COMP1161 PROJECT PART 1

Due February ~~24~~ 25, 2025.

OVERVIEW

The new Director of Security(**DoS**) in the nation of Jamrock has been tasked with coordinating efforts to stem an unprecendented crime wave. While there is significant experience in the mounting of joint police-military **Operation**s to shut down **Criminal** strongholds, current resource challenges, as well as the mobility of bad actors seem to demand a more targeted approach. The **DoS** currently has trust of well thinking **Community** members who are **Resident**s however, who are willing to give vital information on the number of **Criminal** elements operating in the area. Depending on a tactical assessment and availability, a **Raid,** a **ZOSO,** or a **SOE** may be executed to perform arrests to support rehabilitation. Conversely, resource availability may indicate that an **UnderCover** operation, or postponement of action, is advised.

SUBMISSION INSTRUCTIONS

Your submission will be made on OurVLE, . All files in your submission will be placed in a directory of the name <yourIDNumber>.Project1 and then the directory will be zipped. The zip file will be submitted. As an example, if your ID number is 620011011, the zip file should be named **620011011.Project1.zip**

==========

PART A. Business Rules

==========

In this context, a **Raid**, is an **Operation** that deploys a specific number of police officers to a **Community**. A **ZOSO** (Zone of Special Operation) is an **Operation** that deploys a number of soldiers and one police officer(whose task is to ensure compliance with the civilian legal system). A **SOE** (State of Emergency) employs an identical tactical deployment to a **ZOSO**, but has additional social components in feeding, counselling and vocational training programs(ie., a **SOE** is a specialized **ZOSO**). The social services in a **SOE** are supported by civilian specialists, and an additional police officer is allocated as a liaison. An **Undercover Operation** requires less resources , but still needs support personnel. Resources for all **Operation**s are managed by the **Services** class. In order to allow internal communication, as well as maintain the trust, **DoS** is required to produce at least two reports. The first is a summarized report for distribution to policy makers, and the second is a classified internal operation brief for ground commanders.

Architecture

The main **Driver** class loads data into a **DoS** and then populates **Community** and **Resident** objects. A **Resident** that observes a **Criminal** in the **Community** makes a report, and the **Community** keeps track of any **Resident** or **Criminal** associated with it. **DoS** keeps track of each **Community**. After collecting all reports, **DoS** determines if there is a suitable **Operation** based on business rules and availability. Availability is managed by the **Services** class. Under the influence of the **Driver** class, **DoS** loads data from a test file, instantiates **Services**, tracks it's data, assesses each **Community** for the appropriate **Operation**, and then provides it's reports. Sample data in the file

testData.txt is provided with the starting code. A properly implemented solution will give the output below when the default testData.txt file is used as the input.

```
=================SUMMARY FOR POLICY MAKERS============================
We expect 3 operation(s), yielding 8 arrest(s) and 1 rehab(s).

=================CLASSIFIED INTERNAL OPERATION BRIEF============================
Operation DS0 to be deployed as a Raid in Maverly.
Expect 1 arrest(s).
Operation DS1 to be deployed as a SOE in HillTop.
Expect 5 arrest(s) and 1 rehabilitation(s).
Operation DS2 to be deployed as a ZOSO in HillSide.
Expect 2 arrest(s).
Operation DS3 to be deployed as an undercover surveillance operation in Farside.
```

==========

PART B. Starting code

==========

Code the following classes have been provided: **Community, Criminal, DoS, Driver, Operation, Raid, Resident, Services, UnderCover.** Some methods are not fully implemented. Incomplete methods are described in section C. Methods available are described below

THE COMMUNITY CLASS

The public methods in the **Community** class are

1. **public Community(String name)-** Instantiates the **Community**.
2. **public int countResidents() -** Returns the number of **Residents** in the **Community**
3. **public int countCriminals()** - Returns the number of **Criminals** in the **Community**
4. **public ArrayList<Criminal> getCriminals()-** Returns a list of **Criminals** in the **Community**
5. **public Resident getResident(String name)-** Returns a **Resident** whose name matches the supplied name. Adds the **Resident** if does not yet exist.
6. **public Criminal getCriminal(String name)-** Returns a **Criminal** whose name matches the supplied name. Adds the **Criminal** if does not yet exist.
7. **public String getName()** - Returns the name of the **Community**
8. **public String toString()** - String representation of the **Community**

THE CRIMINAL CLASS

The **Criminal** class has the following public methods

1. **public Criminal(String name)** - Initializes the Criminal, given a name
2. **public void arrest()** - arrests the criminal
3. **public String getName()** - returns the name of the Criminal
4. **public void rehabilitate()** - rehabilitates the Criminal
5. **public String toString()**- returns a string representation

THE DoS CLASS

The **DoS** class exposes the following public methods:

1. **public DoS(int forceMultiplier, int gangLimit,int rehabRate, int numSoldiers,int numEquipment,int numPolice, int numSocial, int numSupplies)** - Initializes the **DoS**
2. **public Services getService()** - returns a reference to the **Services** class

3. **public int getForceMultiplier() -** returns the expected ratio of security personnel to criminals .
4. **public int getGangLimit()** - returns the minimum number of criminals in a gang
5. **public Community getCommunity(String cname)** - returns a Community, given a name
6. **public void addCommunity(String cname)** - adds a Community, given a name
7. **public void assessForOps() - INCOMPLETE** - Assesses for viability of **Operations**
8. **public void publicPolicyReport()- INCOMPLETE** - Prints publicly policy report
9. **public void classifiedInformationBrief()- INCOMPLETE** - Prints classified information brief

THE DRIVER CLASS

The Driver class has one public method, namely

public static void main(String[] args) - Reads data from file, assesses for ops then prints reports.

THE OPERATION CLASS

Public methods in **Operation** are

1. **public Operation(Community community)** - Instantiates an operation in a community
2. **public String getCallSign()** - Returns a call sign for the operation

THE RAID CLASS

The following methods are publicly available from **Raid**

1. **public Raid(Community community) - INCOMPLETE** - Instantiates a **Raid** in a **Community.**
2. **public static boolean canDeploy(int numCriminalsAdjusted)-**Tests if resources are available.
3. **public int countArrests() -INCOMPLETE –** Returns number of **Criminals** to be arrested.
4. **public String toString()-**Returns a String representation of the **Raid.**

THE RESIDENT CLASS

Resident offers the following methods

1. **public Resident (String name, Community comm)** - Instantiates a **Resident** in a community.
2. **public String getName()** - Returns the name of the **Resident.**
3. **private String getCommunity()**-**Returns the **Community** in which the **Resident** lives.
4. **public void recordReport(String criminalName)** - Reports a **Criminal** to the **Community.**
5. **public String toString()-** Returns the name of the **Resident.**

THE SERVICES CLASS

Methods in **Services** are:

1. **private Services(int numSoldiers,int numEquipment,int numPolice, int numSocial, int numSupplies)** - Constructor is private so only one Service will be created
2. **public static Services getService(int numSoldiers,int numEquipment,int numPolice, int numSocial, int numSupplies)** -returns a reference to the Service
3. **public static Services getService()**-returns a reference to the Service
4. **public boolean policeAvailable(int req)** - indicates if enough police are available
5. **public boolean soldiersAvailable(int req)-** indicates if enough soldiers are available
6. **public void deployPolice(int numNeeded)-** Models deployment by reducing police availability
7. **public void deploySoldiers(int numNeeded)-** Reduces soldier availability

THE UNDERCOVER CLASS

Public methods in **Undercover** are:
public UnderCover(Community community) - Instantiates an Undercover Operation
public static boolean canDeploy() - Tests if resources are avalable for deployment
public String toString() - returns a string representation of the Undercover Operation

==========
SECTION C. Code you are required to write
==========
You are required to write code to complete the classes **DoS** and **Raid,** and **Services**. You will also implement the classes **ZOSO,** and **SOE**. Do not change the given starting code for other given classes. Otherwise you are free to select the implementation strategies that meet the requirements.

Completing the Raid class*(5 marks)*

The Raid class is partly implemented, as a subclass of the Operation class. Make the following changes to the Raid class:

- Add an integer value that can record the number of arrests in a raid.
- In the constructor of the Raid class, update the number of arrests to match the number of criminals in the community of interest.
- Write a constructor that takes as arguments a **Community** and a multiplier. It first initializes the superclass, then gets a reference to the **Service**. Other tasks involve modelling the deployment of a number a police officers that is equal to the product of the multiplier and the number of criminals in the **Community**, then invoking the arrest method of each **Criminal** in the **Community**. The constructor should also make sure that the number of arrests match the number of criminals in the community.
- Adjust the method countArrests should so that it returns the number of arrests in the raid.
- Call the arrest method on each criminal in the target community.

Completing the Services class*(15 marks)*

The Services class is partly implemented. It tracks the number of resources available to support tactical and social services. Make the following changes to implement it:

- Add an integer attribute that is able to store the number of social workers available.
- Add an integer attribute that is able to store the number of supply kits for that can be distributed by social workers.
- Add public method called **socialAvailable** that takes an integer argument which represents a deployment size. It returns true if both the number of supply kits and the number of social workers are greater than the deployment size, and false otherwise.
- Add a public method called **deploySocial** that records the deployment of social services. The method should accept the expected deployment size as an argument. The method will then reduce the number of social workers available, as well as the number of supplies available by the deployment size. It will also reduce the number of police on available by 1.

Implementing the ZOSO class*(25 marks)*

You are required to implement **ZOSO** as an **Operation** by performing the following actions:

- Create an integer value to record the number of arrests in a **ZOSO**.
- Write a public method called **canDeploy** that does not require an instance, and returns true or false which accepts a required deployment size as an argument, and returns true only if the specified number of military personnel, along with one police officer, is available. The **canDeploy** method first gets a reference to the Service. It then checks the service to see the number of soldiers requested is available, and if one police officer is available. If both conditions are true, **canDeploy** returns true, else **canDeploy** returns false.
- Write a constructor that takes as arguments a **Community** and a multiplier. It first initializes the superclass, then gets a reference to the **Service**. Other tasks involve modelling the deployment of a number a soldiers that is equal to the product of the multiplier and the number of criminals in the **Community**, then invoking the arrest method of each **Criminal** in the **Community**. The constructor should also make sure that the number of arrests match the number of criminals in the community.
- Write a public method called **countArrests** that returns the number of arrests.
- Write a **toString** method that returns a String in the following format: "Operation <callsign> to be deployed as a **ZOSO** in <communityName> \nExpect <numberOfArrests> arrest(s).";

IMPLEMENTING THE SOE CLASS*(25 marks)*

You are required to implement **SOE** as a **ZOSO** by performing the following actions:

- Create an integer value to record the number of rehabilitations supported by the **SOE**.
- Create a public method called **canDeploy** that does not require an instance, and returns true or false which accepts a required deployment size as an argument. **canDeploy** returns true only if the number of military personnel with equipment, the number of social workers with supplies, and two police officers are available, and false otherwise.
- Write a constructor that accepts a **Community**, a multiplier(int), and a rehabrate rate (int). After initializing the superclass, the constructor then evaluates expected the number of rehabilitations, where the rehabrate is a percentage of the number of **Criminal**s in the **Community**. The constructor then removes a number of **Criminal**s from the **Community** that matches the rehabrate, makes them residents (ie, if the rehabrate is 20, and 10 **Criminals** are in the list, then 2 **Criminal**s will be rehabilitated. As a result 2 **Criminal**s are removed from the list of **Criminal**s. For each removal, a **Resident** is created with the name of the former **Criminal**).
- Write a public method **countRehabs** that returns the number of criminals that are expected to be rehabilitated.

- Write a **toString** method that shows the callsign of the operation, the community name and the number of criminals arrested. The number of rehabilitations, if any, should be displayed. An example of the output of **SOE toString** with 1 rehabilitation is as follows:

  Operation DS1 to be deployed as a SOE in HillTop.
  Expect 5 arrest(s) and 1 rehabilitation(s).
  If there are no rehabilitations, the output should end after presenting the number of arrests.

COMPLETING THE DOS CLASS*(30 marks)*

A Raid is useful to arrest individual Criminals, but risk collateral damage if a gang is encountered. Active responses to gang threats will therefore be managed using either a ZOSO or a SOE. In order to handle gangs, the logic to manage ZOSOs and SOEs need to be implemented. Three updates are required.

1. Currently method **assessForOps** only manages situations where the number of criminals is below the gang limit. To manage other cases, test if the number of criminals is less than 30% of the population (30% is stored as a variable known as the emergency Ratio). If the number of criminals is less than the emergency ratio, test if resources exist to implement a ZOSO. If yes, deploy the ZOSO, else, if the resources do not exist, if an Undercover Operation can be deployed, then deploy the Undercover operation, otherwise do nothing.

If the number of criminals is greater than or equal to the emergency ratio, test if resources exist to implement a SOE. If yes, deploy the SOE, else, if the resources do not exist, if an Undercover Operation can be deployed, then deploy the Undercover operation, otherwise do nothing.

2. . The method **publicPolicyReport** first prints a header in the format:

"==========SUMMARY FOR POLICY MAKERS=========="

It then counts the number of Raids, ZOSOs and SOEs that have been approved, by looking at entries the ArrayList ops. For each operation it evaluates the number of arrests and the number of rehabs, if applicable. It then presents a summary report of the format "We expect " + totOps +" operation(s), yielding " +totArrests + " arrest(s)". If the number of rehabs is greater than 0, it also prints “and"+ totRehabs +" rehab(s)"; Note that information on Undercover information is not represented in the policy summary.

3. The method **classifiedInformationBrief** first prints a header in the format:

"==========CLASSIFIED INTERNAL OPERATION BRIEF=========="

It then prints the result of the toString method of each operation in the list ops in new line

---

OPTIONAL READING :
The process in context.

On system initialization, all instances of **Resident** with data to report will submit a report in the **Community** giving a name. A **Resident** may have the same name in a different **Community**, however two instances of the same name in the same **Community** are interpreted to belong to the same person*(logic already implemented)*. **DoS** will then assess the most appropriate **Operation** using the following logic...

1. A Community will be considered in the sequence in which any **Resident** of the **Community** make a report
2. The size of the tactical team is to be a multiple of the expected number of **Criminal**s to be confronted. The value of the multiple provided by the private **forceMultiplier** attribute on the **DoS**. As an example, if **forceMultiplier** has value 3, and the expected number of **Criminal**s is 5, the deployment size is 15. The logic is that if a **Criminal** group is of the impression that overwhelming force is not present, it is more likely to engage in a shootout, increasing the possibility of casualities. **DoS** is resolved to avoid casualities if possible. **forceMultiplier** is specified as a parameter of a test case.
3. If the number of **Criminal**s is less than the minimum number considered to constitute a gang then a **Raid** is the preferred solution. Note that
    a. Only police officers are used for a **Raid.**
    b. If a **Raid** is preferred however sufficient resources do not exist for deployment, **DoS** will seek to deploy an **UnderCover** surveillance operation.
    c. After planning a **Raid**, the resources deployed are not longer available from the **Services**.
    d. A **Raid** is expected to result in the arrest of each **Criminal** in the **Community**.
    e. The number of arrests is stored on the **Raid**.
    f. An **UnderCover** surveillance operation requires two police personnel. If resources are not available for the **UnderCover** operation, no immediate action is taken.
4. If the number of **Criminals** is greater than or equal to the minimum gang limit, but less than 0.3*the number of **Residents**, a **ZOSO** is the preferred operation type. Note that
    a. A **ZOSO** uses one police officer as a liaison, but the tactical team is made up of soldiers.
    b. Each soldier needs to be equipped, so deployment of a soldier also requires deployment of a unit of equipment from the **Services**.
    c. If a **ZOSO** is preferred however sufficient resources do not exist to deploy for the **ZOSO**, **DoS** will seek to deploy an **UnderCover** surveillance operation. The same constraints apply to **UnderCover** operations as stated above.
    d. After planning a **ZOSO**, the resources deployed are not longer available from the Services.
    e. A **ZOSO** is expected to result in the arrest of each **Criminal** in the **Community**.
    f. The number of arrests is stored on the **ZOSO.**
5. If the number of **Criminal**s is greater than the minimum gang limit, and greater than or equal to 0.3*the number of **Residents**, a **SOE** is the preferred operation type. Note that
    a. A **SOE** applies an identical tactical approach to a **ZOSO**.

b. A **SOE** requires a number of social workers that is equal to the number of soldiers deployed, as well as one additional police officer.
c. Each social worker needs supplies, so deployment of a social worker also requires deployment of a unit of supplies from the **Services**.
d. If a **SOE** is preferred however sufficient resources do not exist to deploy for the SOE, **DoS** will seek to deploy an **UnderCover** surveillance operation. The same constraints apply to **UnderCover** operations as stated above.
e. After planning a **SOE**, the resources deployed are not longer available from the **Services**.
f. A **SOE** is expected to result in the arrest of each **Criminal** in the **Community**.
g. A **SOE** allows a number of **Criminal**s to be rehabilitated, where the number is determined by (int)(numCriminals*rehabRate/100).
    i. A **Criminal** that is rehabilitated, is added to the **Community** of interest as a **Resident**, and then removed from the set of **Criminal**s.
    ii. The number of rehabilitations is stored on the **SOE**