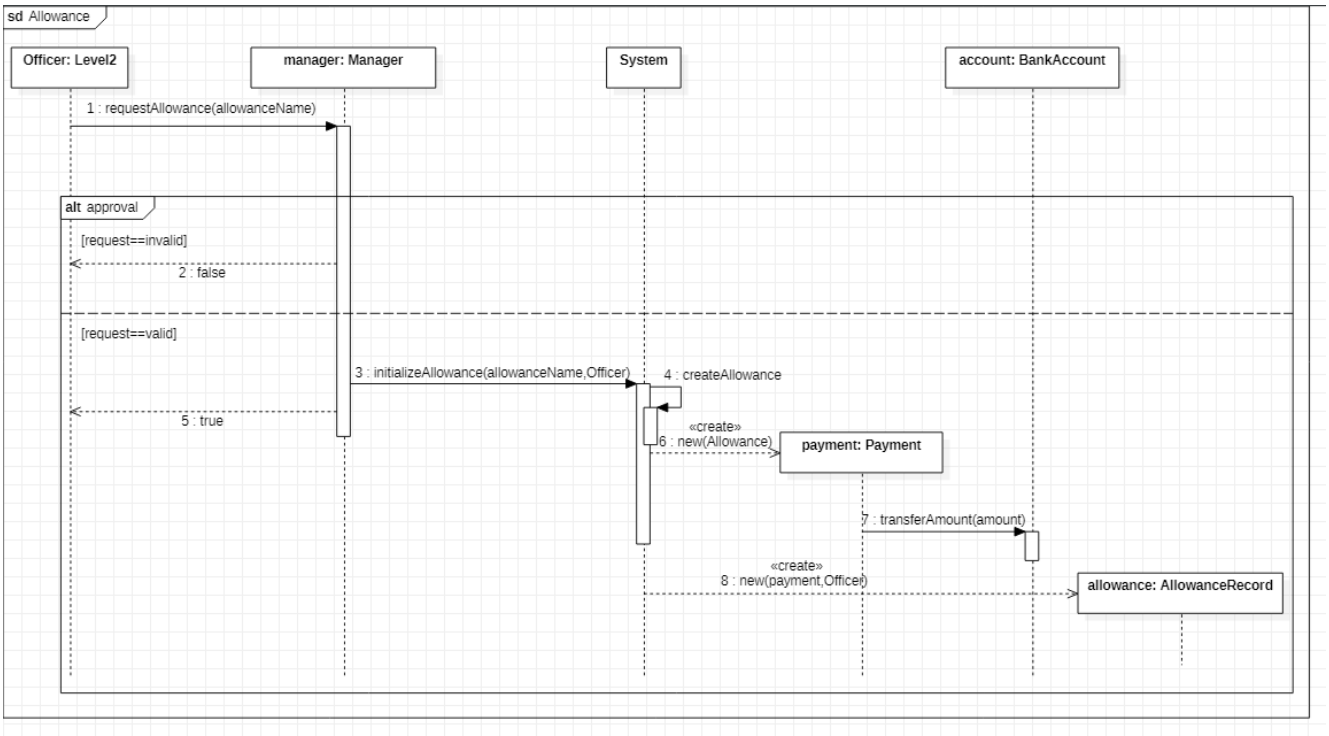
Allowance – name, amount, wing, minimum level

Transaction record – bank account number, amount, id, date

Object Model:



Time sequence diagram:



5. Use case name: Retirement of military personnel

Description: Any salaried employee can decide when they want to stop leading a working life and proceed to retirement. An officer belonging to the armed forces can take up retirement by submitting an application stating their desire to no longer be employed in the armed forces. The application is audited by a reviewing officer who has the power to approve the request of the applicant. After the request of the applicant is approved, they are removed from the existing soldiers list and are placed in a special list which would identify them as a pensioner.

Actors:

Primary Actors:

Military personnel of any level

Secondary Actors:

Manager

Triggers:

External event: officer wants to retire for any reason or has reached the retirement age. Precondition:

The Personnel must be an officer with the required years of service and/or has reached the retirement age.

Flow:

Basic flow:

1. The manager logs into the system.
2. The officer applies for retirement and submits the required document. The application is validated by the system and waits for the senior officer's approval.
3. After the approval, the officer is removed from active service and moved to the veteran list. The veteran begins to receive pension every month according to his final rank in service.
4. The system records this in the retirement record.

Alternate:

If the officer reaches the retirement age corresponding to their level, they are removed from the active service list and moved to the veteran list

Exceptions:

If the officer does not meet the minimum requirements for the retirement, they are not allowed to.

Level: User Goal

Postcondition:

The officer retires from service.

Stakeholders:

Officer

Department of defence

Identifying nouns, conceptual classes and attributes:

Nouns: Manager, senior officer, veteran, system, record, retirement record, pension, approval

Nouns as conceptual class: manager, senior officer, veteran, system, retirement record

Attributes:

Manager - id, name

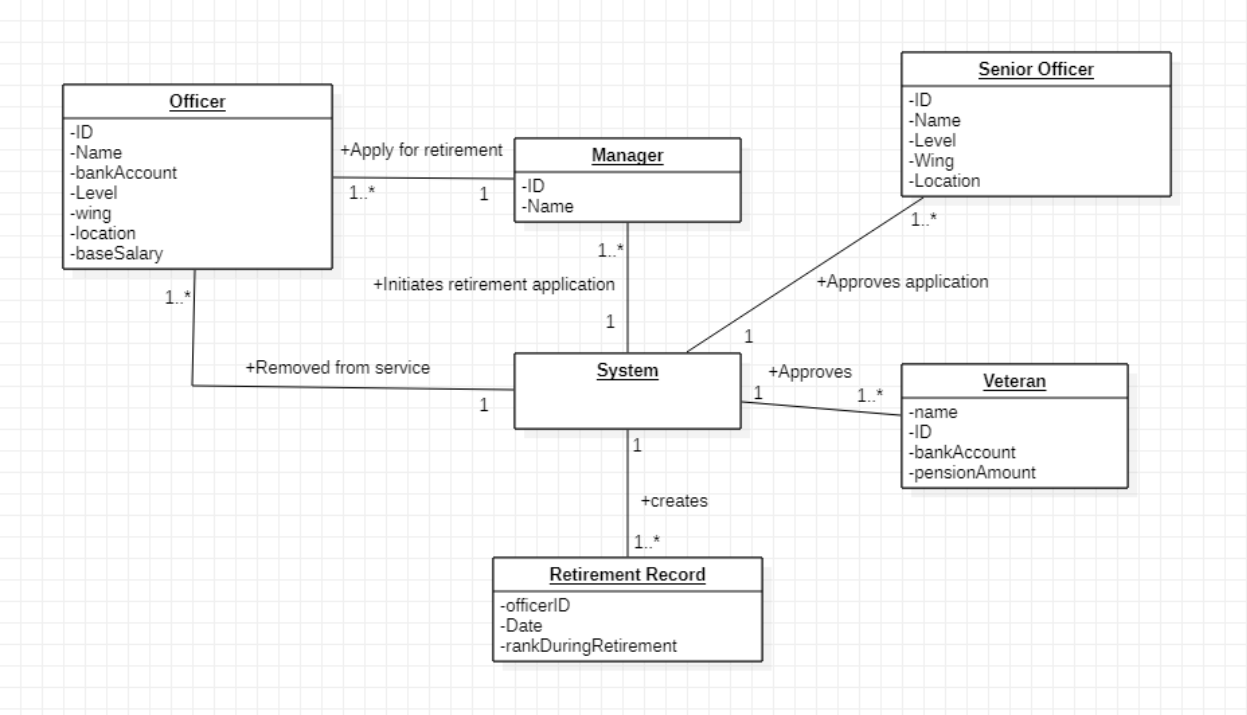
Senior officer – id, name, wing, level, location

officer- id, name, bank account, wing, level, location, base salary

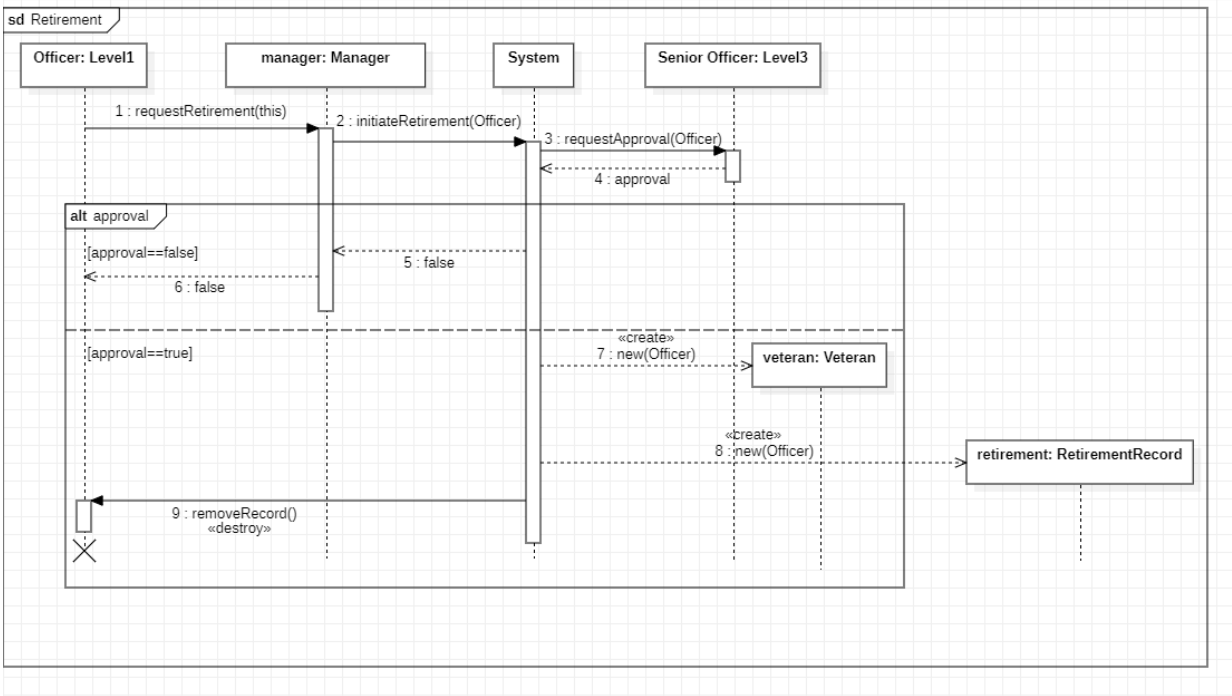
Veteran – id, name, pension amount, bank amount

Retirement record – rank during retirement, date, id

Object Model:



Time sequence diagram



6. Use case name: Discharge of officers (Medical/KIA)

Description: The job of a soldier is very perilous and often due to many factors, a soldier may be eliminated in the line of duty or maybe severely injured to level due to which they may no longer serve in the armed forces. In such cases, the death/injury of the soldier is confirmed by the doctor through a detailed report. The manager on receiving this report removes the officer from the list of active personnel and places them in the veteran list which would allow them to receive their pension.

Actors:

Primary Actors:

Manager

Triggers:

External event: Military personnel is killed in action or deemed unfit for service by the doctor.

Precondition:

The personnel of concern should exist in the military record

Flow:

Basic:

1. The manager receives the report from doctor
2. The personnel is identified
3. The soldier is removed from active service
4. The soldier is recorded into veterans list and are awarded pension according to the respective category
5. The system records the discharge from service

Alternate: NIL

Exceptions:

Server goes down halfway through the process. The transaction is not completed and the state before the transaction is restored. The manager is asked to redo the process again.

Level: User goal

Postcondition:

The discharged soldier or next of kin (in case of KIA) begins to receive pension

Stakeholders:

department of the defense

Soldier (Family of soldier in case of KIA)

Identifying nouns, conceptual classes and attributes:

Nouns: manager, officer, veteran, doctor, doctor's report

Nouns as conceptual class: manager, officer, veteran

Attributes:

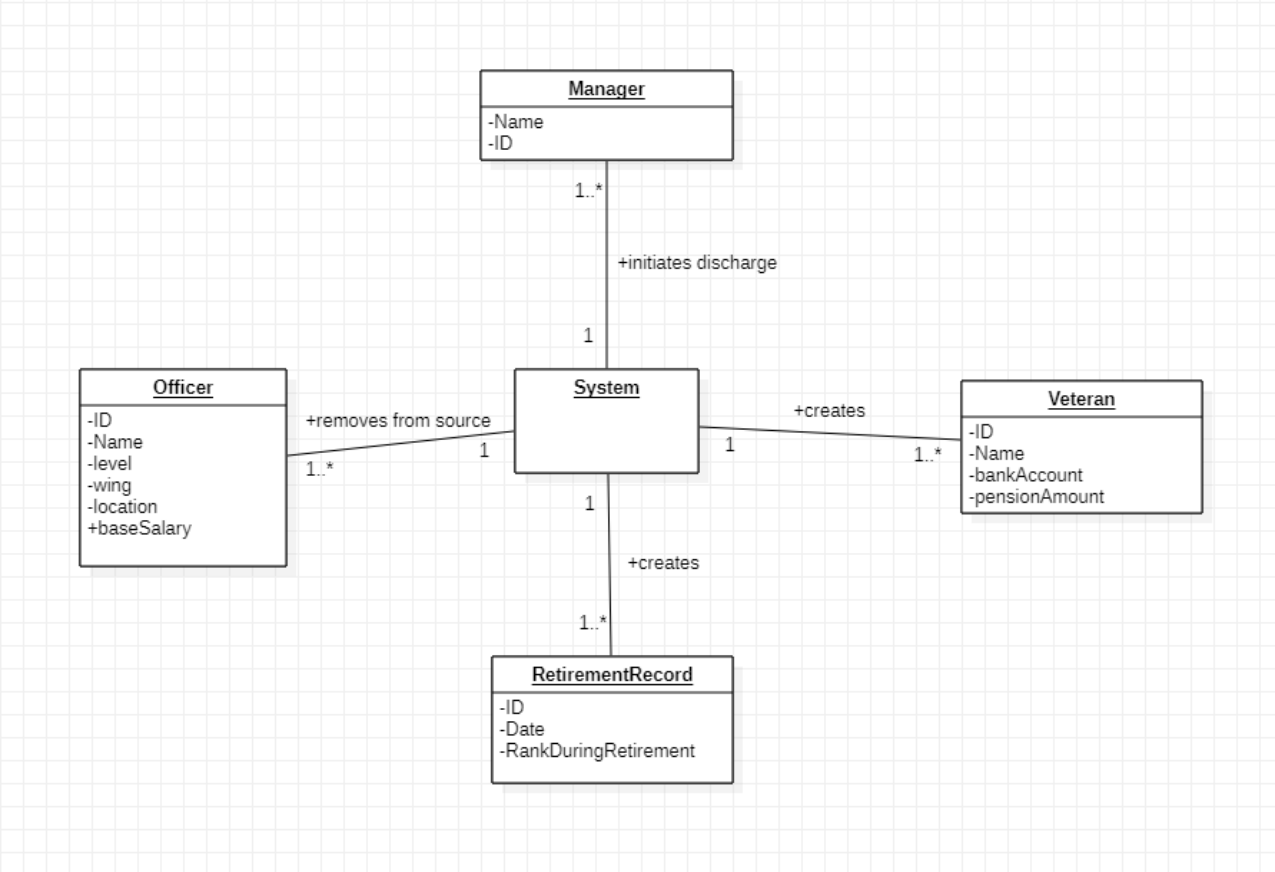
Manager – Id, name

officer- id, name, bank account, wing, level, location, base salary

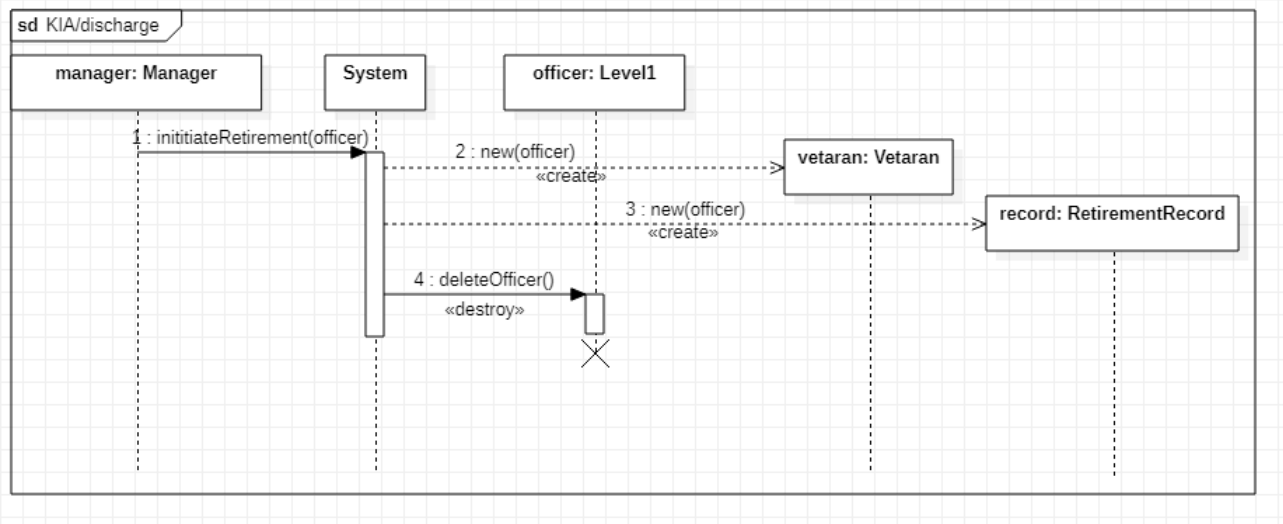
Veteran – id, name, pension amount, bank amount

Retirement record – rank during retirement, date, id

Object Model:



Time sequence diagram:



7. Use case name: Awards for meritorious service

Description: The armed forces is an organization filled with people who have successfully achieved many feats which have either directly or indirectly saved many lives. Thus, such great services are usually acknowledged by awarding the soldiers with some awards. A level 4 officer selects an awardee candidate based on their meritorious services. The soldier is felicitated with the award on an auspicious day in front of a crowd consisting of their peers and senior ranking officers. The prize money that is associated with the award is credited into the awarded soldier's bank account.

Actors:

Primary actor:

L4 military personnel

Secondary actor:

Officer

Triggers:

External event: Senior officer identifying an officer for exceptional service

Precondition:

Military personnel perform exceptional service for the nation

Flow:

Basic flow:

1. The Level 4 personnel logs into the system.
2. The officer selects the personnel for his meritorious services
3. The honor and the award are bestowed upon the selected soldier by the level 4 officer
4. The prize money is transferred to the account of the military personnel.

Alternate: NIL

Exceptions:

When L4 personnel tries to award to personnel from other wings. The L4 officer is informed by the system that they can give awards to those under their command only and is redirected to the home page.

Level: User goal

Postcondition:

The soldier is given the award and the prize money is transferred to his account.

Stakeholders:

Department of the defense

Officer

Identifying nouns, conceptual classes and attributes:

Nouns: Senior officer, level 4 personnel, award, prize money, payment, account, soldier, bank account

Nouns as conceptual class: Senior officer, Record, System, Award, payment, officer, bank account

Attributes:

Senior officer – id, name, wing, level, location

officer- id, name, bank account, wing, level, location, base salary

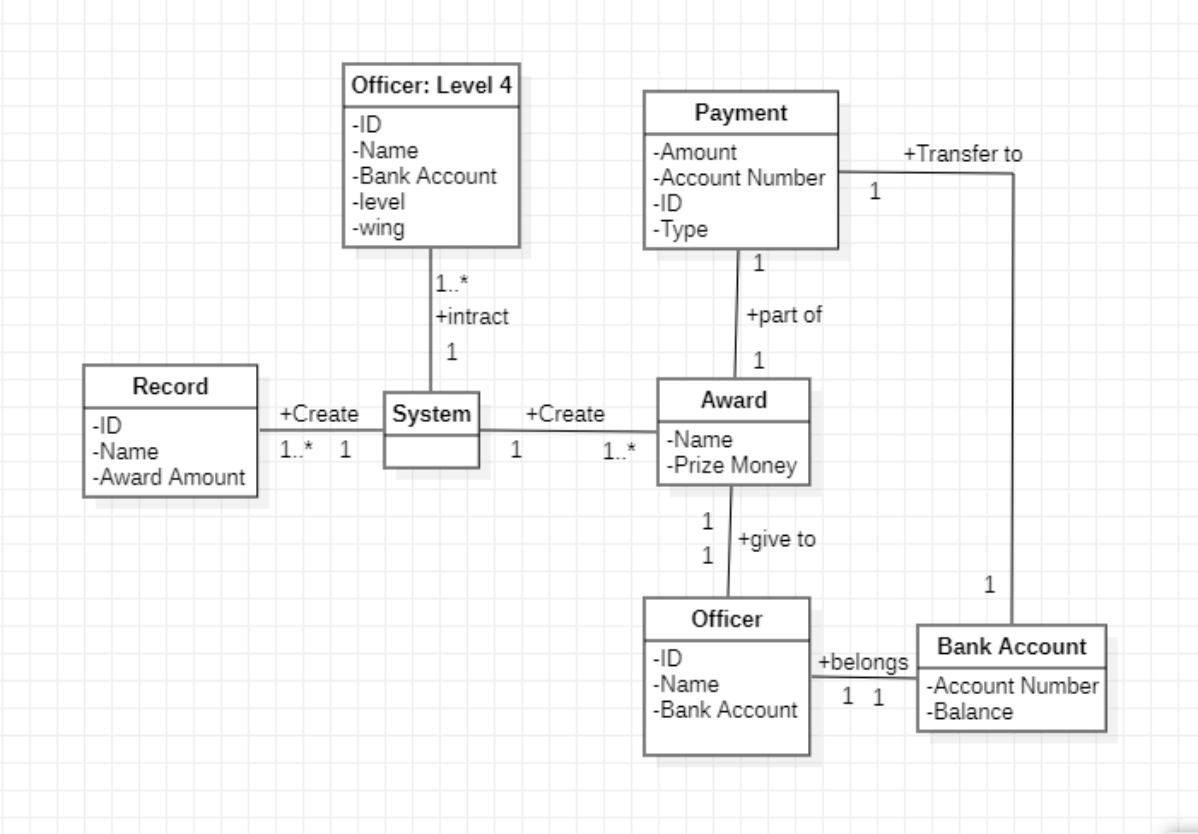
Award record - award name, prize money, id, date

Award – name, prize money, date

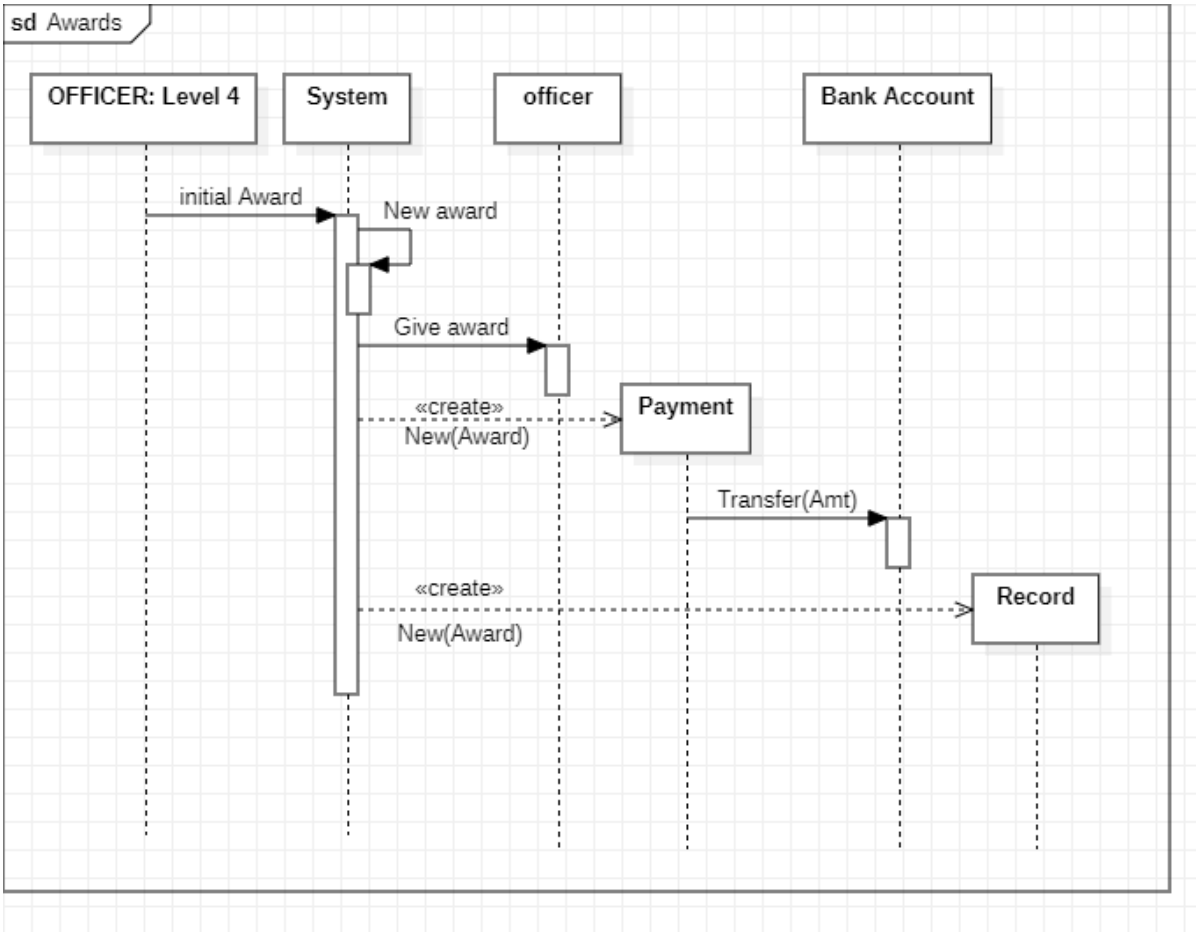
Payment – id, account no, amount, type

Bank account – account number, balance

Object Model:



Time sequence diagram:



8. Use case name: Generation of reports

Description: The functioning of the armed forces like any other organization which provides some sort of work to the people is heavily dependent on managing human resources. It is particularly important to access the records of the armed forces on a regular basis to ensure that no resources are being wasted and is being allocated to the pre-established categories. The reports consist of the transactions, promotions, awards and retirement. These reports are accessible by officers of level-4 who analyse these reports and take the required action.

Actors:

Level 4 officers

Triggers:

External event: Level 4 officers request for reports

Precondition: All transactions are recorded.

Flow:

Basic flow:

1. The Level 4 officer logs into the system and requests for reports.
2. The system searches through the available records.
3. The system filters and compiles the records according to the search categories provided by the officer.
4. The report is generated.
5. The generated report is displayed to the officer.

Alternate: NIL

Exceptions:

If the report file is corrupted. The system regenerates the report.

Level: User goal

Postcondition:

The financial report is generated and displayed to the user.

Stakeholders:

Department of defence

Identifying nouns, conceptual classes and attributes:

Nouns: Senior Officer, Report, Record, Search Category, System, transaction record, award record, promotion record, retirement record

Nouns as conceptual class: Senior Officer, Record, transaction record, award record, promotion record, retirement record

Attributes:

Senior officer – id, name, wing, level, location

Record – id, date

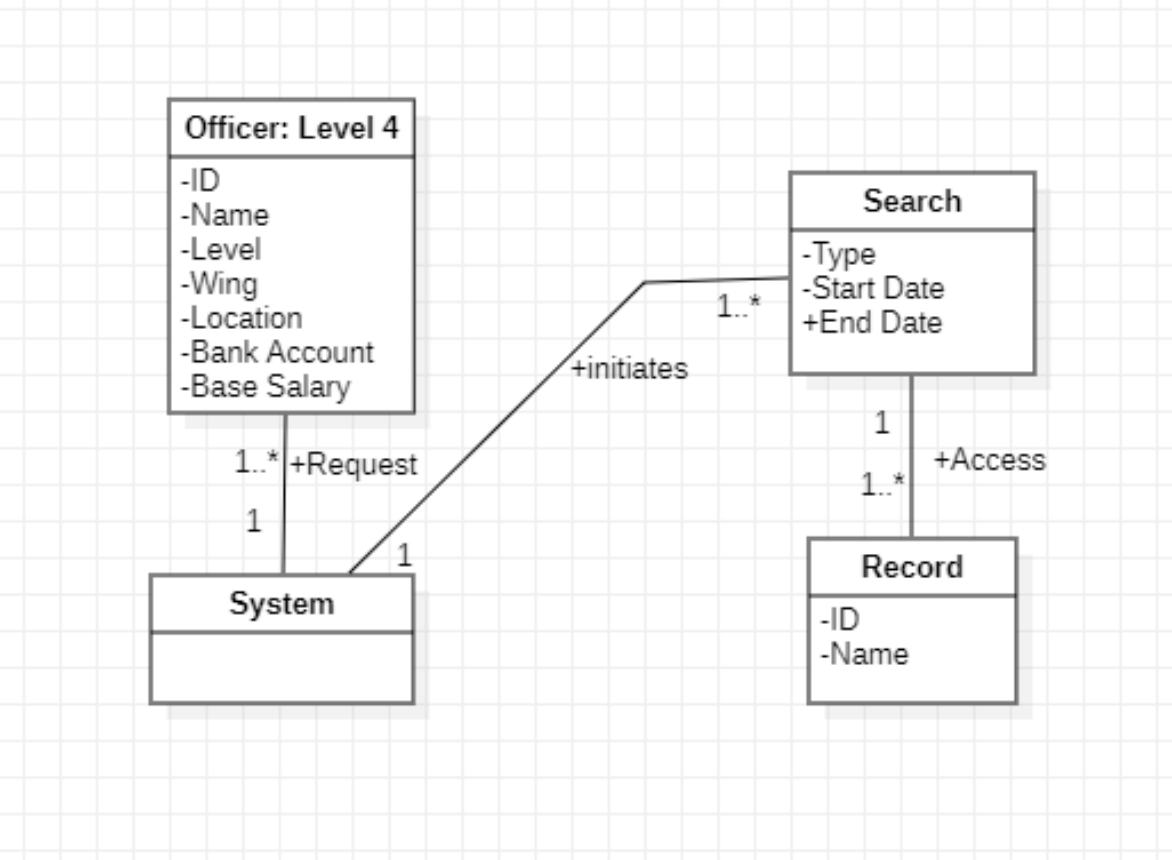
Promotion record – id, date, next level

Transaction record – bank account number, amount, id, date

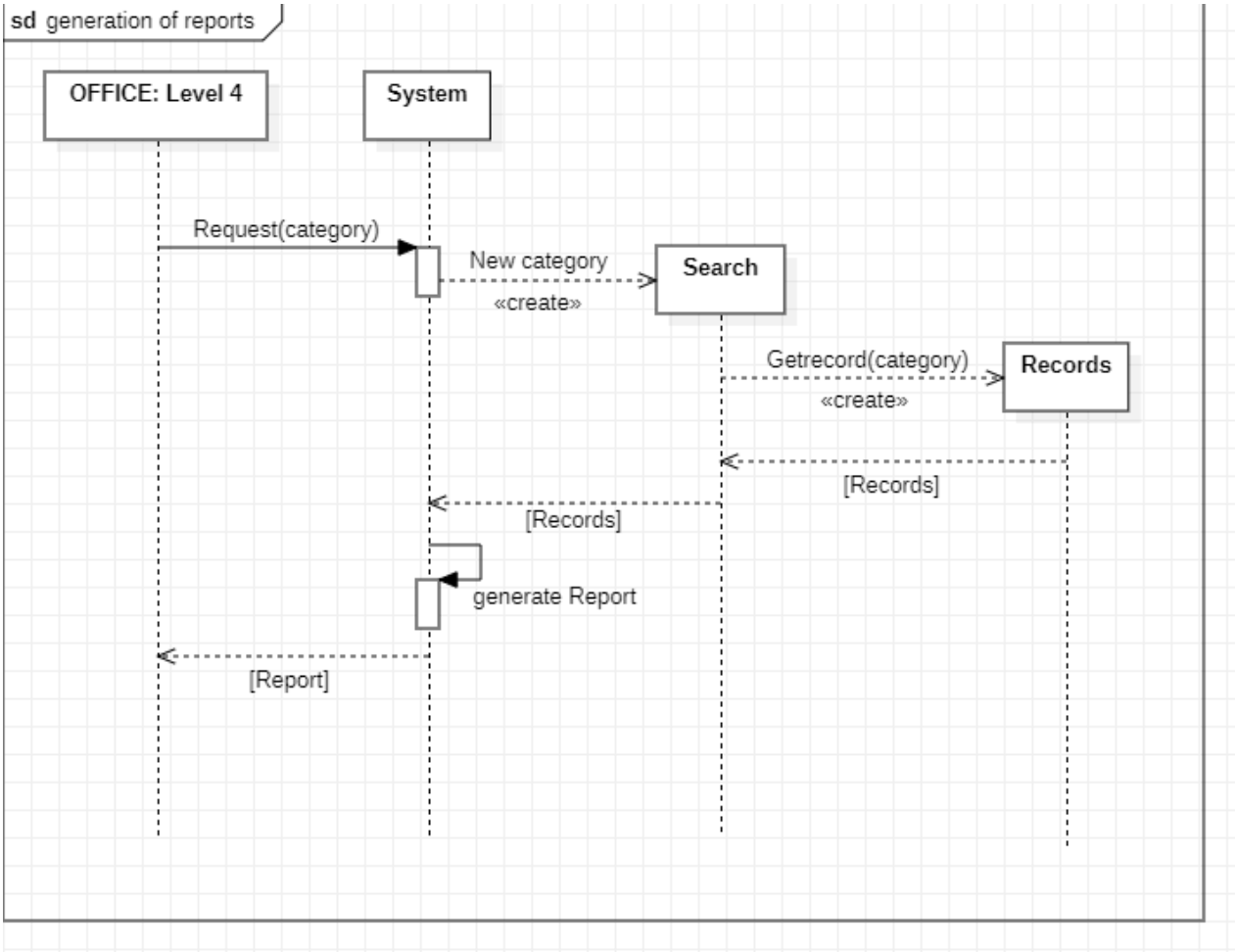
Award record - award name, prize money, id, date

Retirement record – rank during retirement, date, id

Object Model:



Time sequence diagram:



IV. Identification of relations between classes:

Interface:

Login is an interface. It describes the methods required to gain access to the system. The Level3 and Manager class implement Login according to their required access levels. For example, a level3 officer has the ability to add new officers and promote the existing ones but does not have the ability to initiate the retirement process of an officer. It is the exact opposite for the manager.

Inheritance Relationship:

1. Level1 inherits from Personnel with additional attributes which defines the rank and deployment.
2. Level2 inherits from Level1 with the privilege of receiving allowances
3. Level3 inherits from Level2 with the ability to promote junior officers. Any officer above level 3 is considered a senior officer and they are also provided with a login.

4. Level4 inherits from Level3 with the ability to award officers. Level4 officers are allowed to access various reports too.
5. Veteran inherits from Personnel with additional attributes which determines the pension the person receives.
6. Transaction Record, Promotion Record, Retirement Record and Awards Record inherit from Record and each of them are specialized to hold information about their respective processes.
7. Manager class inherits from Login with additional attributes to uniquely identify them.

Static Variables/methods:

1. Personnel, Level1, Veteran and Record classes keep track of their total count through use of static variables.
2. Allowance class makes use of a static method to verify if that particular officer is eligible for the allowance.

Aggregation:

1. Transaction record is aggregated with Payment class.
2. Promotion and Retirement record is aggregated with Level1 class (and its children).
3. Award record is aggregated with Award class.

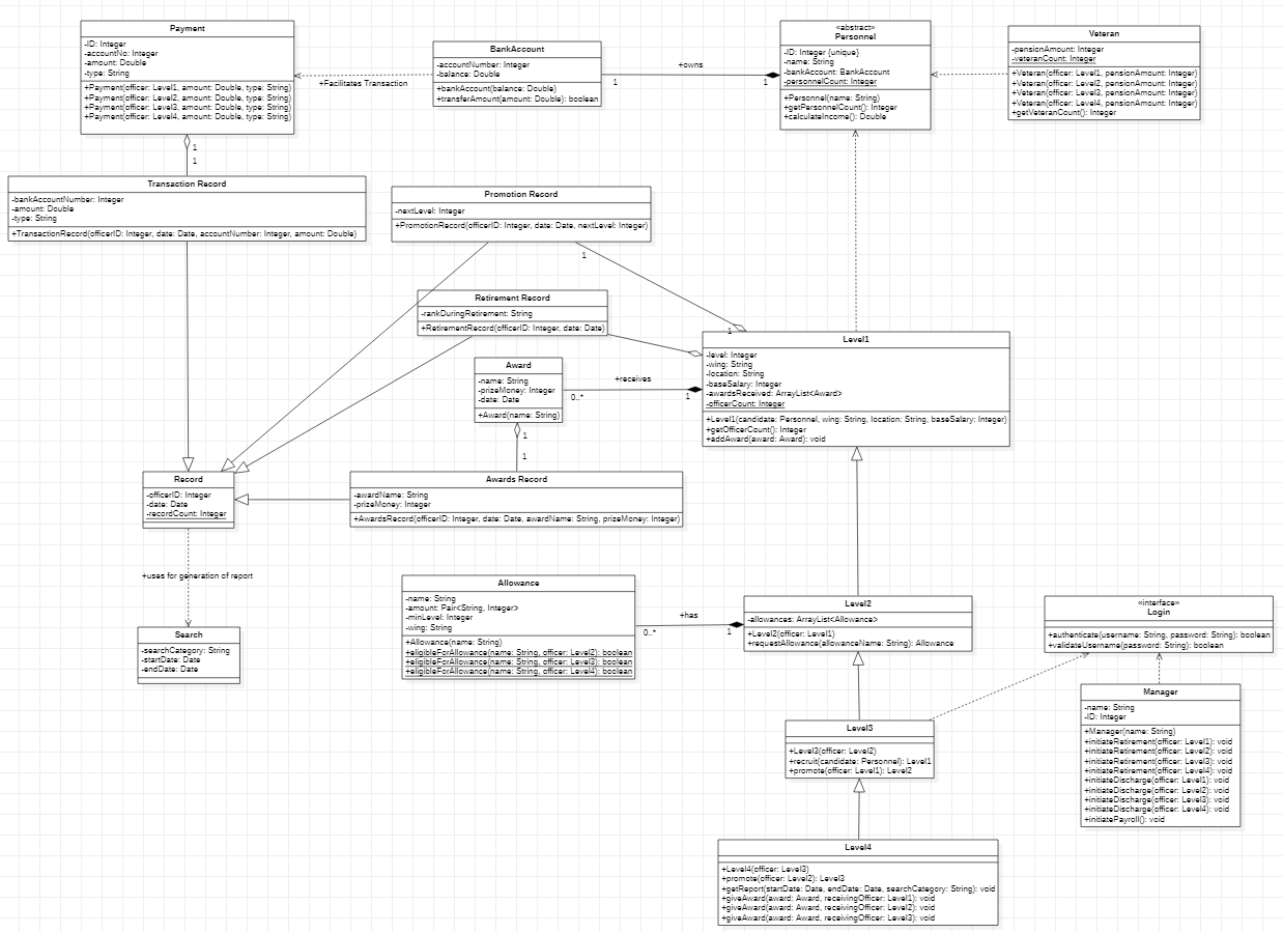
Composition:

1. BankAccount is a part of Personnel class.
2. Award is a part of Level1 class.
3. Allowance is a part of Level2 class.

Dependency:

1. The BankAccount class is dependent on the Payment class for facilitating the transfer of amount. The Payment class is temporarily created and is used as a helper at the time of transaction.
2. The Record class is dependent on Search class for generation of reports. The record class uses the Search class to understand the requirements of the Level4 officer's request.

V. Class Diagram:



VI. Git Repo

Link:- https://github.com/Mystic-Slice/EyeOnArms

VII. Codes

A. Default package

1) Driver class

```
import java.util.Scanner;

import Utility.Allowance;
import Utility.Award;
import Utility.AwardRecord;
import Utility.PromotionRecord;
import Utility.RetirementRecord;
import Utility.TransactionRecord;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.PrintWriter;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

import personnel.Manager;
import personnel.Personnel;
import personnel.Veteran;
import personnel.levelFour;
import personnel.levelOne;
import personnel.levelThree;
import personnel.levelTwo;

public class Driver {

	public static void LoginMenu() {

	System.out.println("**********************************************
*****");
		System.out.println("\t\tLogin Menu\n");
		System.out.println("Choose your designation \n1.Manager
\n2.Level-3 or Level-4 \n3.Exit");

	System.out.println("**********************************************
*****");
	}
```

```
	public static void ManagerView(Manager manager, ArrayList <levelOne>
LevelOne, ArrayList <levelTwo> LevelTwo,
			ArrayList <levelThree> LevelThree, ArrayList <levelFour>
LevelFour, ArrayList<Veteran> veteran,
			ArrayList<RetirementRecord> retirementRecord,
ArrayList<TransactionRecord> transactionRecord) {
		// Menu
		int choice=0;
		char c='N',ch='N';
		Scanner sc = new Scanner (System.in);
		do {
			ch='N';

	System.out.println("*****************************************
**");
			System.out.println("\t\tMenu\n");
			System.out.println("1.Initiate retirement");
			System.out.println("2.Initiate discharge");
			System.out.println("3.Initiate payroll");
			System.out.println("4.Initiate allowance");
			System.out.println("5.Exit");

	System.out.println("*****************************************
**");
			System.out.println("Enter a valid choice");
			choice = sc.nextInt();
			while(choice<1 || choice>5)
			{
				System.out.println("Enter a valid choice: ");
				choice = sc.nextInt();
			}
			switch(choice)
			{
				case 1:
				{
					do {
						c='N';
						int level=0,id=0,i=0;
						boolean flag=false;
```

```
                                        System.out.println("Enter the
ranking level of the officer requesting retirement: ");
                                        level=sc.nextInt();
                                        while(level<=0 || level >=5)
                                        {
                                                System.out.println("Enter
valid level: ");
                                                level=sc.nextInt();
                                        }
                                        System.out.println("Enter the ID of
the officer: ");
                                        id=sc.nextInt();
                                        if(level == 1)
                                        {
                                                for(i=0;i<LevelOne.size();i++)
                                                {

        if(LevelOne.get(i).getID()== id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {

        manager.initiateRetirement(LevelOne.get(i), LevelOne, veteran,
retirementRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                        c=sc.next().charAt(0);
                                                }
                                        }

                                        else if(level == 2)
                                        {
                                                flag=false;
                                                for(i=0;i<LevelTwo.size();i++)
```

```
                                                {

        if(LevelTwo.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {

        manager.initiateRetirement(LevelTwo.get(i), LevelTwo, veteran,
retirementRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                                c=sc.next().charAt(0);
                                                        }
                                                }

                                                else if(level == 3)
                                                {
                                                        flag=false;

        for(i=0;i<LevelThree.size();i++)
                                                        {

        if(LevelThree.get(i).getID() == id)
                                                                {
                                                                        flag=true;
                                                                        break;
                                                                }
                                                        }
                                                        if(flag)
                                                        {

        manager.initiateRetirement(LevelThree.get(i), LevelThree, veteran,
retirementRecord);
                                                        }
```

```
                                        else
                                        {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                c=sc.next().charAt(0);
                                        }
                                }

                                else
                                {
                                        flag=false;
                                        for(i=0;i<LevelFour.size();i++)
                                        {

        if(LevelFour.get(i).getID() == id)
                                                {
                                                        flag=true;
                                                        break;
                                                }
                                        }
                                        if(flag)
                                        {

        manager.initiateRetirement(LevelFour.get(i), LevelFour, veteran,
retirementRecord);
                                        }
                                        else
                                        {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                c=sc.next().charAt(0);
                                        }
                                }

                        }while(c=='Y');

                }
                break;
                case 2:
                {
                        do
```

```
                                        {
                                                c='N';
                                                int level=0,id=0,i=0;
                                                boolean flag=false;
                                                System.out.println("Enter the
ranking level of the officer to be discharged: ");
                                                level=sc.nextInt();
                                                while(level<=0 || level >=5)
                                                {
                                                        System.out.println("Enter
valid level: ");
                                                        level=sc.nextInt();
                                                }
                                                System.out.println("Enter the ID of
the officer: ");
                                                id=sc.nextInt();
                                                if(level == 1)
                                                {
                                                        for(i=0;i<LevelOne.size();i++)
                                                        {

        if(LevelOne.get(i).getID() == id)
                                                                {
                                                                        flag=true;
                                                                        break;
                                                                }
                                                        }
                                                        if(flag)
                                                        {

        manager.initiateDischarge(LevelOne.get(i), LevelOne, veteran,
retirementRecord);
                                                        }
                                                        else
                                                        {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                                c=sc.next().charAt(0);
                                                        }
                                                }
```

```
                                        else if(level == 2)
                                        {
                                                flag=false;
                                                for(i=0;i<LevelTwo.size();i++)
                                                {

        if(LevelTwo.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {

        manager.initiateDischarge(LevelTwo.get(i), LevelTwo, veteran,
retirementRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                        c=sc.next().charAt(0);
                                                }
                                        }

                                        else if(level == 3)
                                        {
                                                flag=false;

        for(i=0;i<LevelThree.size();i++)
                                                {

        if(LevelThree.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {
```

```
        manager.initiateDischarge(LevelThree.get(i), LevelThree, veteran,
retirementRecord);
                                                        }
                                                        else
                                                        {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                                c=sc.next().charAt(0);
                                                        }
                                                }

                                                else
                                                {
                                                        flag=false;
                                                        for(i=0;i<LevelFour.size();i++)
                                                        {

        if(LevelFour.get(i).getID() == id)
                                                                {
                                                                        flag=true;
                                                                        break;
                                                                }
                                                        }
                                                        if(flag)
                                                        {

        manager.initiateDischarge(LevelFour.get(i), LevelFour, veteran,
retirementRecord);
                                                        }
                                                        else
                                                        {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                                c=sc.next().charAt(0);
                                                        }
                                                }

                                        }while(c=='Y');

                                }
```

```
                                        break;
                                        case 3:
                                        {
                                                //Not checking if its first of the month..can
pay only once per month
                                                Date date = new Date();
                                                Integer month = date.getMonth();
                                                if(month != Manager.getMonth())
                                                {
                                                        manager.initiatePayroll(LevelOne,
LevelTwo, LevelThree, LevelFour, veteran, transactionRecord);
                                                }
                                                else
                                                {
                                                        System.out.println("Salary/Pension
paid already for this month");
                                                }
                                        }
                                        break;
                                        case 4:
                                        {
                                                do
                                                {
                                                        c='N';
                                                        int level=0,id=0,i=0;
                                                        boolean flag=false;
                                                        System.out.println("Enter the
ranking level of the officer requesting allowance: ");
                                                        level=sc.nextInt();
                                                        while(level<=1 || level >=5)
                                                        {
                                                                System.out.println("Enter
valid level: ");
                                                                level=sc.nextInt();
                                                        }
                                                        System.out.println("Enter the ID of
the officer: ");
                                                        id=sc.nextInt();

                                                        if(level == 2)
                                                        {
```

```
                                                flag=false;
                                                for(i=0;i<LevelTwo.size();i++)
                                                {

        if(LevelTwo.get(i).getID()== id )
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {
                                                        String
allowanceName;

        System.out.println("Enter the allowance name to claim: ");

        allowanceName=sc.next();

        LevelTwo.get(i).requestAllowance(allowanceName, transactionRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                        c=sc.next().charAt(0);
                                                }
                                        }

                                        else if(level == 3)
                                        {
                                                flag=false;

        for(i=0;i<LevelThree.size();i++)
                                                {

        if(LevelThree.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
```

```
                                                }
                                                if(flag)
                                                {

                                                        String
allowanceName;

        System.out.println("Enter the allowance name to claim: ");

        allowanceName=sc.next();

        LevelThree.get(i).requestAllowance(allowanceName, transactionRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                        c=sc.next().charAt(0);
                                                }
                                        }

                                        else
                                        {
                                                flag=false;
                                                for(i=0;i<LevelFour.size();i++)
                                                {

        if(LevelFour.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {
                                                        String
allowanceName;

        System.out.println("Enter the allowance name to claim: ");

        allowanceName=sc.next();
```

```
LevelFour.get(i).requestAllowance(allowanceName, transactionRecord);
                                                        }
                                                        else
                                                        {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                                c=sc.next().charAt(0);
                                                        }
                                                }

                                        }while(c=='Y');

                                }
                                break;
                                case 5:
                                        break;
                                default:
                                         break;
                        }
                        System.out.println("Do you want to continue(Y/N)?");
                        ch=sc.next().charAt(0);
                }while(ch=='Y');
        }

        public static void Level3View(levelThree LevelThree, ArrayList <levelOne>
LevelOne,  ArrayList <levelTwo> LevelTwo,
                        ArrayList<PromotionRecord> promotionRecord)
        {
                Scanner sc = new Scanner(System.in);
                char ch='N';
                int choice=0;
                do
                {

        System.out.println("**********************************************
**");
                        System.out.println("\t\tMenu\n");
                        System.out.println("1.Change username and password");
                        System.out.println("2.Recruit candidate");
                        System.out.println("3.Promote candidate");
```

```
                        System.out.println("4.Exit");

        System.out.println("****************************************
**");
                        System.out.println("Enter a valid choice");
                        choice = sc.nextInt();
                        while(choice<1 || choice>4)
                        {
                                System.out.println("Enter a valid choice: ");
                                choice = sc.nextInt();
                        }
                        switch(choice)
                        {
                                case 1:
                                {
                                        LevelThree.changeCredentials();

                                }
                                break;
                                case 2:
                                {
                                        String name;
                                        System.out.println("Enter the candinate's
name");
                                        name=sc.next();
                                        Personnel personnel = new Personnel
(name, 0.00);
                                        LevelThree.recruit(personnel, LevelOne,
promotionRecord);
                                }
                                break;
                                case 3:
                                {
                                        char c='N';
                                        do
                                        {
                                                c='N';
                                                int id=0,i;
                                                boolean flag=false;
                                                System.out.println("enter the officer
Id of level 1 to promote: ");
```

```
                                        id=sc.nextInt();
                                        for(i=0;i<LevelOne.size();i++)
                                        {

        if(LevelOne.get(i).getID()==id)
                                                {
                                                        flag=true;
                                                        break;
                                                }
                                        }
                                        if(flag)
                                        {

        LevelThree.promote(LevelOne.get(i), LevelOne, LevelTwo,
promotionRecord);
                                        }
                                        else
                                        {
                                                System.out.println("Invalid
details. \n Do you want to try again (Y/N)");
                                                c=sc.next().charAt(0);
                                        }

                                                }while(c=='Y');
                                        }
                                        break;
                                        case 4:
                                                break;
                                        default:
                                                break;
                                }
                                System.out.println("Do you want to continue(Y/N)");
                                ch=sc.next().charAt(0);

                        }while(ch=='Y');
        }

        public static void Level4View(levelFour LevelFour, ArrayList<levelOne>
LevelOne, ArrayList<levelTwo> LevelTwo,
                                ArrayList<levelThree> LevelThree,
ArrayList<PromotionRecord> promotionRecord,
```

```
                        ArrayList <AwardRecord> awardRecord,
ArrayList<RetirementRecord> retirementRecord,
                        ArrayList<TransactionRecord> transactionRecord)
        {
                Scanner sc = new Scanner(System.in);
                int choice=0;
                char ch='N';

                do
                {

        System.out.println("****************************************
**");
                        System.out.println("\t\tMenu\n");
                        System.out.println("1.Promote candidate");
                        System.out.println("2.Award candidate");
                        System.out.println("3.Generate reports");
                        System.out.println("4.Exit");

        System.out.println("****************************************
**");
                        System.out.println("Enter a valid choice");
                        choice = sc.nextInt();
                        while(choice<1 || choice>4)
                        {
                                System.out.println("Enter a valid choice: ");
                                choice = sc.nextInt();
                        }
                        switch(choice)
                        {
                                case 1:
                                {
                                        char c='N';
                                        do
                                        {
                                                c='N';
                                                int id=0,i=0,level=0;
                                                boolean flag=false;
                                                System.out.println("Enter the level
of officer to promote: ");
                                                level=sc.nextInt();
```

```
                                        while(level<=0 || level>=3)
                                        {
                                                System.out.println("Enter a
valid level");
                                                level=sc.nextInt();

                                        }
                                        if(level == 1)
                                        {
                                                System.out.println("enter
the officer Id to promote: ");
                                                id=sc.nextInt();
                                                for(i=0;i<LevelOne.size();i++)
                                                {

        if(LevelOne.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {

        LevelFour.promote(LevelOne.get(i),LevelOne, LevelTwo,
promotionRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                        c=sc.next().charAt(0);
                                                }
                                        }
                                        else
                                        {
                                                System.out.println("enter
the officer Id to promote: ");
                                                id=sc.nextInt();
                                                for(i=0;i<LevelTwo.size();i++)
                                                {
```

```
        if(LevelTwo.get(i).getID() == id)
                                                        {
                                                                flag=true;
                                                                break;
                                                        }
                                                }
                                                if(flag)
                                                {

        LevelFour.promote(LevelTwo.get(i),LevelTwo ,LevelThree,
promotionRecord);
                                                }
                                                else
                                                {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                        c=sc.next().charAt(0);
                                                }
                                        }

                                }while(c=='Y');

                        }
                        break;
                        case 2:
                        {
                                char c='N';
                                do
                                {
                                        c='N';
                                        int id=0,i=0,level=0;
                                        boolean flag=false;
                                        System.out.println("Enter the level
of officer to award: ");
                                        level=sc.nextInt();
                                        while(level<=0 || level>=4)
                                        {
                                                System.out.println("Enter a
valid level");
```

```
                                        level=sc.nextInt();

                                    }
                                    if(level == 1)
                                    {
                                        System.out.println("enter
the officer Id to award: ");
                                        id=sc.nextInt();
                                        for(i=0;i<LevelOne.size();i++)
                                        {

    if(LevelOne.get(i).getID()== id)
                                            {
                                                flag=true;
                                                break;
                                            }
                                        }
                                        if(flag)
                                        {

    LevelFour.getAward(LevelOne.get(i), awardRecord, transactionRecord);
                                            break;
                                        }
                                        else
                                        {

    System.out.println("Invalid details. Do you want to try again(Y/N)");
                                            c=sc.next().charAt(0);
                                        }
                                    }
                                    else if(level == 2)
                                    {
                                        System.out.println("enter
the officer Id to award: ");
                                        id=sc.nextInt();
                                        for(i=0;i<LevelTwo.size();i++)
                                        {

    if(LevelTwo.get(i).getID() == id)
                                            {
                                                flag=true;
```

```
                                                    break;
                                                }
                                            }
                                            if(flag)
                                            {

        LevelFour.getAward(LevelTwo.get(i), awardRecord, transactionRecord);
                                                    break;
                                            }
                                            else
                                            {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
                                                    c=sc.next().charAt(0);
                                            }
                                        }
                                        else
                                        {
                                            System.out.println("enter
the officer Id to award: ");
                                            id=sc.nextInt();

        for(i=0;i<LevelThree.size();i++)
                                            {

        if(LevelThree.get(i).getID() == id)
                                                {
                                                    flag=true;
                                                    break;
                                                }
                                            }
                                            if(flag)
                                            {

        LevelFour.getAward(LevelThree.get(i),awardRecord, transactionRecord);
                                                    break;
                                            }
                                            else
                                            {

        System.out.println("Invalid details. Do you want to try again(Y/N)");
```

```
                                                c=sc.next().charAt(0);
                                        }
                                }

                        }while(c=='Y');
                }
                break;
                case 3:
                {
                        int searchCategory=0;
                        System.out.println("Enter the start date
from which the report should be generated: ");
                        String sd;
                        sd=sc.next();
                        Date startDate = new Date();
                        try {
                                startDate = new
SimpleDateFormat("dd-MM-yyyy").parse(sd);
                        } catch (ParseException e) {
                                System.out.println("Parse Exception
encountered");
                        }

                        System.out.println("Enter the date till which
the report should be generated: ");
                        String ed;
                        ed=sc.next();
                        Date endDate = new Date();
                        try {
                                endDate = new
SimpleDateFormat("dd-MM-yyyy").parse(ed);
                        } catch (ParseException e) {
                                System.out.println("Parse Exception
encountered");
                        }

                        System.out.println("Enter the search
category\n1. Award report\n2. Transaction report\n3. Retirement report"
                                        + "\n4. Promotion
report\nEnter a valid choice: ");
                        searchCategory = sc.nextInt();
```

```
                                        while(searchCategory<=0 ||
searchCategory>=5)
                                        {
                                                System.out.println("Enter valid
choice: ");
                                                searchCategory=sc.nextInt();
                                        }

                                        LevelFour.getReport(startDate, endDate,
searchCategory, awardRecord, transactionRecord, retirementRecord,
                                                        promotionRecord);
                                }
                                break;
                                case 4:
                                        break;
                                default:
                                        break;
                        }
                        System.out.println("Do you want to continue(Y/N)");
                        ch=sc.next().charAt(0);
                }while(ch=='Y');
        }

        public static void main(String[] args) {
                Allowance.intializeAllowanceMap();
                ArrayList<Manager> manager = new ArrayList<Manager>();
                ArrayList<levelThree> LevelThree = new ArrayList<levelThree>();
                ArrayList<levelFour> LevelFour = new ArrayList<levelFour>();
                ArrayList<levelOne> LevelOne = new ArrayList<levelOne>();
                ArrayList<levelTwo> LevelTwo = new ArrayList<levelTwo>();
                ArrayList<Veteran> veteran = new ArrayList<Veteran>();
                ArrayList<PromotionRecord> promotionRecord = new
ArrayList<PromotionRecord>();
                ArrayList<AwardRecord> awardRecord = new
ArrayList<AwardRecord>();
                ArrayList<RetirementRecord> retirementRecord = new
ArrayList<RetirementRecord>();
                ArrayList<TransactionRecord> transactionRecord = new
ArrayList<TransactionRecord>();
                Scanner sc = new Scanner(System.in);
                /*
```

```
                    * Read from files and put it into arrayLists
                    */
                    Integer totalCount=0, awardCount=0, id, bankAccountNumber,
level,month;
                    String wing, location,temp, name,
awardName,da,username,password, type;
                    Double baseSalary, balance, prizeMoney, amount,
pensionAmount;
                    Date date = new Date();
                    /*
                    * TODO: how to accommodate constant number counts into files
                    */
                    FileReader fileRead;
                    //levelOne
      try
      {
         File directory = new File("");
         //System.out.println(directory.getAbsolutePath()+
"\\Files\\LevelOne.txt");
         fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\LevelOne.txt");
         Scanner fin = new Scanner(fileRead);
         totalCount=fin.nextInt();
         while(totalCount>0)
         {
            ArrayList<Award> award = new ArrayList<Award>();
            temp=fin.next();
            temp=fin.next();
            name=fin.next();
            //System.out.println(name);
            temp=fin.next();
            id=fin.nextInt();
            //System.out.println(id);
            temp=fin.next();
            temp=fin.next();
            temp=fin.next();
            bankAccountNumber=fin.nextInt();
            //System.out.println(bankAccountNumber);
            temp=fin.next();
            temp=fin.next();
            balance=fin.nextDouble();
```

```
            //System.out.println(balance);
            temp=fin.next();
            level=fin.nextInt();
            temp=fin.next();
            wing=fin.next();
            temp=fin.next();
            location=fin.next();
            //System.out.println(location);
            temp=fin.next();
            temp=fin.next();
            baseSalary=fin.nextDouble();
            //System.out.println(baseSalary);
            temp=fin.next();
            awardCount = fin.nextInt();
            while(awardCount>0)
            {
               temp=fin.next();
               awardName=fin.next();
               temp=fin.next();
               temp=fin.next();
               prizeMoney = fin.nextDouble();
               temp=fin.next();
               da=fin.next();
               try
               {
                  date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
               }
               catch (ParseException e)
               {
                  System.out.println("Date not in format as expected");
               }
               Award a = new Award(awardName, prizeMoney, date);
               award.add(a);
               awardCount--;
            }
            levelOne officer = new levelOne(id,name, wing, location, baseSalary,
bankAccountNumber,balance, award);
            LevelOne.add(officer);
            totalCount--;
         }
         fin.close();
```

```
        }
        catch (FileNotFoundException e)
        {
           System.out.println("File not found error");
        }

                    //levelTwo
        try
        {
           File directory = new File("");
           //System.out.println(directory.getAbsolutePath()+
"\\Files\\LevelTwo.txt");
           fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\LevelTwo.txt");
           Scanner fin = new Scanner(fileRead);
           totalCount=fin.nextInt();
           while(totalCount>0)
           {
              ArrayList<Award> award = new ArrayList<Award>();
              temp=fin.next();
              temp=fin.next();
              name=fin.next();
              //System.out.println(name);
              temp=fin.next();
              id=fin.nextInt();
              //System.out.println(id);
              temp=fin.next();
              temp=fin.next();
              temp=fin.next();
              bankAccountNumber=fin.nextInt();
              //System.out.println(bankAccountNumber);
              temp=fin.next();
              temp=fin.next();
              balance=fin.nextDouble();
              //System.out.println(balance);
              temp=fin.next();
              level=fin.nextInt();
              temp=fin.next();
              wing=fin.next();
              temp=fin.next();
              location=fin.next();
```

```
            temp=fin.next();
            temp=fin.next();
            baseSalary=fin.nextDouble();
            temp=fin.next();
            awardCount = fin.nextInt();

            while(awardCount>0)
            {
               temp=fin.next();
               awardName=fin.next();
               temp=fin.next();
               temp=fin.next();
               prizeMoney = fin.nextDouble();
               temp=fin.next();
               da=fin.next();
               try
               {
                  date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
               }
               catch (ParseException e)
               {
                  System.out.println("Date not in format as expected");
               }
               Award a = new Award(awardName, prizeMoney, date);
               award.add(a);
               awardCount--;
            }
            levelTwo officer = new levelTwo(id,name, wing, location, baseSalary,
bankAccountNumber,balance, award);
            LevelTwo.add(officer);
            totalCount--;
         }
         fin.close();
      }
      catch (FileNotFoundException e)
      {
         System.out.println("File not found error");
      }

                  //levelThree
      try
```

```
        {
            File directory = new File("");
            //System.out.println(directory.getAbsolutePath()+
"\\Files\\LevelThree.txt");
            fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\LevelThree.txt");
            Scanner fin = new Scanner(fileRead);
            totalCount=fin.nextInt();
            while(totalCount>0)
            {
               ArrayList<Award> award = new ArrayList<Award>();
               temp=fin.next();
               temp=fin.next();
               name=fin.next();
               //System.out.println(name);
               temp=fin.next();
               id=fin.nextInt();
               //System.out.println(id);
               temp=fin.next();
               username=fin.next();
              //System.out.println(username);
               temp=fin.next();
               password=fin.next();
              //System.out.println(password);
               temp=fin.next();
               temp=fin.next();
               temp=fin.next();
               bankAccountNumber=fin.nextInt();
               //System.out.println(bankAccountNumber);
               temp=fin.next();
               temp=fin.next();
               balance=fin.nextDouble();
               //System.out.println(balance);
               temp=fin.next();
               level=fin.nextInt();
               temp=fin.next();
               wing=fin.next();
               temp=fin.next();
               location=fin.next();
               temp=fin.next();
               temp=fin.next();
```

```
            baseSalary=fin.nextDouble();
            temp=fin.next();
            awardCount = fin.nextInt();

            while(awardCount>0)
            {
               temp=fin.next();
               awardName=fin.next();
               temp=fin.next();
               temp=fin.next();
               prizeMoney = fin.nextDouble();
               temp=fin.next();
               da=fin.next();
               try
               {
                  date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
               }
               catch (ParseException e)
               {
                  System.out.println("Date not in format as expected");
               }
               Award a = new Award(awardName, prizeMoney, date);
               award.add(a);
               awardCount--;
            }
            levelThree officer = new levelThree(id,username, password, name,
wing, location, baseSalary, bankAccountNumber, balance, award);
            LevelThree.add(officer);
            totalCount--;
         }
         fin.close();
      }
      catch (FileNotFoundException e)
      {
         System.out.println("File not found error");
      }


                  //levelFour
      try
      {
```

```
        File directory = new File("");
        //System.out.println(directory.getAbsolutePath()+
"\\Files\\LevelFour.txt");
        fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\LevelFour.txt");
        Scanner fin = new Scanner(fileRead);
        totalCount=fin.nextInt();
        while(totalCount>0)
        {
           ArrayList<Award> award = new ArrayList<Award>();
           temp=fin.next();
           temp=fin.next();
           name=fin.next();
           //System.out.println(name);
           temp=fin.next();
           id=fin.nextInt();
           //System.out.println(id);
           temp=fin.next();
           username=fin.next();
          //System.out.println(username);
           temp=fin.next();
           password=fin.next();
          //System.out.println(password);
           temp=fin.next();
           temp=fin.next();
           temp=fin.next();
           bankAccountNumber=fin.nextInt();
           //System.out.println(bankAccountNumber);
           temp=fin.next();
           temp=fin.next();
           balance=fin.nextDouble();
           //System.out.println(balance);
           temp=fin.next();
           level=fin.nextInt();
           temp=fin.next();
           wing=fin.next();
           temp=fin.next();
           location=fin.next();
           temp=fin.next();
           temp=fin.next();
           baseSalary=fin.nextDouble();
```

```
                temp=fin.next();
                awardCount = fin.nextInt();

                while(awardCount>0)
                {
                   temp=fin.next();
                   awardName=fin.next();
                   temp=fin.next();
                   temp=fin.next();
                   prizeMoney = fin.nextDouble();
                   temp=fin.next();
                   da=fin.next();
                   try
                   {
                      date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
                   }
                   catch (ParseException e)
                   {
                      System.out.println("Date not in format as expected");
                   }
                   Award a = new Award(awardName, prizeMoney, date);
                   award.add(a);
                   awardCount--;
                }
                levelFour officer = new levelFour(id, username, password, name, wing,
location, baseSalary, bankAccountNumber, balance, award);
                LevelFour.add(officer);
                totalCount--;
             }
             fin.close();
          }
          catch (FileNotFoundException e)
          {
             System.out.println("File not found error");
          }

          //Transaction Record
          try
          {
             File directory = new File("");
```

```
        //System.out.println(directory.getAbsolutePath()+
"\\Files\\TransactionRecord.txt");
        fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\TransactionRecord.txt");
        Scanner fin = new Scanner(fileRead);
        totalCount=fin.nextInt();
        while(totalCount>0)
        {
         temp=fin.next();
           temp=fin.next();
           temp=fin.next();
           id=fin.nextInt();
           //System.out.println(id);
           temp=fin.next();
           da=fin.next();
           try
           {
              date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
           }
           catch (ParseException e)
           {
              System.out.println("Date not in format as expected");
           }
           //System.out.println(date);
           temp=fin.next();
           temp=fin.next();
           temp=fin.next();
           bankAccountNumber = fin.nextInt();
           //System.out.println(bankAccountNumber);
           temp=fin.next();
           amount=fin.nextDouble();
           //System.out.println(amount);
           temp=fin.next();
           type=fin.next();
           //System.out.println(type);
           TransactionRecord tr = new
TransactionRecord(id,date,bankAccountNumber,amount,type);
           transactionRecord.add(tr);
           totalCount--;
        }
        fin.close();
```

```
        }
        catch (FileNotFoundException e)
        {
           System.out.println("File not found error");
        }


        //Promotion Record

        try
        {
           File directory = new File("");
           //System.out.println(directory.getAbsolutePath()+
"\\Files\\PromotionRecord.txt");
           fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\PromotionRecord.txt");
           Scanner fin = new Scanner(fileRead);
           totalCount=fin.nextInt();
           while(totalCount>0)
           {
              temp=fin.next();
              temp=fin.next();
              temp=fin.next();
              id=fin.nextInt();
              //System.out.println(id);
              temp=fin.next();
              da=fin.next();
              try
              {
                 date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
              }
              catch (ParseException e)
              {
                 System.out.println("Date not in format as expected");
              }
              //System.out.println(date);
              temp=fin.next();
              temp=fin.next();
              level=fin.nextInt();
              //System.out.println(level);
              PromotionRecord pr = new PromotionRecord(id,date,level);
```

```
            promotionRecord.add(pr);
            totalCount--;
          }
          fin.close();
        }
        catch (FileNotFoundException e)
        {
          System.out.println("File not found error");
        }

        //Retirement Record

        try
        {
          File directory = new File("");
          //System.out.println(directory.getAbsolutePath()+
"\\Files\\RetirementRecord.txt");
          fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\RetirementRecord.txt");
          Scanner fin = new Scanner(fileRead);
          totalCount=fin.nextInt();
          while(totalCount>0)
          {
            temp=fin.next();
            temp=fin.next();
            temp=fin.next();
            id=fin.nextInt();
            //System.out.println(id);
            temp=fin.next();
            da=fin.next();
            try
            {
              date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
            }
            catch (ParseException e)
            {
              System.out.println("Date not in format as expected");
            }
            //System.out.println(date);
            temp=fin.next();
            temp=fin.next();
```

```
            temp=fin.next();
            level=fin.nextInt();
            //System.out.println(level);
            RetirementRecord rr = new RetirementRecord(id,date,level);
            retirementRecord.add(rr);
            totalCount--;
          }
          fin.close();
        }
        catch (FileNotFoundException e)
        {
          System.out.println("File not found error");
        }

        //Awards Record
        try
        {
          File directory = new File("");
          //System.out.println(directory.getAbsolutePath()+
"\\Files\\AwardRecord.txt");
          fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\AwardRecord.txt");
          Scanner fin = new Scanner(fileRead);
          totalCount=fin.nextInt();
          while(totalCount>0)
          {
            temp=fin.next();
            temp=fin.next();
            temp=fin.next();
            id=fin.nextInt();
            //System.out.println(id);
            temp=fin.next();
            da=fin.next();
            try
            {
              date=new SimpleDateFormat("dd-MM-yyyy").parse(da);
            }
            catch (ParseException e)
            {
              System.out.println("Date not in format as expected");
            }
```

```
            //System.out.println(date);
            temp=fin.next();
            temp=fin.next();
            awardName=fin.next();
            //System.out.println(awardName);
            temp=fin.next();
            temp=fin.next();
            prizeMoney=fin.nextDouble();
            //System.out.println(prizeMoney);
            AwardRecord ar = new AwardRecord(id,date,awardName, prizeMoney);
            awardRecord.add(ar);
            totalCount--;
         }
         fin.close();
      }
      catch (FileNotFoundException e)
      {
         System.out.println("File not found error");
      }

      //Manager

      try
      {
         File directory = new File("");
         //System.out.println(directory.getAbsolutePath()+
"\\Files\\Manager.txt");
         fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\Manager.txt");
         Scanner fin = new Scanner(fileRead);
         totalCount=fin.nextInt();
         while(totalCount>0)
         {
          temp=fin.next();
            temp=fin.next();
            name=fin.next();
            //System.out.println(name);
            temp=fin.next();
            id=fin.nextInt();
            //System.out.println(id);
            temp=fin.next();
```

```
            username=fin.next();
            //System.out.println(username);
            temp=fin.next();
            password=fin.next();
            //System.out.println(password);

            Manager m = new Manager(name, id, username, password);
            manager.add(m);
            totalCount--;
          }
          temp=fin.next();
          month=fin.nextInt();
          Manager.setMonth(month);
          fin.close();
        }
        catch (FileNotFoundException e)
        {
          System.out.println("File not found error");
        }


        //Veteran
        try
        {
          File directory = new File("");
          //System.out.println(directory.getAbsolutePath()+ "\\Files\\Veteran.txt");
          fileRead = new FileReader(directory.getAbsolutePath()+
"\\Files\\Veteran.txt");
          Scanner fin = new Scanner(fileRead);
          totalCount=fin.nextInt();
          while(totalCount>0)
          {
           temp=fin.next();
            temp=fin.next();
            name=fin.next();
            //System.out.println(name);
            temp=fin.next();
            id=fin.nextInt();
            //System.out.println(id);
            temp=fin.next();
            temp=fin.next();
```

```
            temp=fin.next();
            bankAccountNumber=fin.nextInt();
            //System.out.println(bankAccountNumber);
            temp=fin.next();
            temp=fin.next();
            balance=fin.nextDouble();
            //System.out.println(balance);
            temp=fin.next();
            temp=fin.next();
            pensionAmount=fin.nextDouble();
            //System.out.println(balance);
            Veteran v = new Veteran(name, id, bankAccountNumber,balance,
pensionAmount);
            veteran.add(v);
            totalCount--;
          }
          fin.close();
      }
      catch (FileNotFoundException e)
      {
        System.out.println("File not found error");
      }

                int choice=-1;
                char ch='N',c='N',check='N';
                do
                {
                        Driver.LoginMenu();
                        choice=sc.nextInt();
                        while(choice<=0 || choice>=4){
                                System.out.println("Enter a valid choice: ");
                                choice = sc.nextInt();
                        }
                        switch(choice) {
                                case 1:
                                {
                                        int i=0;
                                        do
                                        {
                                                c='N';
```

```
                                        System.out.println("Enter your
username: ");
                                        username=sc.next();
                                        System.out.println("Enter your
password: ");
                                        password=sc.next();
                                        boolean flag=false;
                                        for(i=0;i<manager.size();i++)
                                        {


        if(manager.get(i).validate(username, password))
                                                {

                                                        flag=true;
                                                        break;
                                                }
                                        }
                                        if(flag) {
                                                System.out.println("Logged
in successfully!");

        Driver.ManagerView(manager.get(i), LevelOne, LevelTwo, LevelThree,
LevelFour, veteran,

        retirementRecord, transactionRecord);
                                                break;
                                        }
                                        else {
                                                System.out.println("Invalid
credentials\n Do you want to try again(Y/N)");
                                                c=sc.next().charAt(0);

                                                }
                                        }while(c=='Y');

                                }
                                break;
                                case 2:
                                {
                                        do
```

```
                                        {
                                                c='N';
                                                int i=0;
                                                System.out.println("Enter your
username: ");
                                                username=sc.next();
                                                System.out.println("Enter your
password: ");
                                                password=sc.next();
                                                System.out.println("Enter your
ranking level: ");
                                                level=sc.nextInt();
                                                if(level==3)
                                                {
                                                        boolean flag=false;

        for(i=0;i<LevelThree.size();i++)
                                                        {


        if(LevelThree.get(i).validate(username, password))
                                                                {

                                                                        flag=true;
                                                                        break;
                                                                }
                                                        }
                                                        if(flag) {

        System.out.println("Logged in successfully!");

        Driver.Level3View(LevelThree.get(i), LevelOne, LevelTwo,
promotionRecord);
                                                                break;
                                                        }
                                                        else {

        System.out.println("Invalid credentials\n Do you want to try again(Y/N)");
                                                                c=sc.next().charAt(0);
```

```
                }

            }
            if(level==4)
            {
                boolean flag=false;
                for(i=0;i<LevelFour.size();i++)
                {


        if(LevelFour.get(i).validate(username, password))
                        {

                            flag=true;
                            break;
                        }
                    }
                    if(flag) {

        System.out.println("Logged in successfully!");

        Driver.Level4View(LevelFour.get(i), LevelOne, LevelTwo, LevelThree,
promotionRecord,

        awardRecord, retirementRecord, transactionRecord);
                        break;

                    }
                    else {

        System.out.println("Invalid credentials\n Do you want to try again(Y/N)");
                        c=sc.next().charAt(0);

                    }

            }

        }while(c=='Y');

    }
    break;
```

```
                        case 3:
                                break;
                        default:
                                break;
                }
                System.out.println("Do you want to log in(Y/N)");
                check=sc.next().charAt(0);
        }while(check=='Y');

        /*
         * Write into the files
         *
         * PrintWriter fout = new PrintWriter("output.txt");
                fout.println(29.95);
                fout.println(new Rectangle(5, 10, 15, 25));
                fout.println("Hello, World!");
         */

        //Writing into levelOne officer files:
        int i=0;
        PrintWriter fout;
        try {
                File directory = new File("");
                fout = new PrintWriter(directory.getAbsolutePath()+
"\\Files\\LevelOne.txt");
                fout.println(LevelOne.size());
                fout.println("----------------------------------------");

                for(i=0;i<LevelOne.size();i++)
                {
                        fout.println("Name: "+LevelOne.get(i).getName());
                        fout.println("ID: "+LevelOne.get(i).getID());
                        fout.println("Bank account number:
"+LevelOne.get(i).bankaccount.getAccountNumber());
                        fout.println("Bank balance:
"+LevelOne.get(i).getBalance());
                        fout.println("Level: "+LevelOne.get(i).getLevel());
                        fout.println("Wing: "+LevelOne.get(i).getWing());
                        fout.println("Location:
"+LevelOne.get(i).getLocation());
```

```
                                fout.println("Base salary:
"+LevelOne.get(i).getBaseSalary());
                                fout.println("Awards: ");
                                fout.println(LevelOne.get(i).awardsRecieved.size());
                                for(int
j=0;j<LevelOne.get(i).awardsRecieved.size();j++)
                                { //print awards received number
                                        if(j!=LevelOne.get(i).awardsRecieved.size()-
1)
                                        {
                                                fout.println("Name:
"+LevelOne.get(i).awardsRecieved.get(j).getName());
                                                fout.println("Prize money:
"+LevelOne.get(i).awardsRecieved.get(j).getPrizeMoney());
                                                fout.println("Date:
"+LevelOne.get(i).awardsRecieved.get(j).getStringDate());
                                        }
                                        else
                                        {
                                                fout.println("Name:
"+LevelOne.get(i).awardsRecieved.get(j).getName());
                                                fout.println("Prize money:
"+LevelOne.get(i).awardsRecieved.get(j).getPrizeMoney());
                                                fout.println("Date:
"+LevelOne.get(i).awardsRecieved.get(j).getStringDate());
                                                fout.println("----------------------------
----------");
                                        }
                                }
                        }
                        fout.close();
                } catch (FileNotFoundException e) {

                        System.out.println("File not found error");
                }

                try {
                        File directory = new File("");
                        fout = new PrintWriter(directory.getAbsolutePath()+
"\\Files\\LevelTwo.txt");
                        fout.println(LevelTwo.size());
```

```
                        fout.println("----------------------------------------");
                        for(i=0;i<LevelTwo.size();i++)
                        {
                                fout.println("Name: "+LevelTwo.get(i).getName());
                                fout.println("ID: "+LevelTwo.get(i).getID());
                                fout.println("Bank account number:
"+LevelTwo.get(i).bankaccount.getAccountNumber());
                                fout.println("Bank balance:
"+LevelTwo.get(i).getBalance());
                                fout.println("Level: "+LevelTwo.get(i).getLevel());
                                fout.println("Wing: "+LevelTwo.get(i).getWing());
                                fout.println("Location:
"+LevelTwo.get(i).getLocation());
                                fout.println("Base salary:
"+LevelTwo.get(i).getBaseSalary());
                                fout.println("Awards: ");
                                fout.println(LevelTwo.get(i).awardsRecieved.size());
                                for(int
j=0;j<LevelTwo.get(i).awardsRecieved.size();j++)
                                {
                                        if(j!=LevelTwo.get(i).awardsRecieved.size()-
1)
                                        {
                                                fout.println("Name:
"+LevelTwo.get(i).awardsRecieved.get(j).getName());
                                                fout.println("Prize money:
"+LevelTwo.get(i).awardsRecieved.get(j).getPrizeMoney());
                                                fout.println("Date:
"+LevelTwo.get(i).awardsRecieved.get(j).getStringDate());
                                        }
                                        else
                                        {
                                                fout.println("Name:
"+LevelTwo.get(i).awardsRecieved.get(j).getName());
                                                fout.println("Prize money:
"+LevelTwo.get(i).awardsRecieved.get(j).getPrizeMoney());
                                                fout.println("Date:
"+LevelTwo.get(i).awardsRecieved.get(j).getStringDate());
                                                fout.println("----------------------------
----------");
                                        }
```

```
                        }
                    }
                    fout.close();
                } catch (FileNotFoundException e) {
                    System.out.println("File not found error");
                }

                try {
                    File directory = new File("");
                    fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\LevelThree.txt");
                    fout.println(LevelThree.size());
                    fout.println("----------------------------------------");
                    for(i=0;i<LevelThree.size();i++)
                    {
                        fout.println("Name:
"+LevelThree.get(i).getName());
                        fout.println("ID: "+LevelThree.get(i).getID());
                        fout.println("Username:
"+LevelThree.get(i).getUsername());
                        fout.println("Password:
"+LevelThree.get(i).getPassword());
                        fout.println("Bank account number:
"+LevelThree.get(i).bankaccount.getAccountNumber());
                        fout.println("Bank balance:
"+LevelThree.get(i).getBalance());
                        fout.println("Level: "+LevelThree.get(i).getLevel());
                        fout.println("Wing: "+LevelThree.get(i).getWing());
                        fout.println("Location:
"+LevelThree.get(i).getLocation());
                        fout.println("Base salary:
"+LevelThree.get(i).getBaseSalary());
                        fout.println("Awards: ");

        fout.println(LevelThree.get(i).awardsRecieved.size());
                        for(int
j=0;j<LevelThree.get(i).awardsRecieved.size();j++)
                        {

        if(j!=LevelThree.get(i).awardsRecieved.size()-1)
                            {
```

```
                                        fout.println("Name:
"+LevelThree.get(i).awardsRecieved.get(j).getName());
                                        fout.println("Prize money:
"+LevelThree.get(i).awardsRecieved.get(j).getPrizeMoney());
                                        fout.println("Date:
"+LevelThree.get(i).awardsRecieved.get(j).getStringDate());
                                    }
                                    else
                                    {
                                        fout.println("Name:
"+LevelThree.get(i).awardsRecieved.get(j).getName());
                                        fout.println("Prize money:
"+LevelThree.get(i).awardsRecieved.get(j).getPrizeMoney());
                                        fout.println("Date:
"+LevelThree.get(i).awardsRecieved.get(j).getStringDate());
                                        fout.println("----------------------------
----------");
                                    }
                                }
                        }
                        fout.close();
                } catch (FileNotFoundException e) {
                        System.out.println("File not found error");

                }

                try {
                        File directory = new File("");
                        fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\LevelFour.txt");
                        fout.println(LevelFour.size());
                        fout.println("----------------------------------------");
                        for(i=0;i<LevelFour.size();i++)
                        {
                                fout.println("Name: "+LevelFour.get(i).getName());
                                fout.println("ID: "+LevelFour.get(i).getID());
                                fout.println("Username:
"+LevelFour.get(i).getUsername());
                                fout.println("Password:
"+LevelFour.get(i).getPassword());
```

```
                                fout.println("Bank account number:
"+LevelFour.get(i).bankaccount.getAccountNumber());
                                fout.println("Bank balance:
"+LevelFour.get(i).getBalance());
                                fout.println("Level: "+LevelFour.get(i).getLevel());
                                fout.println("Wing: "+LevelFour.get(i).getWing());
                                fout.println("Location:
"+LevelFour.get(i).getLocation());
                                fout.println("Base salary:
"+LevelFour.get(i).getBaseSalary());
                                fout.println("Awards: ");

        fout.println(LevelFour.get(i).awardsRecieved.size());
                                for(int
j=0;j<LevelFour.get(i).awardsRecieved.size();j++)
                                {
                                        if(j!=LevelFour.get(i).awardsRecieved.size()-
1)
                                        {
                                                fout.println("Name:
"+LevelFour.get(i).awardsRecieved.get(j).getName());
                                                fout.println("Prize money:
"+LevelFour.get(i).awardsRecieved.get(j).getPrizeMoney());
                                                fout.println("Date:
"+LevelFour.get(i).awardsRecieved.get(j).getStringDate());
                                        }
                                        else
                                        {
                                                fout.println("Name:
"+LevelFour.get(i).awardsRecieved.get(j).getName());
                                                fout.println("Prize money:
"+LevelFour.get(i).awardsRecieved.get(j).getPrizeMoney());
                                                fout.println("Date:
"+LevelFour.get(i).awardsRecieved.get(j).getStringDate());
                                                fout.println("----------
----------");
                                        }
                                }
                                /*
                                 *TODO: Do we need to print allowances for each
person?
```

```
                                */

                                }
                                fout.close();
                        } catch (FileNotFoundException e) {
                                System.out.println("File not found error");
                        }

                        try {
                                File directory = new File("");
                                fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\AwardRecord.txt");
                                fout.println(awardRecord.size());
                                fout.println("----------------------------------------");
                                for(i=0;i<awardRecord.size();i++)
                                {
                                        fout.println("Officer ID:
"+awardRecord.get(i).getOfficerID());
                                        fout.println("Date:
"+awardRecord.get(i).getStringDate());
                                        fout.println("Award name:
"+awardRecord.get(i).getAwardName());
                                        fout.println("Prize money:
"+awardRecord.get(i).getPrizeMoney());
                                        fout.println("----------------------------------------");
                                }
                                fout.close();
                        } catch (FileNotFoundException e) {
                                System.out.println("File not found error");
                        }


                        try {
                                File directory = new File("");
                                fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\PromotionRecord.txt");
                                fout.println(promotionRecord.size());
                                fout.println("----------------------------------------");
                                for(i=0;i<promotionRecord.size();i++)
                                {
```

```
                                        fout.println("Officer ID:
"+promotionRecord.get(i).getOfficerID());
                                        fout.println("Date:
"+promotionRecord.get(i).getStringDate());
                                        fout.println("Next level:
"+promotionRecord.get(i).getNextLevel());
                                        fout.println("----------------------------------------");
                                }
                                fout.close();
                        } catch (FileNotFoundException e) {
                                System.out.println("File not found error");
                        }

                        try {
                                File directory = new File("");
                                fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\RetirementRecord.txt");
                                fout.println(retirementRecord.size());
                                fout.println("----------------------------------------");
                                for(i=0;i<retirementRecord.size();i++)
                                {
                                        fout.println("Officer ID:
"+retirementRecord.get(i).getOfficerID());
                                        fout.println("Date:
"+retirementRecord.get(i).getStringDate());
                                        fout.println("Level during retirement:
"+retirementRecord.get(i).getRankDuringRetirement());
                                        fout.println("----------------------------------------");
                                }
                                fout.close();
                        } catch (FileNotFoundException e) {
                                System.out.println("File not found error");
                        }

                        try {
                                File directory = new File("");
                                fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\TransactionRecord.txt");
                                fout.println(transactionRecord.size());
                                fout.println("----------------------------------------");
                                for(i=0;i<transactionRecord.size();i++)
```

```
                        {
                                fout.println("Officer ID:
"+transactionRecord.get(i).getOfficerID());
                                fout.println("Date:
"+transactionRecord.get(i).getStringDate());
                                fout.println("Bank account number:
"+transactionRecord.get(i).getBankAccountNumber());
                                fout.println("Amount:
"+transactionRecord.get(i).getAmount());
                                fout.println("Type:
"+transactionRecord.get(i).getType());
                                fout.println("----------------------------------------");
                        }
                        fout.close();
                } catch (FileNotFoundException e) {
                        System.out.println("File not found error");
                }

                //Manager

                try {
                        File directory = new File("");
                        fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\Manager.txt");
                        fout.println(manager.size());
                        fout.println("----------------------------------------");
                        for(i=0;i<manager.size();i++)
                        {
                                fout.println("Name: "+manager.get(i).getName());
                                fout.println("ID: "+manager.get(i).getID());
                                fout.println("Username:
"+manager.get(i).getUsername());
                                fout.println("Password:
"+manager.get(i).getPassword());
                                fout.println("----------------------------------------");
                        }
                        fout.println(Manager.getMonth());
                        fout.close();
                } catch (FileNotFoundException e) {
                        System.out.println("File not found error");
                }
```

```
                    //Veteran
                    try {
                            File directory = new File("");
                            fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\Veteran.txt");
                            fout.println(veteran.size());
                            fout.println("----------------------------------------");
                            for(i=0;i<veteran.size();i++)
                            {
                                    fout.println("Name: "+veteran.get(i).getName());
                                    fout.println("ID: "+veteran.get(i).getID());
                                    fout.println("Bank account number:
"+veteran.get(i).bankaccount.getAccountNumber());
                                    fout.println("Bank balance:
"+veteran.get(i).getBalance());
                                    fout.println("Pension amount:
"+veteran.get(i).getPensionAmount());
                                    fout.println("----------------------------------------");
                            }
                            fout.close();
                    } catch (FileNotFoundException e) {
                            System.out.println("File not found error");
                    }
            }
}
```

B. Personnel package

1) Personnel class

```
package personnel;

import java.util.Scanner;
import Utility.BankAccount;

public class Personnel {
        Scanner sc=new Scanner(System.in);
```

```
	public Integer  id;
	public String name;
	public BankAccount bankaccount;
	//give bank account belonging to the class bank account
	public static int PersonnelCount=0;
	public Personnel(Integer id, String name, BankAccount bankaccount)
//for promotion
	{
		this.id=id;
		this.name=name;
		this.bankaccount=bankaccount;
	}
	public Personnel(Integer id, String name, Integer bid, Double balance) //
from reading file
	{
		this.id=id;
		this.name=name;
		this.bankaccount=new BankAccount(id, balance);
		PersonnelCount++;
	}
	public Personnel(String name, Double balance) // creating new officer
	{
		this.name=name;
		this.id=++PersonnelCount;
		this.bankaccount=new BankAccount(balance);
	}
	public static int GetPersonnelCount() {
		return PersonnelCount;
	}

	public Double getBalance()
	{
		return this.bankaccount.getBalance();
	}
	public int getID() {
		return this.id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
```

```
		this.name = name;
	}
	public BankAccount getBankaccount() {
		return bankaccount;
	}
	public void setBankaccount(BankAccount bankaccount) {
		this.bankaccount = bankaccount;
	}
	public static int getPersonnelCount() {
		return PersonnelCount;
	}
	public static void setPersonnelCount(int personnelCount) {
		PersonnelCount = personnelCount;
	}



}
```

2) levelOne class

```
package personnel;


import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import Utility.Award;

import Utility.BankAccount;


public class levelOne extends Personnel {

	public Integer level;

	public String wing;

	public String location;

	public Double BaseSalary;

	public ArrayList<Award>awardsRecieved = new ArrayList<Award>();
```

```
	public static Integer OfficerCount=0;
	public static Integer GetOfficerCount() {
		return OfficerCount;
	}
	public levelOne(Integer id, String name,String wing,String location,Double BaseSalary,
BankAccount bankaccount){ //For recruiting
		super(id, name,bankaccount);
		this.level=1;
		this.wing=wing;
		this.location=location;
		this.BaseSalary=BaseSalary;
		this.awardsRecieved=new ArrayList<Award>();
		OfficerCount++;
	}
	public levelOne(Integer id, String name,String wing,String location,Double BaseSalary, Integer
bid, Double balance, ArrayList<Award> awards){//from files
		super(id,name,bid,balance);
		this.level=1;
		this.wing=wing;
		this.location=location;
		this.BaseSalary=BaseSalary;
		this.awardsRecieved=awards;
		OfficerCount++;
	}


	public Award AddAward() throws Exception{
		String aname;
		Double pmoney;
```

```
            System.out.println("enter the award name");
            aname=sc.next();
            System.out.println("enter the prize money");
            pmoney=sc.nextDouble();
            this.bankaccount.setBalance(this.bankaccount.getBalance()+pmoney);
            Date d=new Date();
            Award dw=new Award(aname,pmoney,d);
            awardsRecieved.add(dw);
            return dw;
    }
    /*
     * How to implement calculateIncome
     */

    double CalculateIncome() {
            return this.BaseSalary;
    }

    public Integer getLevel() {
            return level;
    }
    public void setLevel(Integer level) {
            this.level = level;
    }
    public String getWing() {
            return wing;
    }
    public void setWing(String wing) {
            this.wing = wing;
```

```
	}
	public String getLocation() {
		return location;
	}
	public void setLocation(String location) {
		this.location = location;
	}
	public Double getBaseSalary() {
		return BaseSalary;
	}
	public void setBaseSalary(Double baseSalary) {
		BaseSalary = baseSalary;
	}
	public ArrayList<Award> getAwardsRecieved() {
		return awardsRecieved;
	}
	public void setAwardsRecieved(ArrayList<Award> awardsRecieved) {
		this.awardsRecieved = awardsRecieved;
	}
	public static Integer getOfficerCount() {
		return OfficerCount;
	}
	public static void setOfficerCount(Integer officerCount) {
		OfficerCount = officerCount;
	}


}
```

3) levelTwo class

```
package personnel;


import java.util.ArrayList;

import Utility.Allowance;

import Utility.Award;

import Utility.BankAccount;

import Utility.TransactionRecord;


public class levelTwo extends levelOne {

	public ArrayList<Allowance>allowance;

	public levelTwo(Integer id,String name,String wing,String location,Double BaseSalary,
BankAccount bankAccount, ArrayList<Award> award){//promoting

		super(id,name,wing,location,BaseSalary,bankAccount);

		this.awardsRecieved=award;

		this.level=2;

		allowance = new ArrayList<Allowance>();

	}

	public levelTwo(Integer id,String name,String wing,String location,Double BaseSalary, Integer
bid, Double balance, ArrayList<Award> award){//from files

		super(id,name,wing,location,BaseSalary,bid,balance,award);

		this.level=2;

		allowance = new ArrayList<Allowance>();

	}

	public void requestAllowance(String allowancename, ArrayList<TransactionRecord>
transactionRecord) {

		/*

		 * TODO: check request allowance function

		 */

		boolean flag= false;
```

```
                flag=Allowance.eligibleforAllowance(allowancename, this, transactionRecord);
                if(flag)
                {
                        System.out.println("Allowance claimed");
                }
                else
                {
                        System.out.println("Officer not eligible for allowance");
                }
        }


}
```

4) levelThree class

```
package personnel;

import java.util.ArrayList;
import java.util.Date;
import java.util.Scanner;

import Utility.Award;
import Utility.BankAccount;
import Utility.Login;
import Utility.PromotionRecord;

public class levelThree extends levelTwo implements Login {
        private String username,password;
        /*
         * TODO: should user name and password be private or protected?
         */
        public levelThree(Integer id, String username, String password, String
name,String wing,String location,
                        Double BaseSalary, BankAccount bankAccount, ArrayList<Award>
award){ //for promotion
                super(id,name,wing,location,BaseSalary,bankAccount,award);
                this.username = username;
                this.password = password;
                this.level=3;
        }
        public levelThree(Integer id, String username, String password, String
name,String wing,String location,
                        Double BaseSalary, Integer bid, Double balance, ArrayList<Award>
award){ //for promotion
```

```
            super(id,name,wing,location,BaseSalary,bid,balance,award);
            this.username = username;
            this.password = password;
            this.level=3;
      }
      public void recruit(Personnel candidate, ArrayList<levelOne> LevelOne,
ArrayList<PromotionRecord> promotionRecord) {
            String w;
            System.out.println("enter the wing name");
            w=sc.next();
            String l;
            System.out.println("enter the location");
            l=sc.next();
            /*
             * TODO: Check logic here
             */
            levelOne officer=new
levelOne(candidate.id,candidate.name,w,l,50000.00,candidate.bankaccount);
            LevelOne.add(officer);
//          Date date = new Date();
//          PromotionRecord prObject = new PromotionRecord(officer.getID(), date,
1);
//          promotionRecord.add(prObject);
            System.out.println("Officer: "+officer.getID()+" "+officer.getName()+"
has been recruited to level one"+" "+officer.getWing()+" wing");
      }
      public void promote(levelOne off, ArrayList<levelOne> LevelOne,
ArrayList<levelTwo> LevelTwo,
                  ArrayList<PromotionRecord> promotionRecord) {

            levelTwo officer=new
levelTwo(off.id,off.name,off.wing,off.location,55000.00,off.bankaccount,off.awardsRec
ieved);
            LevelTwo.add(officer);
            Date date = new Date();
            PromotionRecord prObject = new PromotionRecord(officer.getID(), date,
2);
            promotionRecord.add(prObject);
            System.out.println("Officer: "+officer.getID()+" "+officer.getName()+"
has been promoted to level two"+" "+officer.getWing()+" wing");
            int i=0;
            for(i=0;i<LevelOne.size();i++)
            {
                  if(off.getID()==LevelOne.get(i).getID())
                  {
                        break;
                  }
            }
            if(i!=LevelOne.size())
            {
                  LevelOne.remove(i);
            }
            /*
             * TODO: Check creation of new record and deletion of previous one
             */
```

```
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}

	public boolean validate(String username,String password)
	{
		/*
		 * TODO: check validate function and integrate
		 */

		return (this.username.equals(username) &&
this.password.equals(password));
	}

	public void changeCredentials()
	{
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the new username: ");
		this.username=sc.next();
		System.out.println("Enter the new password: ");
		this.password=sc.next();
		System.out.println("Credentials changed successfully!");
	}


}
```

5) levelFour class

```
package personnel;


import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.Date;
```

```
import Utility.Award;
import Utility.AwardRecord;
import Utility.PromotionRecord;
import Utility.RetirementRecord;
import Utility.TransactionRecord;


public class levelFour extends levelThree {
	public levelFour(Integer id,String username, String password, String name,String wing,String location,Double BaseSalary,Integer bid,Double balance,
				ArrayList<Award> award){ //from files
		super(id,username, password, name,wing,location,BaseSalary,bid,balance,award);
		this.level=4;
	}
	public void promote(levelTwo off, ArrayList <levelTwo> LevelTwo, ArrayList<levelThree> LevelThree,
				ArrayList<PromotionRecord> promotionRecord) {
		/*
		 * For now - when creation of a new level 3 officer - User name will be name and password will be id
		 */
		levelThree officer=new levelThree(off.id, off.name, off.id.toString() ,off.name,off.wing,off.location,60000.00,off.bankaccount,off.awardsRecieved);
		LevelThree.add(officer);
		Date date = new Date();
		PromotionRecord prObject = new PromotionRecord(officer.getID(), date, 3);
		promotionRecord.add(prObject);
		System.out.println("Officer: "+officer.getID()+" "+officer.getName()+" has been promoted to level three"+" "+officer.getWing()+" wing");
		int i=0;
```

```
		for(i=0;i<LevelTwo.size();i++)
		{
			if(off.getID()==LevelTwo.get(i).getID())
			{
				break;
			}
		}
		if(i!=LevelTwo.size())
		{
			LevelTwo.remove(i);
		}
	}
	public void getAward(levelOne officer, ArrayList<AwardRecord> awardRecord,
ArrayList<TransactionRecord> transactionRecord) {
		try {
			Award award = new Award();
			award = officer.AddAward();
			AwardRecord ar = new AwardRecord(officer.getID(), award.getDate(),
award.getName(), award.getPrizeMoney());
			awardRecord.add(ar);
			TransactionRecord tr = new TransactionRecord(officer.getID(), award.getDate(),
officer.bankaccount.getAccountNumber(),
					award.getPrizeMoney(), "Award");
			System.out.println("Officer: "+officer.getID()+" "+officer.getName()+" has been
awarded"+" "+award.getName());
			System.out.println("They have received "+award.getPrizeMoney());
			transactionRecord.add(tr);
		}
		catch (Exception e) {
			System.out.print("error");
```

```
		}
	}

	public void getAward(levelTwo officer, ArrayList<AwardRecord> awardRecord,
ArrayList<TransactionRecord> transactionRecord) {
		try {
			Award award = new Award();
			award = officer.AddAward();
			AwardRecord ar = new AwardRecord(officer.getID(), award.getDate(),
award.getName(), award.getPrizeMoney());
			awardRecord.add(ar);
			TransactionRecord tr = new TransactionRecord(officer.getID(), award.getDate(),
officer.bankaccount.getAccountNumber(),
					award.getPrizeMoney(), "Award");
			System.out.println("Officer: "+officer.getID()+" "+officer.getName()+" has been
awarded"+" "+award.getName());
			System.out.println("They have received "+award.getPrizeMoney());
			transactionRecord.add(tr);
		}
		catch (Exception e) {
			System.out.print("error");
		}
	}
	public void getAward(levelThree officer, ArrayList<AwardRecord> awardRecord,
ArrayList<TransactionRecord> transactionRecord) {
		try {
			Award award = new Award();
			award = officer.AddAward();
			AwardRecord ar = new AwardRecord(officer.getID(), award.getDate(),
award.getName(), award.getPrizeMoney());
			awardRecord.add(ar);
```

```
                    TransactionRecord tr = new TransactionRecord(officer.getID(), award.getDate(),
officer.bankaccount.getAccountNumber(),
                                award.getPrizeMoney(), "Award");
                    System.out.println("Officer: "+officer.getID()+" "+officer.getName()+" has been
awarded"+" "+award.getName());
                    System.out.println("They have received "+award.getPrizeMoney());
                    transactionRecord.add(tr);
                }
                catch (Exception e) {
                    System.out.print("error");
                }
        }
        public void getReport(Date startDate, Date endDate, int searchCategory,
ArrayList<AwardRecord> awardRecord,
                    ArrayList<TransactionRecord> transactionRecord, ArrayList<RetirementRecord>
retirementRecord,
                    ArrayList<PromotionRecord> promotionRecord)
        {
            if(searchCategory == 1)
            {
                //award report
                int i=0;
                ArrayList<AwardRecord> awardRecordReport = new ArrayList<AwardRecord>();
                for(i=0;i<awardRecord.size();i++)
                {
                    if(!startDate.after(awardRecord.get(i).getDate()) &&
!endDate.before(awardRecord.get(i).getDate()))
                    {
                        /* Start date<= iterator date <= End date */
                        awardRecordReport.add(awardRecord.get(i));
```

```
                }
            }
            if(awardRecordReport.isEmpty())
            {
                System.out.println("No records are available between the given dates");
            }
            else
            {
                System.out.println(awardRecordReport.size());
                System.out.println("----------------------------------------");
                for(i=0;i<awardRecordReport.size();i++)
                {
                    System.out.println("Officer ID:
"+awardRecord.get(i).getOfficerID());
                    System.out.println("Date:
"+awardRecord.get(i).getStringDate());
                    System.out.println("Award name:
"+awardRecord.get(i).getAwardName());
                    System.out.println("Prize money:
"+awardRecord.get(i).getPrizeMoney());
                    System.out.println("----------------------------------------");

                }
                PrintWriter fout;
                try {
                    /*
                     * TODO check path to file locations
                     */
                    File directory = new File("");
```

```
                                        fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\AwardRecordReport.txt");
                                        fout.println(awardRecordReport.size());
                                        fout.println("----------------------------------------");
                                        for(i=0;i<awardRecordReport.size();i++)
                                        {
                                                fout.println("Officer ID:
"+awardRecordReport.get(i).getOfficerID());
                                                fout.println("Date:
"+awardRecordReport.get(i).getStringDate());
                                                fout.println("Award name:
"+awardRecordReport.get(i).getAwardName());
                                                fout.println("Prize money:
"+awardRecordReport.get(i).getPrizeMoney());
                                                fout.println("----------------------------------------");
                                        }
                                        fout.close();
                                } catch (FileNotFoundException e) {
                                        System.out.println("File not found");
                                }
                        }
                }
                else if(searchCategory ==  2)
                {
                        //transaction report
                        int i=0;
                        ArrayList<TransactionRecord> transactionRecordReport = new
ArrayList<TransactionRecord>();
                        for(i=0;i<transactionRecord.size();i++)
                        {
```

```
                    if(!startDate.after(transactionRecord.get(i).getDate()) &&
!endDate.before(transactionRecord.get(i).getDate()))
                    {
                        /* Start date<= iterator date <= End date */
                        transactionRecordReport.add(transactionRecord.get(i));
                    }
                }
                if(transactionRecordReport.isEmpty())
                {
                    System.out.println("No records are available between the given dates");
                }
                else
                {

                    System.out.println(transactionRecordReport.size());
                    System.out.println("----------------------------------------");
                    for(i=0;i<transactionRecordReport.size();i++)
                    {
                        System.out.println("Officer ID:
"+transactionRecordReport.get(i).getOfficerID());
                        System.out.println("Date:
"+transactionRecordReport.get(i).getStringDate());
                        System.out.println("Bank account number:
"+transactionRecordReport.get(i).getBankAccountNumber());
                        System.out.println("Amount:
"+transactionRecordReport.get(i).getAmount());
                        System.out.println("Type:
"+transactionRecordReport.get(i).getType());
                        System.out.println("----------------------------------------");
                    }
```

```
                PrintWriter fout;
                try {
                        File directory = new File("");
                        fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\TransactionRecordReport.txt");
                        /*
                         * Check file path
                         */
                        fout.println(transactionRecordReport.size());
                        fout.println("----------------------------------------");
                        for(i=0;i<transactionRecordReport.size();i++)
                        {
                                fout.println("Officer ID:
"+transactionRecordReport.get(i).getOfficerID());
                                fout.println("Date:
"+transactionRecordReport.get(i).getStringDate());
                                fout.println("Bank account number:
"+transactionRecordReport.get(i).getBankAccountNumber());
                                fout.println("Amount:
"+transactionRecordReport.get(i).getAmount());
                                fout.println("Type:
"+transactionRecordReport.get(i).getType());
                                fout.println("----------------------------------------");
                        }
                        fout.close();
                } catch (FileNotFoundException e) {
                        System.out.println("File not found");
                }
        }

}
```

```
else if(searchCategory == 3)
{
	//Retirement report
	int i=0;
	ArrayList<RetirementRecord> retirementRecordReport = new
ArrayList<RetirementRecord>();
	for(i=0;i<retirementRecord.size();i++)
	{
		if(!startDate.after(retirementRecord.get(i).getDate()) &&
!endDate.before(retirementRecord.get(i).getDate()))
		{
			/* Start date<= iterator date <= End date */
			retirementRecordReport.add(retirementRecord.get(i));
		}
	}
	if(retirementRecordReport.isEmpty())
	{
		System.out.println("No records are available between the given dates");
	}
	else
	{
		System.out.println(retirementRecordReport.size());
		System.out.println("----------------------------------------");
		for(i=0;i<retirementRecordReport.size();i++)
		{
			System.out.println("Officer ID:
"+retirementRecordReport.get(i).getOfficerID());
			System.out.println("Date:
"+retirementRecordReport.get(i).getStringDate());
```

```
                                        System.out.println("Level during retirement:
"+retirementRecordReport.get(i).getRankDuringRetirement());
                                        System.out.println("----------------------------------------");
                                }


                                PrintWriter fout;
                                try {
                                        /*
                                         * TODO check file path
                                         */
                                        File directory = new File("");
                                        fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\RetirementRecordReport.txt");
                                        fout.println(retirementRecordReport.size());
                                        fout.println("----------------------------------------");
                                        for(i=0;i<retirementRecordReport.size();i++)
                                        {
                                                fout.println("Officer ID:
"+retirementRecordReport.get(i).getOfficerID());
                                                fout.println("Date:
"+retirementRecordReport.get(i).getStringDate());
                                                fout.println("Level during retirement:
"+retirementRecordReport.get(i).getRankDuringRetirement());
                                                fout.println("----------------------------------------");
                                        }
                                        fout.close();
                                } catch (FileNotFoundException e) {
                                        System.out.println("File not found");
                                }
                        }
```

```
		}
		else
		{
			//Promotion report
			int i=0;
			ArrayList<PromotionRecord> promotionRecordReport = new
ArrayList<PromotionRecord>();
			for(i=0;i<promotionRecord.size();i++)
			{
				if(!startDate.after(promotionRecord.get(i).getDate()) &&
!endDate.before(promotionRecord.get(i).getDate()))
				{
					/* Start date<= iterator date <= End date */
					promotionRecordReport.add(promotionRecord.get(i));
				}
			}
			if(promotionRecordReport.isEmpty())
			{
				System.out.println("No records are available between the given dates");
			}
			else
			{
				System.out.println(promotionRecordReport.size());
				System.out.println("----------------------------------------");
				for(i=0;i<promotionRecordReport.size();i++)
				{
					System.out.println("Officer ID:
"+promotionRecordReport.get(i).getOfficerID());
```

```
                                        System.out.println("Date:
"+promotionRecordReport.get(i).getStringDate());
                                        System.out.println("Next level:
"+promotionRecordReport.get(i).getNextLevel());
                                        System.out.println("----------------------------------------");
                                }
                                PrintWriter fout;
                                try {
                                        File directory = new File("");
                                        fout = new
PrintWriter(directory.getAbsolutePath()+"\\Files\\PromotionRecordReport.txt");
                                        fout.println(promotionRecordReport.size());
                                        fout.println("----------------------------------------");
                                        for(i=0;i<promotionRecordReport.size();i++)
                                        {
                                                fout.println("Officer ID:
"+promotionRecordReport.get(i).getOfficerID());
                                                fout.println("Date:
"+promotionRecordReport.get(i).getStringDate());
                                                fout.println("Next level:
"+promotionRecordReport.get(i).getNextLevel());
                                                fout.println("----------------------------------------");
                                        }
                                        fout.close();
                                } catch (FileNotFoundException e) {
                                        System.out.println("File not found");
                                }
                        }

                }
        }
```

```
}
```

6) Manager class

```
package personnel;

import java.util.ArrayList;
import java.util.Date;

import Utility.Login;
import Utility.RetirementRecord;
import Utility.TransactionRecord;

public class Manager implements Login{
	private String name;
	private String username, password;

	private Integer ID;
	private static Integer managerCount=0;
	private static Integer month=-1;

	public Manager (String name, int ID, String username, String password){
		this.name=name;
		this.ID= ID;
		this.username=username;
		this.password=password;
		managerCount++;
	}

	public static Integer getMonth() {
```

```
		return month;
	}
	public static void setMonth(Integer month) {
		Manager.month = month;
	}

	public void initiateRetirement(levelOne officer,ArrayList<levelOne> LevelOne,
ArrayList<Veteran> veteran,
			ArrayList<RetirementRecord> retirementRecord) {
		Veteran vt = new Veteran(officer, 8000.00);
		Date date = new Date();
		RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
		retirementRecord.add(rt);
		veteran.add(vt);
		System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
		System.out.println("Their pension amount is: 8000.00");
		int i=0;
		for(i=0;i<LevelOne.size();i++)
		{
			if(officer.getID()==LevelOne.get(i).getID())
			{
				break;
			}
		}
		if(i!=LevelOne.size())
		{
			LevelOne.remove(i);
		}
```

```
	}
	public void initiateDischarge(levelOne officer,ArrayList<levelOne> LevelOne, ArrayList<Veteran>
veteran,
				ArrayList<RetirementRecord> retirementRecord) {
		Veteran vt = new Veteran(officer, 8000.00);
		Date date = new Date();
		RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
		retirementRecord.add(rt);
		veteran.add(vt);
		System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
		System.out.println("Their pension amount is: 8000.00");
		int i=0;
		for(i=0;i<LevelOne.size();i++)
		{
			if(officer.getID()==LevelOne.get(i).getID())
			{
				break;
			}
		}
		if(i!=LevelOne.size())
		{
			LevelOne.remove(i);
		}
		/*
		 * TODO: Check if both retirement and discharge differ in any way?
		 */
	}
```

```
	public void initiateRetirement(levelTwo officer,ArrayList<levelTwo> LevelTwo,
ArrayList<Veteran> veteran,
				ArrayList<RetirementRecord> retirementRecord) {
			Veteran vt = new Veteran(officer, 10000.00);
			Date date = new Date();
			RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
			retirementRecord.add(rt);
			veteran.add(vt);
			 System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
			 System.out.println("Their pension amount is: 10000.00");
			int i=0;
			for(i=0;i<LevelTwo.size();i++)
			{
				if(officer.getID()==LevelTwo.get(i).getID())
				{
					break;
				}
			}
			if(i!=LevelTwo.size())
			{
				LevelTwo.remove(i);
			}
		}
	public void initiateDischarge(levelTwo officer,ArrayList<levelTwo> LevelTwo, ArrayList<Veteran>
veteran,
				ArrayList<RetirementRecord> retirementRecord) {
			Veteran vt = new Veteran(officer, 10000.00);
			Date date = new Date();
```

```
                RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
                retirementRecord.add(rt);
                veteran.add(vt);
                System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
                System.out.println("Their pension amount is: 10000.00");
                int i=0;
                for(i=0;i<LevelTwo.size();i++)
                {
                        if(officer.getID()==LevelTwo.get(i).getID())
                        {
                                break;
                        }
                }
                if(i!=LevelTwo.size())
                {
                        LevelTwo.remove(i);
                }
        }

        public void initiateRetirement(levelThree officer,ArrayList<levelThree> LevelThree,
ArrayList<Veteran> veteran,
                        ArrayList<RetirementRecord> retirementRecord) {
                Veteran vt = new Veteran(officer, 12000.00);
                Date date = new Date();
                RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
                retirementRecord.add(rt);
                veteran.add(vt);
                System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
```

```
                System.out.println("Their pension amount is: 12000.00");
                int i=0;
                for(i=0;i<LevelThree.size();i++)
                {
                        if(officer.getID()==LevelThree.get(i).getID())
                        {
                                break;
                        }
                }
                if(i!=LevelThree.size())
                {
                        LevelThree.remove(i);
                }
        }
        public void initiateDischarge(levelThree officer,ArrayList<levelThree> LevelThree,
ArrayList<Veteran> veteran,
                        ArrayList<RetirementRecord> retirementRecord) {
                Veteran vt = new Veteran(officer, 12000.00);
                Date date = new Date();
                RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
                retirementRecord.add(rt);
                veteran.add(vt);
                System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
                System.out.println("Their pension amount is: 12000.00");
                int i=0;
                for(i=0;i<LevelThree.size();i++)
                {
                        if(officer.getID()==LevelThree.get(i).getID())
```

```
                {
                        break;
                }
        }
        if(i!=LevelThree.size())
        {
                LevelThree.remove(i);
        }
    }

    public void initiateRetirement(levelFour officer,ArrayList<levelFour> LevelFour,
ArrayList<Veteran> veteran,
                ArrayList<RetirementRecord> retirementRecord) {
        Veteran vt = new Veteran(officer, 14000.00);
        Date date = new Date();
        RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
        retirementRecord.add(rt);
        veteran.add(vt);
        System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
         System.out.println("Their pension amount is: 14000.00");
        int i=0;
        for(i=0;i<LevelFour.size();i++)
        {
                if(officer.getID()==LevelFour.get(i).getID())
                {
                        break;
                }
        }
```

```
		if(i!=LevelFour.size())
		{
			LevelFour.remove(i);
		}
	}
	public void initiateDischarge(levelFour officer,ArrayList<levelFour> LevelFour,
ArrayList<Veteran> veteran,
			ArrayList<RetirementRecord> retirementRecord) {
		Veteran vt = new Veteran(officer, 14000.00);
		Date date = new Date();
		RetirementRecord rt = new RetirementRecord(officer.getID(),date, officer.getLevel());
		retirementRecord.add(rt);
		veteran.add(vt);
		System.out.println("Officer: "+officer.getName()+" with Officer ID: "+officer.getID()+"
has been moved to veteran's list");
		System.out.println("Their pension amount is: 14000.00");
		int i=0;
		for(i=0;i<LevelFour.size();i++)
		{
			if(officer.getID()==LevelFour.get(i).getID())
			{
				break;
			}
		}
		if(i!=LevelFour.size())
		{
			LevelFour.remove(i);
		}
	}
```

```
	public boolean validate(String username,String password)
	{
		return (this.username.equals(username) && this.password.equals(password));
	}
	public void initiatePayroll(ArrayList<levelOne> LevelOne, ArrayList <levelTwo> LevelTwo,
ArrayList<levelThree> LevelThree,
			ArrayList<levelFour> LevelFour, ArrayList<Veteran> veteran,
ArrayList<TransactionRecord> transactionRecord) {

		Date date = new Date();
		Integer month = date.getMonth();
		Manager.month = month;
		int i=0;
		for(i=0;i<LevelOne.size();i++)
		{

	LevelOne.get(i).bankaccount.setBalance(LevelOne.get(i).getBalance()+LevelOne.get(i).getBaseSa
lary());
			System.out.println("Salary paid for: "+LevelOne.get(i).getID()+"
"+LevelOne.get(i).getName());
			TransactionRecord tr = new TransactionRecord(LevelOne.get(i).getID(), date,
				LevelOne.get(i).bankaccount.getAccountNumber(),
LevelOne.get(i).getBaseSalary(), "Salary");
			transactionRecord.add(tr);
		}

		for(i=0;i<LevelTwo.size();i++)
		{

	LevelTwo.get(i).bankaccount.setBalance(LevelTwo.get(i).getBalance()+LevelTwo.get(i).getBaseS
alary());
```

```
                    System.out.println("Salary paid for: "+LevelTwo.get(i).getID()+"
"+LevelTwo.get(i).getName());

                    TransactionRecord tr = new TransactionRecord(LevelTwo.get(i).getID(), date,

                                        LevelTwo.get(i).bankaccount.getAccountNumber(),
LevelTwo.get(i).getBaseSalary(), "Salary");

                    transactionRecord.add(tr);

                }


                for(i=0;i<LevelThree.size();i++)

                {


        LevelThree.get(i).bankaccount.setBalance(LevelThree.get(i).getBalance()+LevelThree.get(i).getB
aseSalary());

                    System.out.println("Salary paid for: "+LevelThree.get(i).getID()+"
"+LevelThree.get(i).getName());

                    TransactionRecord tr = new TransactionRecord(LevelThree.get(i).getID(), date,

                                        LevelThree.get(i).bankaccount.getAccountNumber(),
LevelThree.get(i).getBaseSalary(), "Salary");

                    transactionRecord.add(tr);

                }


                for(i=0;i<LevelFour.size();i++)

                {


        LevelFour.get(i).bankaccount.setBalance(LevelFour.get(i).getBalance()+LevelFour.get(i).getBaseS
alary());

                    System.out.println("Salary paid for: "+LevelFour.get(i).getID()+"
"+LevelFour.get(i).getName());

                    TransactionRecord tr = new TransactionRecord(LevelFour.get(i).getID(), date,

                                        LevelFour.get(i).bankaccount.getAccountNumber(),
LevelFour.get(i).getBaseSalary(), "Salary");

                    transactionRecord.add(tr);
```

```
                }
                for(i=0;i<veteran.size();i++)
                {

        veteran.get(i).bankaccount.setBalance(veteran.get(i).getBalance()+veteran.get(i).getPensionAm
ount());
                        System.out.println("Pension paid for: "+veteran.get(i).getID()+"
"+veteran.get(i).getName());
                        TransactionRecord tr = new TransactionRecord(veteran.get(i).getID(), date,
                                        veteran.get(i).bankaccount.getAccountNumber(),
veteran.get(i).getPensionAmount() , "Pension");
                        transactionRecord.add(tr);
                }
        }

        public String getUsername() {
                return username;
        }
        public void setUsername(String username) {
                this.username = username;
        }
        public String getPassword() {
                return password;
        }
        public void setPassword(String password) {
                this.password = password;
        }

        public String getName() {
                return name;
```

```
	}


	public void setName(String name) {
		this.name = name;
	}


	public Integer getID() {
		return ID;
	}


	public void setID(Integer iD) {
		ID = iD;
	}




}
```

7) Veteran class

```
package personnel;

public class Veteran extends Personnel {
	public double pensionAmount;
	public static int veteranCount=0;
	public Veteran(String name, Integer id, Integer accountNumber, Double balance,
double pensionAmount)//from files
	{
		super(id, name, accountNumber, balance);
		veteranCount++;
	}
	public Veteran(levelOne officer,double pensionAmount) {//creating new veteran
		super(officer.getID(), officer.getName(), officer.getBankaccount());
		this.pensionAmount = pensionAmount;
		veteranCount++;
	}
	public Veteran(levelTwo officer,double pensionAmount) {
		super(officer.getID(),  officer.getName(), officer.getBankaccount());
		this.pensionAmount = pensionAmount;
```

```
		veteranCount++;
	}
	public Veteran(levelThree officer,double pensionAmount) {
		super(officer.getID(), officer.getName(), officer.getBankaccount());
		this.pensionAmount = pensionAmount;
		veteranCount++;
	}
	public Veteran(levelFour officer,double pensionAmount) {
		super(officer.getID(), officer.getName(), officer.getBankaccount());
		this.pensionAmount = pensionAmount;
		veteranCount++;
	}

	public static int getVeteranCount() {
		return veteranCount;
	}
	public double getPensionAmount() {
		return pensionAmount;
	}
	public void setPensionAmount(double pensionAmount) {
		this.pensionAmount = pensionAmount;
	}
```

C. Utility

1) Allowance class

```
package Utility;


import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;


import personnel.levelFour;

import personnel.levelThree;

import personnel.levelTwo;;


public class Allowance {

	private String name;
```

```
	private Pair amount;
	private Integer minLevel;
	private String wing;

	public Allowance(String name, Pair amount, Integer minLevel, String wing) {
		this.name = name;
		this.amount = amount;
		this.minLevel = minLevel;
		this.wing = wing;
	}
	public static HashMap<String, Allowance> map = new HashMap<>();
	public static void intializeAllowanceMap()
	{
		//Army
		map.put("Classification-Allowance", new Allowance("Classification-Allowance", new
Pair("Fixed", 225.00), 2, "Army"));
		map.put("Rum-Allowance", new Allowance("Rum-Allowance", new Pair("Fixed",
360.00), 2, "Army"));

		map.put("CPM-Allowance", new Allowance("CPM-Allowance", new Pair("Fixed", 90.00),
3, "Army"));
		map.put("Dress-Allowance", new Allowance("Dress-Allowance", new Pair("Fixed",
1700.00), 3, "Army"));
		map.put("Extra-Allowance", new Allowance("Extra-Allowance", new Pair("Variable",
2.00), 3, "Army"));

		map.put("Language-Allowance", new Allowance("Language-Allowance", new
Pair("Fixed", 2025.00), 4, "Army"));
		map.put("PG-Allowance", new Allowance("PG-Allowance", new Pair("Fixed", 2250.00),
4, "Army"));
```

```
//Navy

map.put("Diving-Allowance", new Allowance("Diving-Allowance", new Pair("Fixed",
1800.00), 2, "Navy"));

map.put("Dip-Allowance", new Allowance("Dip-Allowance", new Pair("Fixed", 3500.00),
2, "Navy"));

map.put("Diving-attendant-Allowance", new Allowance("Diving-attendant-Allowance",
new Pair("Fixed", 700.00), 2, "Navy"));


map.put("ISD-Allowance", new Allowance("ISD-Allowance", new Pair("Variable", 16.00),
3, "Navy"));

map.put("Sea-going-Allowance", new Allowance("Sea-going-Allowance", new
Pair("Fixed", 10500.00), 3, "Navy"));


map.put("Submarine-duty-Allowance", new Allowance("Submarine-duty-Allowance",
new Pair("Fixed", 5300.00), 4, "Navy"));

map.put("Submarine-technical-Allowance", new Allowance("Submarine-technical-
Allowance", new Pair("Fixed", 1000.00), 4, "Navy"));


//Air force

map.put("Air-worthiness-Allowance", new Allowance("Air-worthiness-Allowance", new
Pair("Fixed", 338.00), 2, "Air-Force"));

map.put("High-altitude-Allowance", new Allowance("High-altitude-Allowance", new
Pair("Fixed", 5300.00), 2, "Air-Force"));

map.put("Flying-Allowance", new Allowance("Flying-Allowance", new Pair("Fixed",
17300.00), 2, "Air-Force"));


map.put("CPM-Allowance", new Allowance("CPM-Allowance", new Pair("Fixed",
1700.00), 3, "Air-Force"));

map.put("Dress-Allowance", new Allowance("Dress-Allowance", new Pair("Fixed",
1700.00), 3, "Air-Force"));


map.put("Piloting-Allowance", new Allowance("Piloting-Allowance", new Pair("Fixed",
25000.00), 4, "Air-Force"));
```

```
	}


	/*
	 * TODO: functions to check eligibility (after officer classes)
	 */
	public static boolean eligibleforAllowance(String allowanceName, levelTwo officer,
ArrayList<TransactionRecord> transactionRecord)
	{
		if(Allowance.map.containsKey(allowanceName))
		{
			Allowance temp = map.get(allowanceName);
			if(officer.getLevel()>=temp.minLevel && officer.getWing().equals(temp.wing))
			{
				if(temp.amount.getKey().equals("Fixed"))
				{

	officer.bankaccount.setBalance(officer.getBalance()+temp.amount.getValue());
					Date date = new Date();
					TransactionRecord tr = new TransactionRecord(officer.getID(),
date, officer.bankaccount.getAccountNumber(),
							temp.amount.getValue(), "Allowance");
					transactionRecord.add(tr);
					System.out.println("Officer: "+officer.getID()+"
"+officer.getName()+" is eligible for this allowance");
					System.out.println(temp.name+": "+temp.amount.getValue());
					return true;
				}
				else
				{
```

```
                                        Double allowMoney =
(officer.getBaseSalary()*temp.amount.getValue())/100;


        officer.bankaccount.setBalance(officer.getBalance()+allowMoney);

                                        System.out.println("Officer: "+officer.getID()+"
"+officer.getName()+" is eligible for this allowance");

                                        System.out.println(temp.name+": "+allowMoney);

                                        return true;

                                }

                        }

                        else

                        {

                                return false;

                        }

                }

                else

                {

                        return false;

                }


        }

        public static boolean eligibleforAllowance(String allowanceName, levelThree officer,
ArrayList<TransactionRecord> transactionRecord)

        {

                if(Allowance.map.containsKey(allowanceName))

                {

                        Allowance temp = map.get(allowanceName);

                        if(officer.getLevel()>=temp.minLevel && officer.getWing().equals(temp.wing))

                        {

                                if( temp.amount.getKey().equals("Fixed"))
```

```
                    {

        officer.bankaccount.setBalance(officer.getBalance()+temp.amount.getValue());
                        Date date = new Date();
                        TransactionRecord tr = new TransactionRecord(officer.getID(),
date, officer.bankaccount.getAccountNumber(),
                                temp.amount.getValue(), "Allowance");
                        transactionRecord.add(tr);
                        System.out.println("Officer: "+officer.getID()+"
"+officer.getName()+" is eligible for this allowance");
                        System.out.println(temp.name+": "+temp.amount.getValue());
                        return true;
                    }
                    else
                    {
                        Double allowMoney =
(officer.getBaseSalary()*temp.amount.getValue())/100;

        officer.bankaccount.setBalance(officer.getBalance()+allowMoney);
                        System.out.println("Officer: "+officer.getID()+"
"+officer.getName()+" is eligible for this allowance");
                        System.out.println(temp.name+": "+allowMoney);
                        return true;
                    }
                }
                else
                {
                    return false;
                }
            }
            else
```

```
		{
			return false;
		}
	}
	public static boolean eligibleforAllowance(String allowanceName, levelFour officer,
ArrayList<TransactionRecord> transactionRecord)
	{
		if(Allowance.map.containsKey(allowanceName))
		{
			Allowance temp = map.get(allowanceName);
			if(officer.getLevel()>=temp.minLevel && officer.getWing().equals(temp.wing))
			{
				if( temp.amount.getKey().equals("Fixed"))
				{

	officer.bankaccount.setBalance(officer.getBalance()+temp.amount.getValue());
					Date date = new Date();
					TransactionRecord tr = new TransactionRecord(officer.getID(),
date, officer.bankaccount.getAccountNumber(),
							temp.amount.getValue(), "Allowance");
					transactionRecord.add(tr);
					System.out.println("Officer: "+officer.getID()+"
"+officer.getName()+" is eligible for this allowance");
					System.out.println(temp.name+": "+temp.amount.getValue());
					return true;
				}
				else
				{
					Double allowMoney =
(officer.getBaseSalary()*temp.amount.getValue())/100;
```

```
officer.bankaccount.setBalance(officer.getBalance()+allowMoney);
                                        System.out.println("Officer: "+officer.getID()+"
"+officer.getName()+" is eligible for this allowance");
                                        System.out.println(temp.name+": "+allowMoney);
                                        return true;
                                }
                        }
                        else
                        {
                                return false;
                        }
                }
                else
                {
                        return false;
                }
        }
}
```

2) Award Class

```
package Utility;

import java.text.SimpleDateFormat;
import java.util.Date;

public class Award {
        private String name;
        private Double prizeMoney;
```

```
private Date date;

public Award()
{
        this.name=" ";
        this.prizeMoney=0.0;
        this.date= new Date();
}
public Award(String name, Double prizeMoney, Date date) {
        this.name = name;
        this.prizeMoney = prizeMoney;
        this.date = date;
}

public String getName() {
        return name;
}

public void setName(String name) {
        this.name = name;
}

public Double getPrizeMoney() {
        return prizeMoney;
}

public void setPrizeMoney(Double prizeMoney) {
        this.prizeMoney = prizeMoney;
}
```

```
	public Date getDate() {
		return date;
	}

	public String getStringDate()
	{
		String pattern = "dd-MM-yyyy";
		SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
		String Sdate = simpleDateFormat.format(this.date);
		return Sdate;
	}

	public void setDate(Date date) {
		this.date = date;
	}

}
```

3) AwardRecord class

```
package Utility;

import java.util.Date;

public class AwardRecord extends Record{
	public String getAwardName() {
		return awardName;
	}

	public void setAwardName(String awardName) {
		this.awardName = awardName;
	}

	public Double getPrizeMoney() {
		return prizeMoney;
```

```
	}

	public void setPrizeMoney(Double prizeMoney) {
		this.prizeMoney = prizeMoney;
	}

	private String awardName;
	private Double prizeMoney;

	public AwardRecord(Integer officerID, Date date, String awardName, Double
prizeMoney) {
		super(officerID, date);
		this.awardName = awardName;
		this.prizeMoney = prizeMoney;
	}
}
```

4) BankAccount class

```
package Utility;

public class BankAccount {
	public static Integer getAccountCount() {
		return accountCount;
	}

	public static void setAccountCount(Integer accountCount) {
		BankAccount.accountCount = accountCount;
	}

	public Integer getAccountNumber() {
		return accountNumber;
	}

	public void setAccountNumber(Integer accountNumber) {
		this.accountNumber = accountNumber;
	}

	public Double getBalance() {
		return balance;
	}

	public void setBalance(Double balance) {
		this.balance = balance;
	}
	private static Integer accountCount = 0;
	private Integer accountNumber;
	private Double balance;

	public BankAccount(Double balance) {
		this.accountNumber = accountCount++;
		this.balance = balance;
	}
```

```
	public BankAccount(Integer accountNumber, Double balance) {
		this.accountNumber=accountNumber;
		this.balance = balance;
		accountCount++;
	}

	/**
	 * @param amount Amount to be transferred (should be positive)
	 * @return boolean Validity of transaction
	 */
	public boolean tranferAmount(Double amount) {
		if(amount < 0) return false;
		this.balance += amount;
		return true;
	}
	public Integer getBankAccountId() {
		return this.accountNumber;
	}
}
```

5) Login interface

```
package Utility;
public interface Login {

	boolean validate(String username, String password);

}
```

6) Pair class

```
package Utility;

public class Pair {
	private String key;
	private Double value;
	public Pair(String key, Double value)
	{
		this.key=key;
		this.value=value;
	}
	public String getKey() {
		return key;
	}
	public void setKey(String key) {
		this.key = key;
	}
	public Double getValue() {
		return value;
```

```
	}
	public void setValue(Double value) {
		this.value = value;
	}

}
```

7) Payment Class

```
package Utility;


import personnel.levelOne;

import personnel.levelTwo;

import personnel.levelThree;

import personnel.levelFour;


public class Payment {

	private static Integer PaymentCount=0;

	private Integer ID, accountNo;

	private Double amount;

	private String type;


	/*

	 * TODO: Constructors after officer classes are complete - Completed check the constructors

	 */

	public Payment(levelOne officer, Double amount, String type) {

		this.ID=PaymentCount++;

		this.accountNo=officer.bankaccount.getBankAccountId();

		this.amount=amount;

		this.type=type;

	}
```

```
	public Payment(levelTwo officer, Double amount, String type) {
		this.ID=PaymentCount++;
		this.accountNo=officer.bankaccount.getBankAccountId();
		this.amount=amount;
		this.type=type;
	}

	public Payment(levelThree officer, Double amount, String type) {
		this.ID=PaymentCount++;
		this.accountNo=officer.bankaccount.getBankAccountId();
		this.amount=amount;
		this.type=type;
	}
	public Payment(levelFour officer, Double amount, String type) {
		this.ID=PaymentCount++;
		this.accountNo=officer.bankaccount.getBankAccountId();
		this.amount=amount;
		this.type=type;
	}

}
```

8) PromotionRecord class

```
package Utility;

import java.util.Date;

public class PromotionRecord extends Record {
	private Integer nextLevel;

	public PromotionRecord(Integer officerID, Date date, Integer nextLevel) {
		super(officerID, date);
		this.nextLevel = nextLevel;
	}
```

```
	public Integer getNextLevel() {
		return nextLevel;
	}

	public void setNextLevel(Integer nextLevel) {
		this.nextLevel = nextLevel;
	}

}
```

9) Record class

```
package Utility;

import java.text.SimpleDateFormat;

import java.util.Date;


public class Record {

	static Integer recordCount = 0;

	Integer officerID;

	Date date;


	protected Record(Integer officerID, Date date) {

		this.officerID = officerID;

		this.date = date;

	}


	public static Integer getRecordCount() {

		return recordCount;

	}


	public static void setRecordCount(Integer recordCount) {

		Record.recordCount = recordCount;
```

```
	}

	public Integer getOfficerID() {
		return officerID;
	}

	public void setOfficerID(Integer officerID) {
		this.officerID = officerID;
	}

	public Date getDate() {
		return date;
	}

	public String getStringDate()
	{
		String pattern = "dd-MM-yyyy";
		SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
		String Sdate = simpleDateFormat.format(this.date);
		return Sdate;
	}

	public void setDate(Date date) {
		this.date = date;
	}

}
```

10) Retirement class

```
package Utility;

import java.util.Date;

public class RetirementRecord extends Record{
	private Integer rankDuringRetirement;

	public RetirementRecord(Integer officerID, Date date, Integer
rankDuringRetirement) {
		super(officerID, date);
		this.rankDuringRetirement = rankDuringRetirement;
	}

	public Integer getRankDuringRetirement() {
		return rankDuringRetirement;
	}

	public void setRankDuringRetirement(Integer rankDuringRetirement) {
		this.rankDuringRetirement = rankDuringRetirement;
	}


}
```

11) Search class

```
package Utility;

import java.util.Date;

public class Search {
	private String searchCategory;
	private Date startDate, endDate;
}
```

12) Transaction Class

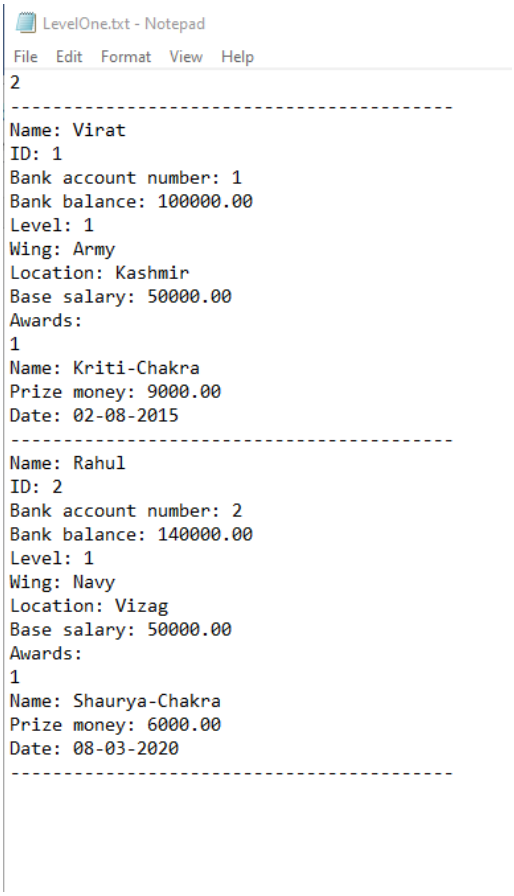
```
package Utility;

import java.util.Date;

public class TransactionRecord extends Record {
	private Integer bankAccountNumber;
	private Double amount;
	private String type;

	public TransactionRecord(Integer officerID, Date date, Integer
bankAccountNumber, Double amount, String type) {
		super(officerID, date);
```

```
		this.bankAccountNumber = bankAccountNumber;
		this.amount = amount;
		this.type = type;
	}

	public Integer getBankAccountNumber() {
		return bankAccountNumber;
	}

	public void setBankAccountNumber(Integer bankAccountNumber) {
		this.bankAccountNumber = bankAccountNumber;
	}

	public Double getAmount() {
		return amount;
	}

	public void setAmount(Double amount) {
		this.amount = amount;
	}

	public String getType() {
		return type;
	}

	public void setType(String type) {
		this.type = type;
	}

}
```

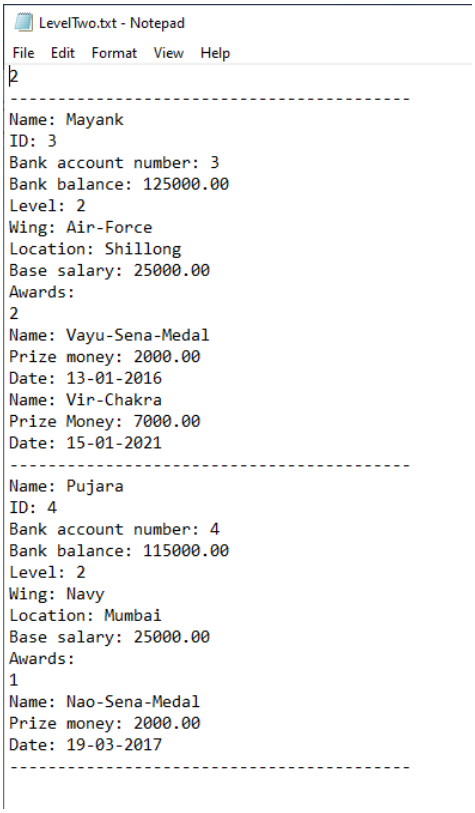
VIII. Inputs and Outputs

Input files

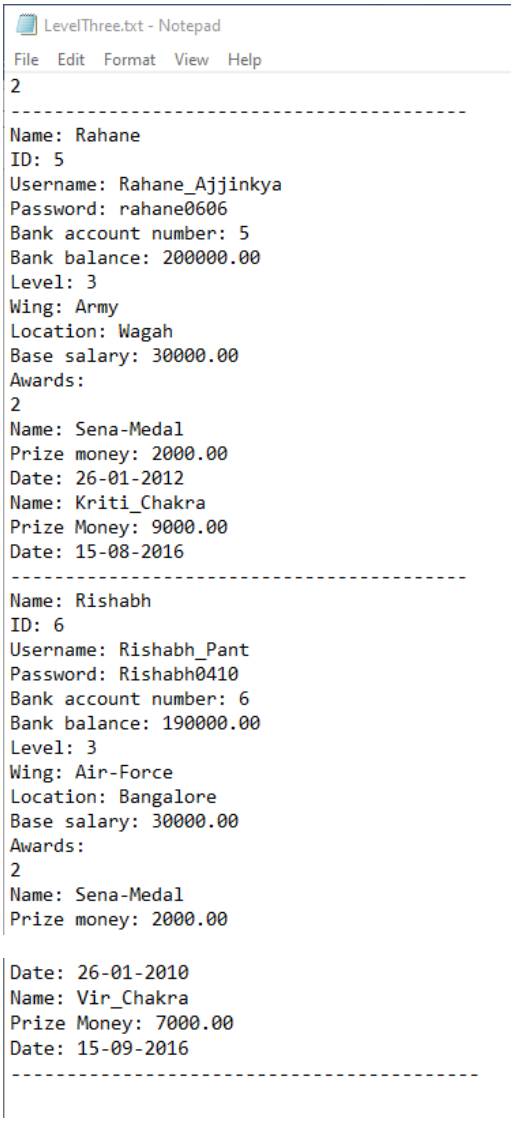
LevelOne.txt



LevelTwo.txt



LevelThree.txt


LevelThree.txt - Notepad
File Edit Format View Help
2

Name: Rahane
ID: 5
Username: Rahane_Ajjinkya
Password: rahane0606
Bank account number: 5
Bank balance: 200000.00
Level: 3
Wing: Army
Location: Wagah
Base salary: 30000.00
Awards:
2
Name: Sena-Medal
Prize money: 2000.00
Date: 26-01-2012
Name: Kriti_Chakra
Prize Money: 9000.00
Date: 15-08-2016

Name: Rishabh
ID: 6
Username: Rishabh_Pant
Password: Rishabh0410
Bank account number: 6
Bank balance: 190000.00
Level: 3
Wing: Air-Force
Location: Bangalore
Base salary: 30000.00
Awards:
2
Name: Sena-Medal
Prize money: 2000.00
Date: 26-01-2010
Name: Vir_Chakra
Prize Money: 7000.00
Date: 15-09-2016

LevelFour.txt

LevelFour.txt - Notepad

File Edit Format View Help

```
2
----------------------------------------
Name: Ashwin
ID: 7
Username: Ashwin_Ravi
Password: Ashwin1709
Bank account number: 7
Bank balance: 230000.00
Level: 4
Wing: Navy
Location: Kochi
Base salary: 35000.00
Awards:
3
Name: Nao-Sena-Medal
Prize money: 2000.00
Date: 19-03-2008
Name: Shaurya-Chakra
Prize money: 6000.00
Date: 29-03-2012
Name: Mahavir-Chakra
Prize money: 10000.00
Date: 16-09-2016
----------------------------------------
Name: Bumrah
ID: 8
Username: Bumrah_Jasprit
Password: bumrah0612
Bank account number: 8
Bank balance: 250000.00
Level: 4
Wing: Army
Location: Kashmir
Base salary: 35000.00
Awards:

2
Name: Sena-Medal
Prize money: 2000.00
Date: 19-03-2014
Name: Ashoka-Chakra
Prize money: 12000.00
Date: 26-10-2019
----------------------------------------
```

Manager

Manager.txt - Notepad

File Edit Format View Help

```
4
----------------------------------------
Name: Kishore
ID: 1
Username: Kishore_V
Password: kishore2809
----------------------------------------
Name: Ashwath
ID: 2
Username: Ashwath_VA
Password: ashwath2107
----------------------------------------
Name: Anish
ID: 3
Username: Anish_Ayyagari
Password: anish1207
----------------------------------------
Name: Yashasvi
ID: 4
Username: Chowta_Yashasvi
Password: yashasvi0101
----------------------------------------
11
```

Veteran.txt

Veteran.txt - Notepad

File Edit Format View Help

```
2
----------------------------------------
Name: Shardul
ID: 9
Bank account number: 9
Bank balance: 100000.00
Pension amount: 8000.00
----------------------------------------
Name: Umesh
ID: 10
Bank account number: 10
Bank balance: 110000.00
Pension amount: 10000.00
----------------------------------------
```

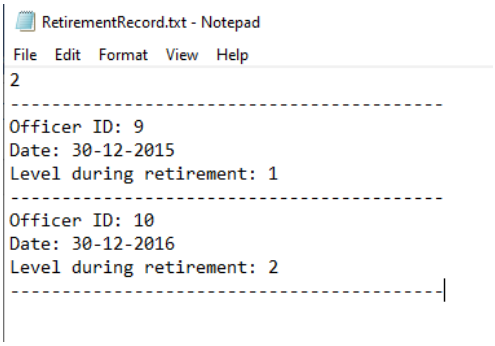
PromotionRecord.txt

PromotionRecord.txt - Notepad

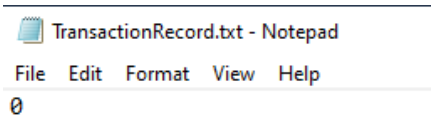
File Edit Format View Help

```
0
```

RetirementRecord.txt



TransactionRecord.txt



AwardRecord.txt

AwardRecord.txt - Notepad

File Edit Format View Help

```
12
-----------------------------------------
Officer ID: 7
Date: 19-03-2008
Award name: Nao-Sena-Medal
Prize money: 2000.00
-----------------------------------------
Officer ID: 6
Date: 26-01-2010
Award name: Vir-Chakra
Prize money: 7000.00
-----------------------------------------
Officer ID: 5
Date: 26-01-2012
Award name: Sena-Medal
Prize money: 2000.00
-----------------------------------------
Officer ID: 7
Date: 29-03-2012
Award name: Mahavir-Chakra
Prize money: 10000.00
-----------------------------------------
Officer ID: 8
Date: 19-03-2014
Award name: Sena-Medal
Prize money: 2000.00
-----------------------------------------
Officer ID: 1
Date: 02-08-2015
Award name: Kriti-Chakra
Prize money: 9000.00
-----------------------------------------
Officer ID: 7
Date: 16-09-2016
Award name: Mahavir-Chakra

Prize money: 10000.00
-----------------------------------------
Officer ID: 6
Date: 15-09-2016
Award name: Vir-Chakra
Prize money: 7000.00
-----------------------------------------
Officer ID: 5
Date: 15-08-2016
Award name: Kriti-Chakra
Prize money: 9000.00
-----------------------------------------
Officer ID: 3
Date: 13-01-2016
Award name: Vir-Chakra
Prize money: 7000.00
-----------------------------------------
Officer ID: 3
Date: 19-03-2017
Award name: Nao-Sena-Medal
Prize money: 2000.00
-----------------------------------------
Officer ID: 8
Date: 26-10-2019
Award name: Ashoka-Chakra
Prize money: 12000.00
-----------------------------------------
Officer ID: 2
Date: 08-03-2020
Award name: Shaurya-Chakra
Prize money: 6000.00
-----------------------------------------
```

Manager side output screenshots

```
**********************************************
                Login Menu

Choose your designation
1.Manager
2.Level-3 or Level-4
3.Exit
**********************************************
1
Enter your username:
Kishore_V
Enter your password:
kishore2809
Logged in successfully!
**********************************************
                Menu

1.Initiate retirement
2.Initiate discharge
3.Initiate payroll
4.Initiate allowance
5.Exit
**********************************************
Enter a valid choice
1
Enter the ranking level of the officer requesting retirement:
1
Enter the ID of the officer:
1
Officer: Virat with Officer ID: 1 has been moved to veteran's list
Their pension amount is: 8000.00
Do you want to continue(Y/N)?
Y
```

```
Y
**********************************************
                Menu

1.Initiate retirement
2.Initiate discharge
3.Initiate payroll
4.Initiate allowance
5.Exit
**********************************************
Enter a valid choice
2
Enter the ranking level of the officer to be discharged:
1
Enter the ID of the officer:
2
Officer: Rahul with Officer ID: 2 has been moved to veteran's list
Their pension amount is: 8000.00
Do you want to continue(Y/N)?
Y
**********************************************
                Menu

1.Initiate retirement
2.Initiate discharge
3.Initiate payroll
4.Initiate allowance
5.Exit
**********************************************
Enter a valid choice
3
Salary paid for: 3 Mayank
Salary paid for: 4 Pujara
```

```
Salary paid for: 4 Pujara
Salary paid for: 5 Rahane
Salary paid for: 6 Rishabh
Salary paid for: 7 Ashwin
Salary paid for: 8 Bumrah
Pension paid for: 9 Shardul
Pension paid for: 10 Umesh
Pension paid for: 1 Virat
Pension paid for: 2 Rahul
Do you want to continue(Y/N)?
Y
*****************************************
                Menu

1.Initiate retirement
2.Initiate discharge
3.Initiate payroll
4.Initiate allowance
5.Exit
*****************************************
Enter a valid choice
4
Enter the ranking level of the officer requesting allowance:
2
Enter the ID of the officer:
3
Enter the allowance name to claim:
Flying-Allowance
Officer: 3 Mayank is eligible for this allowance
Flying-Allowance: 17300.0
Allowance claimed
Do you want to continue(Y/N)?
n
```

Level-3 Output

```
Do you want to log in(Y/N)
Y
*********************************************
                Login Menu

Choose your designation
1.Manager
2.Level-3 or Level-4
3.Exit
*********************************************
2
Enter your username:
Rahane_Ajjinkya
Enter your password:
rahane0606
Enter your ranking level:
3
Logged in successfully!
*****************************************
                Menu

1.Change username and password
2.Recruit candidate
3.Promote candidate
4.Exit
*****************************************
Enter a valid choice
1
Enter the new username:
Rahane_Ajjinkya
Enter the new password:
rahane0606
Credentials changed successfully!
```

```
Credentials changed successfully!
Do you want to continue(Y/N)
Y
*******************************************
                Menu

1.Change username and password
2.Recruit candidate
3.Promote candidate
4.Exit
*******************************************
Enter a valid choice
2
Enter the candinate's name
Shami
enter the wing name
Army
enter the location
Chennai
Officer: 11 Shami has been recruited to level one Army wing
Do you want to continue(Y/N)
Y
*******************************************
                Menu

1.Change username and password
2.Recruit candidate
3.Promote candidate
4.Exit
*******************************************
Enter a valid choice
3
enter the officer Id of level 1 to promote:
```

```
enter the officer Id of level 1 to promote:
11
Officer: 11 Shami has been promoted to level two Army wing
Do you want to continue(Y/N)
n
```

Level-4 View

```
Do you want to log in(Y/N)
Y
**********************************************
                Login Menu

Choose your designation
1.Manager
2.Level-3 or Level-4
3.Exit
**********************************************
2
Enter your username:
Ashwin_Ravi
Enter your password:
Ashwin1709
Enter your ranking level:
4
Logged in successfully!
**********************************************
                Menu

1.Promote candidate
2.Award candidate
3.Generate reports
4.Exit
**********************************************
Enter a valid choice
1
Enter the level of officer to promote:
2
enter the officer Id to promote:
4
Officer: 4 Pujara has been promoted to level three Navy wing
```

```
Officer: 4 Pujara has been promoted to level three Navy wing
Do you want to continue(Y/N)
Y
**********************************************
                Menu

1.Promote candidate
2.Award candidate
3.Generate reports
4.Exit
**********************************************
Enter a valid choice
2
Enter the level of officer to award:
3
enter the officer Id to award:
6
enter the award name
Vir-Chakra
enter the prize money
7000.00
Officer: 6 Rishabh has been awarded Vir-Chakra
They have received 7000.0
Do you want to continue(Y/N)
Y
**********************************************
                Menu

1.Promote candidate
2.Award candidate
3.Generate reports
4.Exit
**********************************************
```

```
Enter a valid choice
3
Enter the start date from which the report should be generated:
01-01-2000
Enter the date till which the report should be generated:
16-01-2022
Enter the search category
1. Award report
2. Transaction report
3. Retirement report
4.Promotion report
Enter a valid choice:
1
13
-----------------------------------------
Officer ID: 7
Date: 19-03-2008
Award name: Nao-Sena-Medal
Prize money: 2000.0
-----------------------------------------
Officer ID: 6
Date: 26-01-2010
Award name: Vir-Chakra
Prize money: 7000.0
-----------------------------------------
Officer ID: 5
Date: 26-01-2012
Award name: Sena-Medal
Prize money: 2000.0
-----------------------------------------
Officer ID: 7
Date: 29-03-2012
Award name: Mahavir-Chakra
```

```
Award name: Mahavir-Chakra
Prize money: 10000.0
-----------------------------------------
Officer ID: 8
Date: 19-03-2014
Award name: Sena-Medal
Prize money: 2000.0
-----------------------------------------
Officer ID: 1
Date: 02-08-2015
Award name: Kriti-Chakra
Prize money: 9000.0
-----------------------------------------
Officer ID: 7
Date: 16-09-2016
Award name: Mahavir-Chakra
Prize money: 10000.0
-----------------------------------------
Officer ID: 6
Date: 15-09-2016
Award name: Vir-Chakra
Prize money: 7000.0
-----------------------------------------
Officer ID: 5
Date: 15-08-2016
Award name: Kriti-Chakra
Prize money: 9000.0
-----------------------------------------
Officer ID: 3
Date: 13-01-2016
Award name: Vir-Chakra
Prize money: 7000.0
-----------------------------------------
```

```
Award name: Vir-Chakra
Prize money: 7000.0
----------------------------------------
Officer ID: 3
Date: 19-03-2017
Award name: Nao-Sena-Medal
Prize money: 2000.0
----------------------------------------
Officer ID: 8
Date: 26-10-2019
Award name: Ashoka-Chakra
Prize money: 12000.0
----------------------------------------
Officer ID: 6
Date: 15-01-2022
Award name: Vir-Chakra
Prize money: 7000.0
----------------------------------------
Do you want to continue(Y/N)
Y
*******************************************
                Menu

1.Promote candidate
2.Award candidate
3.Generate reports
4.Exit
*******************************************
Enter a valid choice
3
Enter the start date from which the report should be generated:
01-01-2000
Enter the date till which the report should be generated:
```

```
Enter the date till which the report should be generated:
16-01-2022
Enter the search category
1. Award report
2. Transaction report
3. Retirement report
4.Promotion report
Enter a valid choice:
2
12
----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 25000.0
Type: Salary
----------------------------------------
Officer ID: 4
Date: 15-01-2022
Bank account number: 4
Amount: 25000.0
Type: Salary
----------------------------------------
Officer ID: 5
Date: 15-01-2022
Bank account number: 5
Amount: 30000.0
Type: Salary
----------------------------------------
Officer ID: 6
Date: 15-01-2022
Bank account number: 6
Amount: 30000.0
```

```
Amount: 30000.0
Type: Salary
-----------------------------------------
Officer ID: 7
Date: 15-01-2022
Bank account number: 7
Amount: 35000.0
Type: Salary
-----------------------------------------
Officer ID: 8
Date: 15-01-2022
Bank account number: 8
Amount: 35000.0
Type: Salary
-----------------------------------------
Officer ID: 9
Date: 15-01-2022
Bank account number: 9
Amount: 0.0
Type: Pension
-----------------------------------------
Officer ID: 10
Date: 15-01-2022
Bank account number: 10
Amount: 0.0
Type: Pension
-----------------------------------------
Officer ID: 1
Date: 15-01-2022
Bank account number: 1
Amount: 8000.0
Type: Pension
-----------------------------------------
```

```
-----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 17300.0
Type: Allowance
-----------------------------------------
Officer ID: 6
Date: 15-01-2022
Bank account number: 6
Amount: 7000.0
Type: Award
-----------------------------------------
Do you want to continue(Y/N)
Y
*****************************************
                Menu

1.Promote candidate
2.Award candidate
3.Generate reports
4.Exit
*****************************************
Enter a valid choice
3
Enter the start date from which the report should be generated:
01-01-2000
Enter the date till which the report should be generated:
16-01-2022
Enter the search category
1. Award report
2. Transaction report
3. Retirement report
```

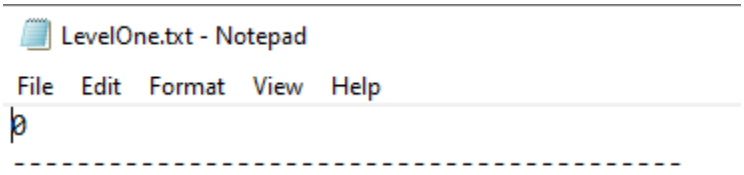
```
3. Retirement report
4.Promotion report
Enter a valid choice:
3
4
-----------------------------------------
Officer ID: 9
Date: 30-12-2015
Level during retirement: 1
-----------------------------------------
Officer ID: 10
Date: 30-12-2016
Level during retirement: 2
-----------------------------------------
Officer ID: 1
Date: 15-01-2022
Level during retirement: 1
-----------------------------------------
Officer ID: 2
Date: 15-01-2022
Level during retirement: 1
-----------------------------------------
Do you want to continue(Y/N)
Y
*********************************************
                Menu

1.Promote candidate
2.Award candidate
3.Generate reports
4.Exit
*********************************************
```

```
4.Exit
*********************************************
Enter a valid choice
3
Enter the start date from which the report should be generated:
01-01-2000
Enter the date till which the report should be generated:
16-01-2022
Enter the search category
1. Award report
2. Transaction report
3. Retirement report
4.Promotion report
Enter a valid choice:
4
2
-----------------------------------------
Officer ID: 11
Date: 15-01-2022
Next level: 2
-----------------------------------------
Officer ID: 4
Date: 15-01-2022
Next level: 3
-----------------------------------------
Do you want to continue(Y/N)
n
Do you want to log in(Y/N)
```

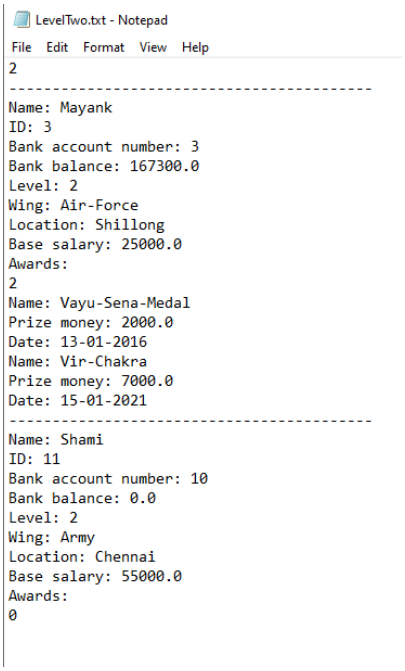
Output Files

Name	Date modified	Type	Size
AwardRecord.txt	15-01-2022 05:57 PM	Text Document	2 KB
AwardRecordReport.txt	15-01-2022 05:47 PM	Text Document	2 KB
LevelFour.txt	15-01-2022 05:57 PM	Text Document	1 KB
LevelOne.txt	15-01-2022 05:57 PM	Text Document	1 KB
LevelThree.txt	15-01-2022 05:57 PM	Text Document	2 KB
LevelTwo.txt	15-01-2022 05:57 PM	Text Document	1 KB
Manager.txt	15-01-2022 05:57 PM	Text Document	1 KB
PromotionRecord.txt	15-01-2022 05:57 PM	Text Document	1 KB
PromotionRecordReport.txt	15-01-2022 05:49 PM	Text Document	1 KB
RetirementRecord.txt	15-01-2022 05:57 PM	Text Document	1 KB
RetirementRecordReport.txt	15-01-2022 05:48 PM	Text Document	1 KB
TransactionRecord.txt	15-01-2022 05:57 PM	Text Document	2 KB
TransactionRecordReport.txt	15-01-2022 05:47 PM	Text Document	2 KB
Veteran.txt	15-01-2022 05:57 PM	Text Document	1 KB

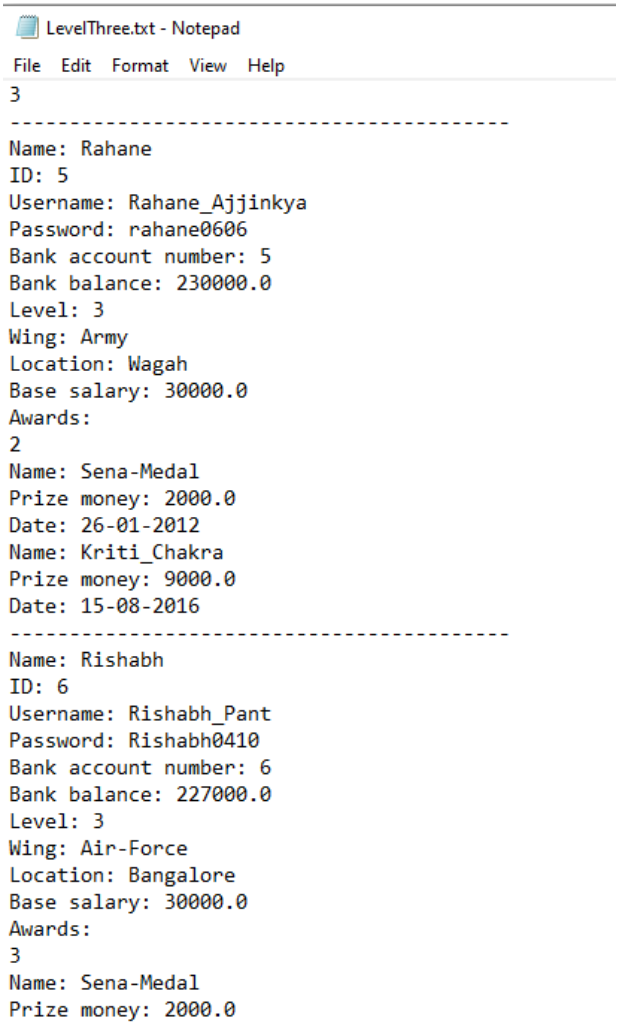
LevelOne.txt



LevelTwo.txt



LevelThree.txt


LevelThree.txt - Notepad

File Edit Format View Help

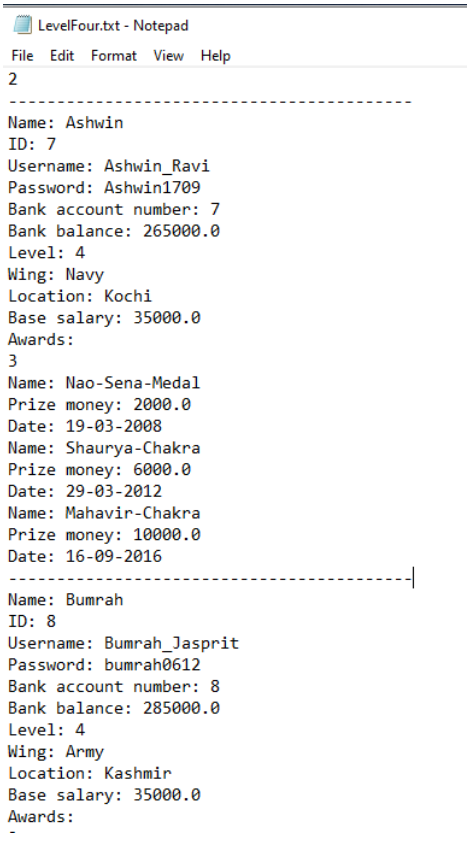
```
3
----------------------------------------
Name: Rahane
ID: 5
Username: Rahane_Ajjinkya
Password: rahane0606
Bank account number: 5
Bank balance: 230000.0
Level: 3
Wing: Army
Location: Wagah
Base salary: 30000.0
Awards:
2
Name: Sena-Medal
Prize money: 2000.0
Date: 26-01-2012
Name: Kriti_Chakra
Prize money: 9000.0
Date: 15-08-2016
----------------------------------------
Name: Rishabh
ID: 6
Username: Rishabh_Pant
Password: Rishabh0410
Bank account number: 6
Bank balance: 227000.0
Level: 3
Wing: Air-Force
Location: Bangalore
Base salary: 30000.0
Awards:
3
Name: Sena-Medal
Prize money: 2000.0
```

LevelThree.txt - Notepad

File Edit Format View Help

```
Bank account number: 6
Bank balance: 227000.0
Level: 3
Wing: Air-Force
Location: Bangalore
Base salary: 30000.0
Awards:
3
Name: Sena-Medal
Prize money: 2000.0
Date: 26-01-2010
Name: Vir_Chakra
Prize money: 7000.0
Date: 15-09-2016
Name: Vir-Chakra
Prize money: 7000.0
Date: 15-01-2022
-----------------------------------------
Name: Pujara
ID: 4
Username: Pujara
Password: 4
Bank account number: 4
Bank balance: 140000.0
Level: 3
Wing: Navy
Location: Mumbai
Base salary: 60000.0
Awards:
1
Name: Nao-Sena-Medal
Prize money: 2000.0
Date: 19-03-2017
-----------------------------------------
```

LevelFour.txt

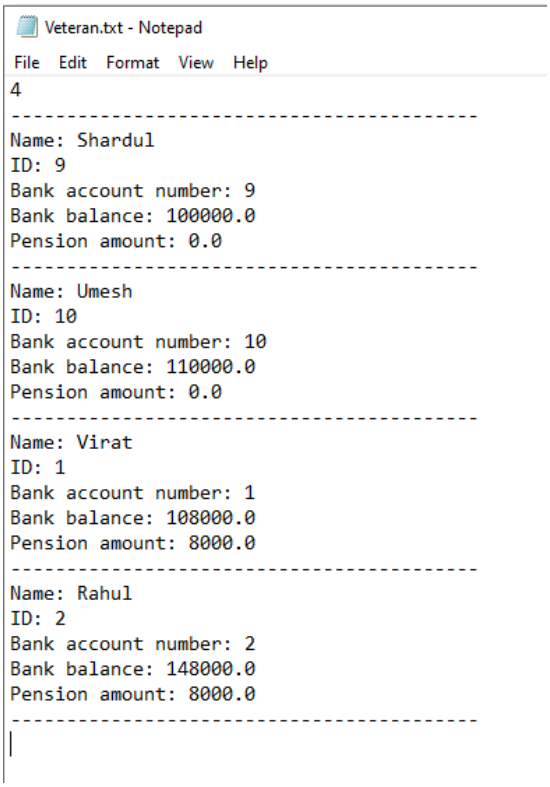


```
LevelFour.txt - Notepad
File  Edit  Format  View  Help
2
----------------------------------------
Name: Ashwin
ID: 7
Username: Ashwin_Ravi
Password: Ashwin1709
Bank account number: 7
Bank balance: 265000.0
Level: 4
Wing: Navy
Location: Kochi
Base salary: 35000.0
Awards:
3
Name: Nao-Sena-Medal
Prize money: 2000.0
Date: 19-03-2008
Name: Shaurya-Chakra
Prize money: 6000.0
Date: 29-03-2012
Name: Mahavir-Chakra
Prize money: 10000.0
Date: 16-09-2016
----------------------------------------
Name: Bumrah
ID: 8
Username: Bumrah_Jasprit
Password: bumrah0612
Bank account number: 8
Bank balance: 285000.0
Level: 4
Wing: Army
Location: Kashmir
Base salary: 35000.0
Awards:
-
```

Manager.txt

```
Manager.txt - Notepad
File  Edit  Format  View  Help
4
----------------------------------------
Name: Kishore
ID: 1
Username: Kishore_V
Password: kishore2809
----------------------------------------
Name: Ashwath
ID: 2
Username: Ashwath_VA
Password: ashwath2107
----------------------------------------
Name: Anish
ID: 3
Username: Anish_Ayyagari
Password: anish1207
----------------------------------------
Name: Yashasvi
ID: 4
Username: Chowta_Yashasvi
Password: yashasvi0101
----------------------------------------
11
```

Veteran.txt


Veteran.txt - Notepad
File Edit Format View Help
4

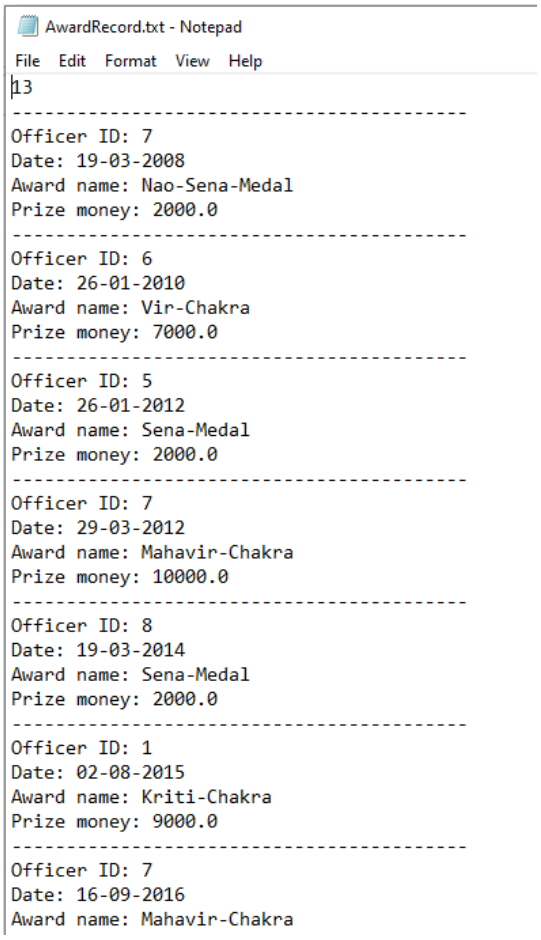
Name: Shardul
ID: 9
Bank account number: 9
Bank balance: 100000.0
Pension amount: 0.0

Name: Umesh
ID: 10
Bank account number: 10
Bank balance: 110000.0
Pension amount: 0.0

Name: Virat
ID: 1
Bank account number: 1
Bank balance: 108000.0
Pension amount: 8000.0

Name: Rahul
ID: 2
Bank account number: 2
Bank balance: 148000.0
Pension amount: 8000.0

AwardRecord.txt



AwardRecord.txt - Notepad

File Edit Format View Help

```
Date: 16-09-2016
Award name: Mahavir-Chakra
Prize money: 10000.0
----------------------------------------
Officer ID: 6
Date: 15-09-2016
Award name: Vir-Chakra
Prize money: 7000.0
----------------------------------------
Officer ID: 5
Date: 15-08-2016
Award name: Kriti-Chakra
Prize money: 9000.0
----------------------------------------
Officer ID: 3
Date: 13-01-2016
Award name: Vir-Chakra
Prize money: 7000.0
----------------------------------------
Officer ID: 3
Date: 19-03-2017
Award name: Nao-Sena-Medal
Prize money: 2000.0
----------------------------------------
Officer ID: 8
Date: 26-10-2019
Award name: Ashoka-Chakra
Prize money: 12000.0
----------------------------------------
Officer ID: 6
Date: 15-01-2022
Award name: Vir-Chakra
Prize money: 7000.0
----------------------------------------
```

PromotionRecord.txt

PromotionRecord.txt - Notepad

File Edit Format View Help

```
2
----------------------------------------
Officer ID: 11
Date: 15-01-2022
Next level: 2
----------------------------------------
Officer ID: 4
Date: 15-01-2022
Next level: 3
----------------------------------------
```

RetirementRecord.txt

RetirementRecord.txt - Notepad

File Edit Format View Help

```
4
----------------------------------------
Officer ID: 9
Date: 30-12-2015
Level during retirement: 1
----------------------------------------
Officer ID: 10
Date: 30-12-2016
Level during retirement: 2
----------------------------------------
Officer ID: 1
Date: 15-01-2022
Level during retirement: 1
----------------------------------------
Officer ID: 2
Date: 15-01-2022
Level during retirement: 1
----------------------------------------
```

TransactionRecord.txt

TransactionRecord.txt - Notepad

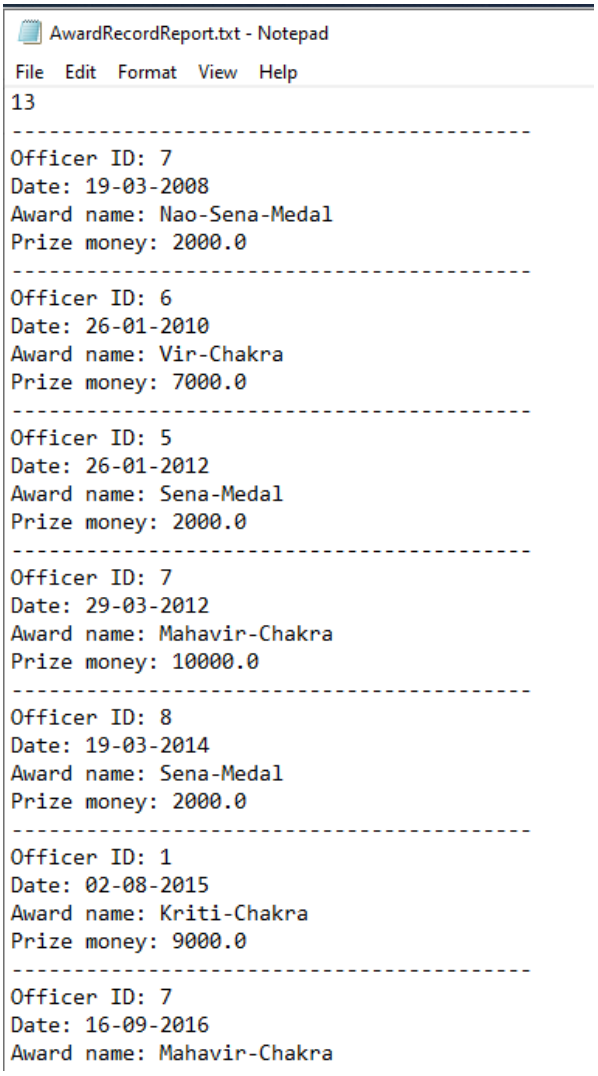
File Edit Format View Help

```
12
----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 25000.0
Type: Salary
----------------------------------------
Officer ID: 4
Date: 15-01-2022
Bank account number: 4
Amount: 25000.0
Type: Salary
----------------------------------------
Officer ID: 5
Date: 15-01-2022
Bank account number: 5
Amount: 30000.0
Type: Salary
----------------------------------------
Officer ID: 6
Date: 15-01-2022
Bank account number: 6
Amount: 30000.0
Type: Salary
----------------------------------------
Officer ID: 7
Date: 15-01-2022
Bank account number: 7
Amount: 35000.0
Type: Salary
----------------------------------------
Officer ID: 8
Date: 15-01-2022
Bank account number: 8
```

```
Date: 15-01-2022
Bank account number: 8
Amount: 35000.0
Type: Salary
----------------------------------------
Officer ID: 9
Date: 15-01-2022
Bank account number: 9
Amount: 0.0
Type: Pension
----------------------------------------
Officer ID: 10
Date: 15-01-2022
Bank account number: 10
Amount: 0.0
Type: Pension
----------------------------------------
Officer ID: 1
Date: 15-01-2022
Bank account number: 1
Amount: 8000.0
Type: Pension
----------------------------------------
Officer ID: 2
Date: 15-01-2022
Bank account number: 2
Amount: 8000.0
Type: Pension
----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 17300.0
Type: Allowance
----------------------------------------
```

```
----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 17300.0
Type: Allowance
----------------------------------------
Officer ID: 6
Date: 15-01-2022
Bank account number: 6
Amount: 7000.0
Type: Award
----------------------------------------
```

AwardRecordReport.txt


AwardRecordReport.txt - Notepad
File Edit Format View Help
13

Officer ID: 7
Date: 19-03-2008
Award name: Nao-Sena-Medal
Prize money: 2000.0

Officer ID: 6
Date: 26-01-2010
Award name: Vir-Chakra
Prize money: 7000.0

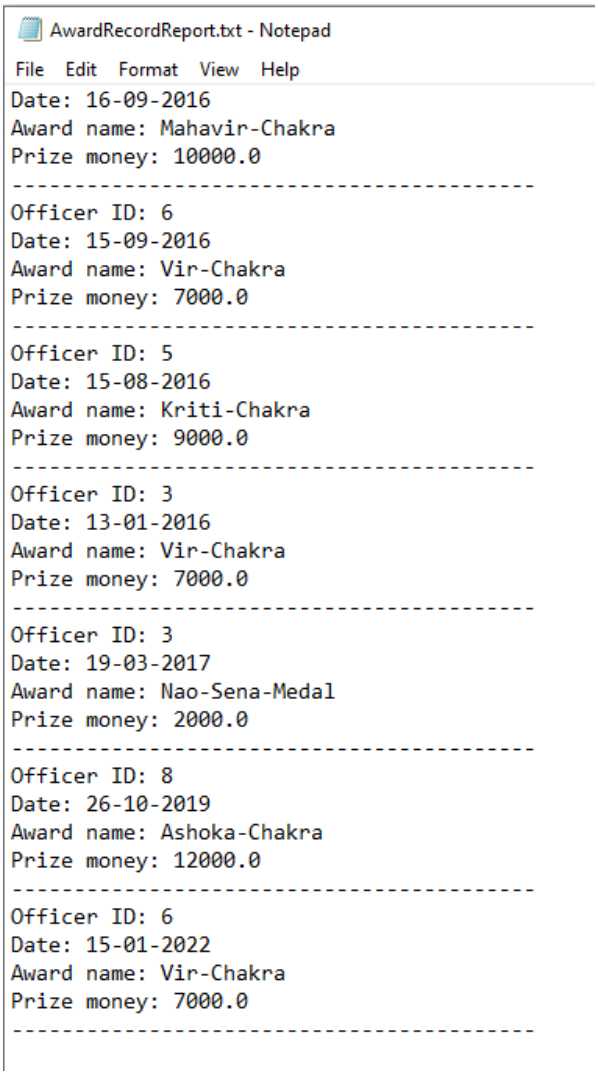
Officer ID: 5
Date: 26-01-2012
Award name: Sena-Medal
Prize money: 2000.0

Officer ID: 7
Date: 29-03-2012
Award name: Mahavir-Chakra
Prize money: 10000.0

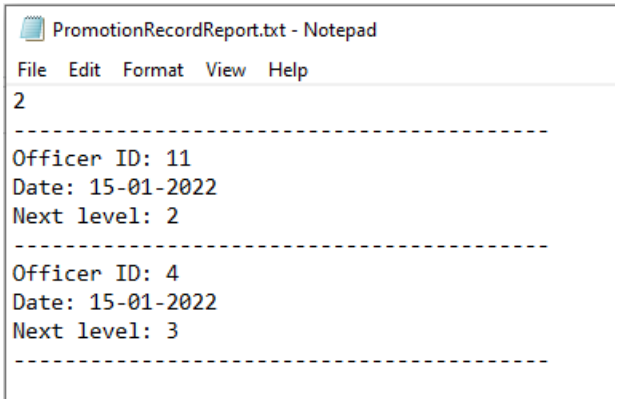
Officer ID: 8
Date: 19-03-2014
Award name: Sena-Medal
Prize money: 2000.0

Officer ID: 1
Date: 02-08-2015
Award name: Kriti-Chakra
Prize money: 9000.0

Officer ID: 7
Date: 16-09-2016
Award name: Mahavir-Chakra



PromotionRecordReport.txt



RetirementRecord.txt

RetirementRecordReport.txt - Notepad

File Edit Format View Help

```
4
-----------------------------------------
Officer ID: 9
Date: 30-12-2015
Level during retirement: 1
-----------------------------------------
Officer ID: 10
Date: 30-12-2016
Level during retirement: 2
-----------------------------------------
Officer ID: 1
Date: 15-01-2022
Level during retirement: 1
-----------------------------------------
Officer ID: 2
Date: 15-01-2022
Level during retirement: 1
-----------------------------------------
```

TransactionRecordReport.txt

TransactionRecordReport.txt - Notepad

File Edit Format View Help

```
12
-----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 25000.0
Type: Salary
-----------------------------------------
Officer ID: 4
Date: 15-01-2022
Bank account number: 4
Amount: 25000.0
Type: Salary
-----------------------------------------
Officer ID: 5
Date: 15-01-2022
Bank account number: 5
Amount: 30000.0
Type: Salary
-----------------------------------------
Officer ID: 6
Date: 15-01-2022
Bank account number: 6
Amount: 30000.0
Type: Salary
-----------------------------------------
Officer ID: 7
Date: 15-01-2022
Bank account number: 7
Amount: 35000.0
Type: Salary
-----------------------------------------
Officer ID: 8
Date: 15-01-2022
Bank account number: 8
```

```
Bank account number: 8
Amount: 35000.0
Type: Salary
-----------------------------------------
Officer ID: 9
Date: 15-01-2022
Bank account number: 9
Amount: 0.0
Type: Pension
-----------------------------------------
Officer ID: 10
Date: 15-01-2022
Bank account number: 10
Amount: 0.0
Type: Pension
-----------------------------------------
Officer ID: 1
Date: 15-01-2022
Bank account number: 1
Amount: 8000.0
Type: Pension
-----------------------------------------
Officer ID: 2
Date: 15-01-2022
Bank account number: 2
Amount: 8000.0
Type: Pension
-----------------------------------------
Officer ID: 3
Date: 15-01-2022
Bank account number: 3
Amount: 17300.0
Type: Allowance
-----------------------------------------
Officer ID: 6
```

```
-----------------------------------------
Officer ID: 6
Date: 15-01-2022
Bank account number: 6
Amount: 7000.0
Type: Award
-----------------------------------------
```