Course: Object Oriented Programming with C++

1. Title: Design Activity document of Open-Ended Assessment
   Armed Forces

2. Team members list with roll numbers

   Isha Bhandary: 01FE18BCS063(63)
   Atul Kumar : 01FE18BCS056(56)

3. **Problem Definition (Description)**

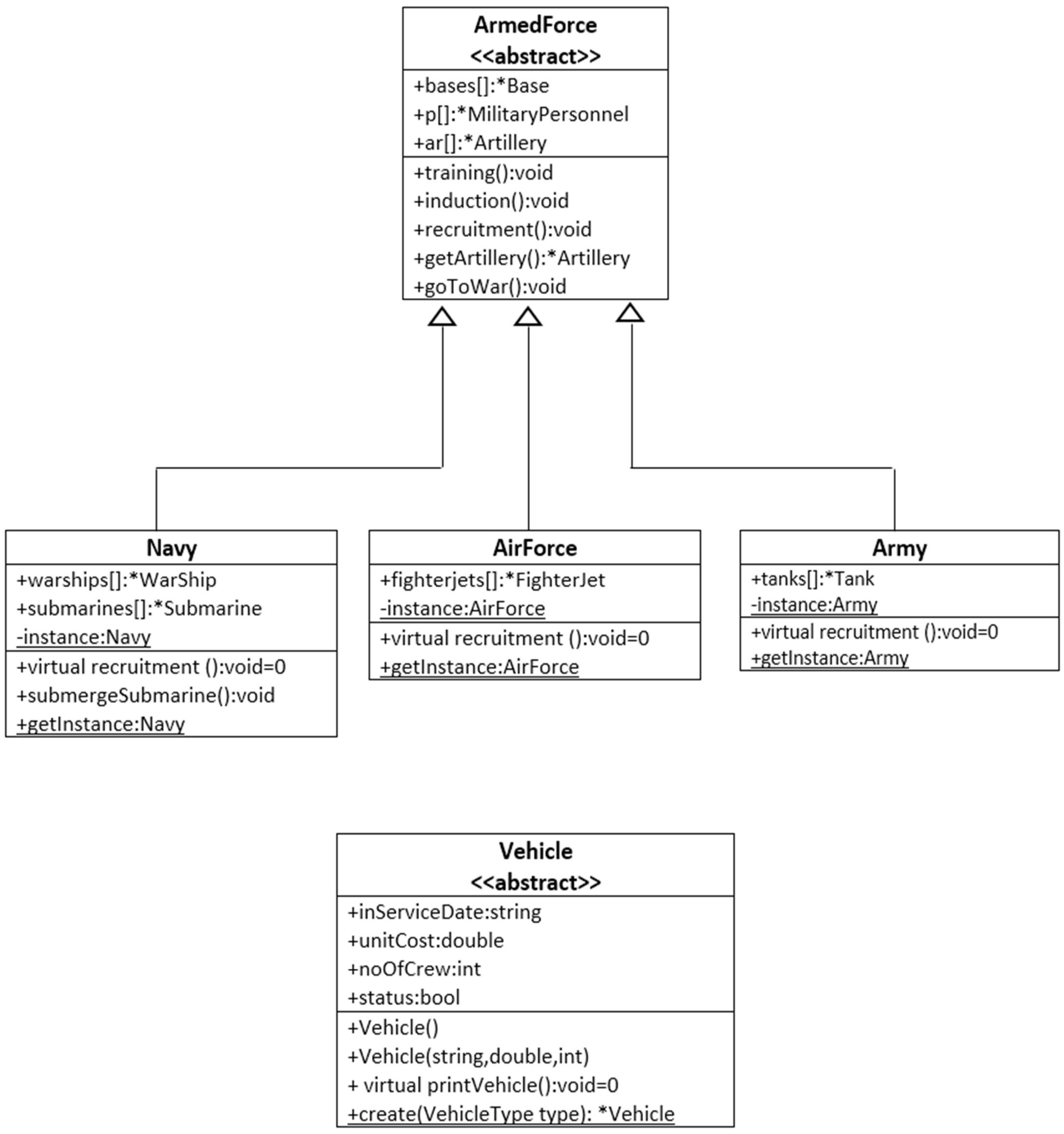
   The Indian Armed Forces consist of different divisions: Indian Air Force, Indian Army and Indian Navy. All these divisions have bases all over the country. They consist of military personnel, artillery and military vehicles. Each of these forces conduct a recruitment drive after which the selected individuals are trained and inducted in that force. During wartime all these forces engage in combat. Our application will provide information about the military personnel, bases all over the country and the forces which are engaged in war and other activities. The recruitment drive is conducted after fixed intervals of time during which only fixed number of individuals are selected.

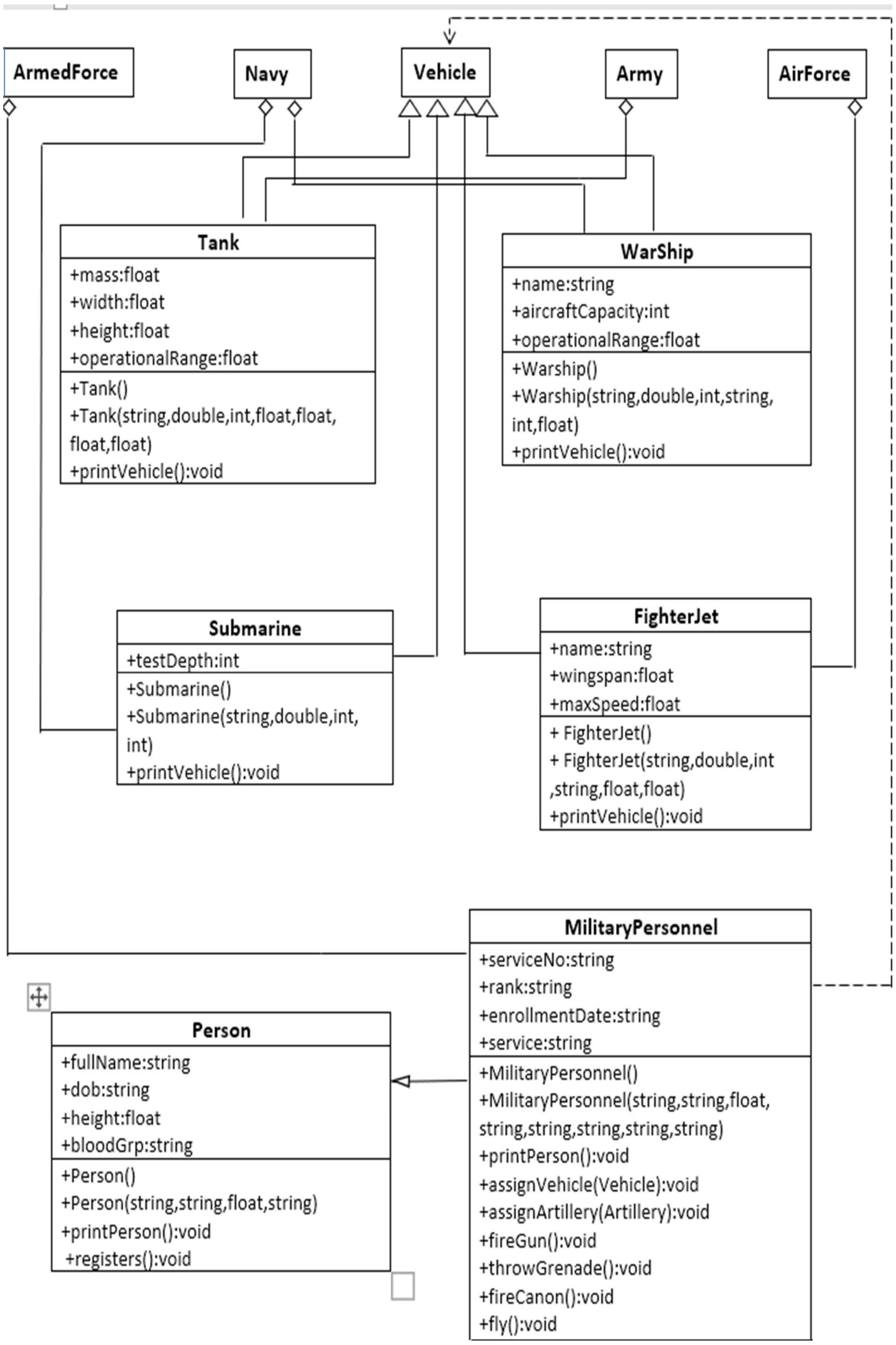
4. **List of objects identified**

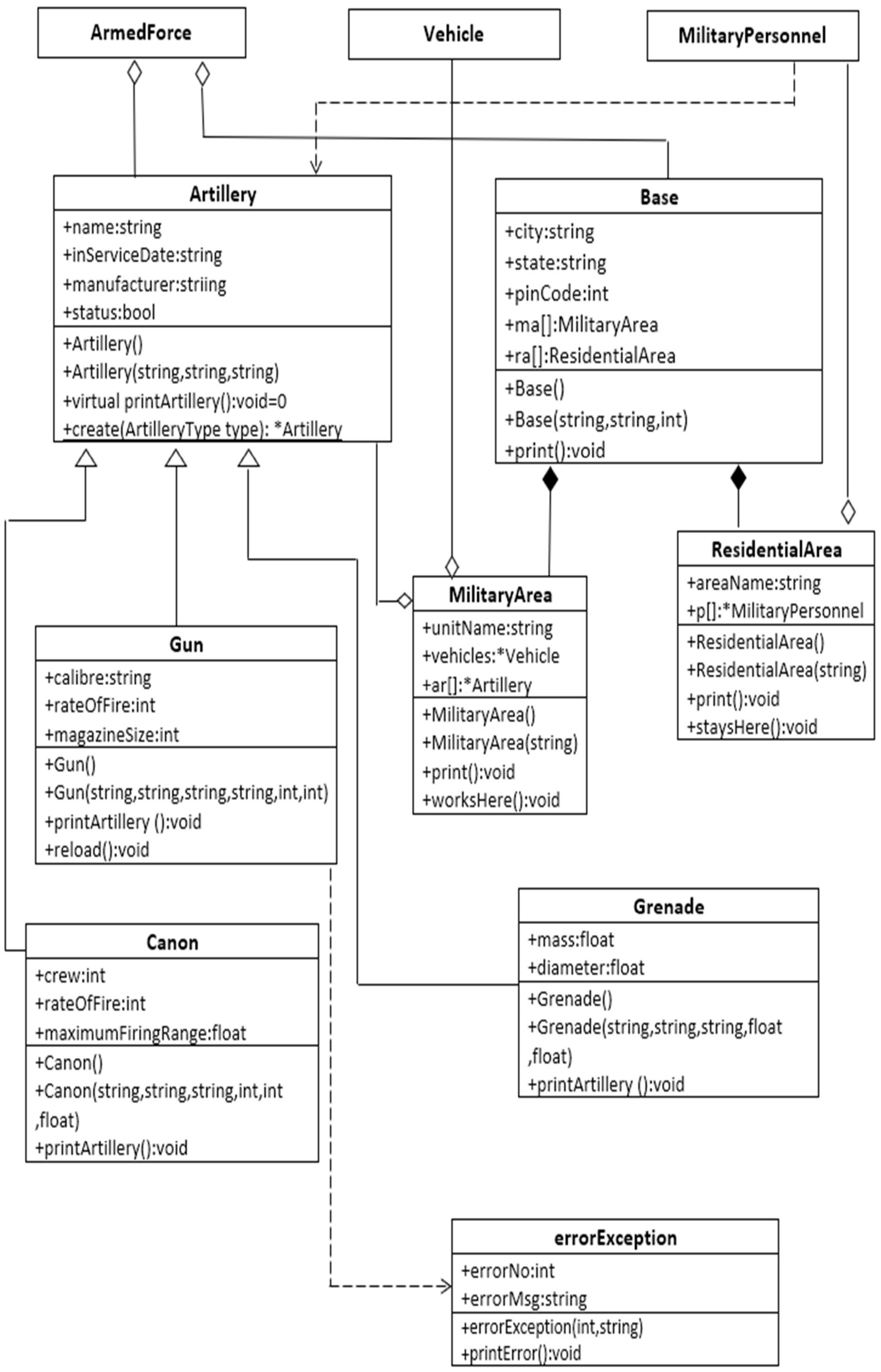
   - ArmedForce
   - Navy
   - Army
   - AirForce
   - Vehicle
   - Tank
   - FighterJet
   - WarShip
   - Submarine
   - Person
   - MilitaryPersonnel
   - Artillery
   - Gun
   - Canon
   - Grenade
   - Base
   - ResidentialArea

- MilitaryArea
- errorException

5. Class Diagram




ArmedForce
Navy
Vehicle
Army
AirForce
Tank
+mass:float
+width:float
+height:float
+operationalRange:float
+Tank()
+Tank(string,double,int,float,float,
float,float)
+printVehicle():void
WarShip
+name:string
+aircraftCapacity:int
+operationalRange:float
+Warship()
+Warship(string,double,int,string,
int,float)
+printVehicle():void
Submarine
+testDepth:int
+Submarine()
+Submarine(string,double,int,
int)
+printVehicle():void
FighterJet
+name:string
+wingspan:float
+maxSpeed:float
+ FighterJet()
+ FighterJet(string,double,int
,string,float,float)
+printVehicle():void
MilitaryPersonnel
+serviceNo:string
+rank:string
+enrollmentDate:string
+service:string
+MilitaryPersonnel()
+MilitaryPersonnel(string,string,float,
string,string,string,string,string)
+printPerson():void
+assignVehicle(Vehicle):void
+assignArtillery(Artillery):void
+fireGun():void
+throwGrenade():void
+fireCanon():void
+fly():void
Person
+fullName:string
+dob:string
+height:float
+bloodGrp:string
+Person()
+Person(string,string,float,string)
+printPerson():void
+registers():void


ArmedForce
Vehicle
MilitaryPersonnel
Artillery
+name:string
+inServiceDate:string
+manufacturer:striing
+status:bool
+Artillery()
+Artillery(string,string,string)
+virtual printArtillery():void=0
+create(ArtilleryType type): *Artillery
Base
+city:string
+state:string
+pinCode:int
+ma[]:MilitaryArea
+ra[]:ResidentialArea
+Base()
+Base(string,string,int)
+print():void
ResidentialArea
+areaName:string
+p[]:*MilitaryPersonnel
+ResidentialArea()
+ResidentialArea(string)
+print():void
+staysHere():void
MilitaryArea
+unitName:string
+vehicles:*Vehicle
+ar[]:*Artillery
+MilitaryArea()
+MilitaryArea(string)
+print():void
+worksHere():void
Gun
+calibre:string
+rateOfFire:int
+magazineSize:int
+Gun()
+Gun(string,string,string,string,int,int)
+printArtillery ():void
+reload():void
Grenade
+mass:float
+diameter:float
+Grenade()
+Grenade(string,string,string,float ,float)
+printArtillery ():void
Canon
+crew:int
+rateOfFire:int
+maximumFiringRange:float
+Canon()
+Canon(string,string,string,int,int ,float)
+printArtillery():void
errorException
+errorNo:int
+errorMsg:string
+errorException(int,string)
+printError():void

6. Description of each class

ArmedForce:-

ArmedForce   **<<abstract>>**
+bases[]:*Base   +p[]:*MilitaryPersonnel   +ar[]:*Artillery
+training():void   +induction():void   +virtual recruitment():void=0   +fireGun():void   +throwGrenade():void   +fireCanon():void   +getArtillery():*Artillery

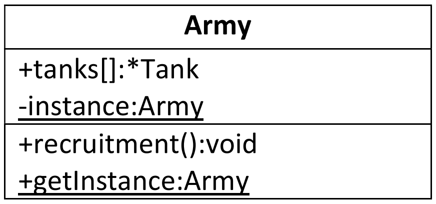
- This is an Abstract class where we have a pure virtual function "recruitment()" which is used for recruiting persons into a particular force if they clear the recruitment process.
- It acts like a factory to create new artillery through artillery class
- It has 3 sub-classes: Navy, Army & AirForce
- The artillery Array should start with 1 and further it should be auto incremented for each artillery purchased.

Navy

Navy
+warships[]:*WarShip   +submarines[]:*Submarine   -instance:Navy
+recruitment():void   +submergeSubmarine():void   +getInstance:Navy

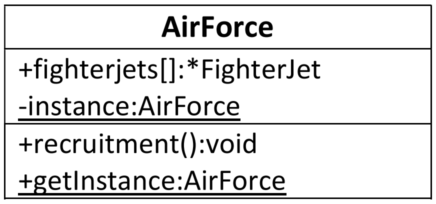
- It is a singleton class as it has only one object
- It has an array of warships and submarines which are created through factory design pattern in vehicle class
- The warships and submarines Array should start with 1 and further it should be auto incremented for each warship or submarine purchased.

Army



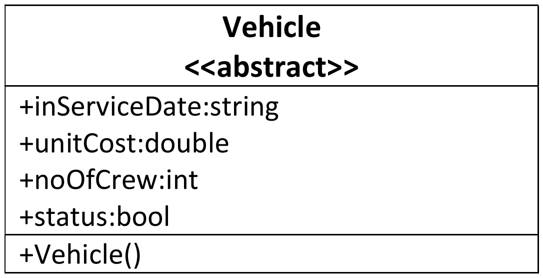
- It is a singleton class as it has only one object
- It has an array of tanks which are created through factory design pattern in vehicle class
- The tank Array should start with 1 and further it should be auto incremented for each tank purchased.

AirForce



- It is a singleton class as it has only one object
- It has an array of fighter jets which are created through factory design pattern in vehicle class
- The fighterjets Array should start with 1 and further it should be auto incremented for each fighterjet purchased.

Vehicle



+Vehicle(string,double,int) + virtual printVehicle():void=0 +create(VehicleType type): *Vehicle

- This is an Abstract class where we have a pure virtual function “printVehicle()”
- It creates the type of vehicle as per demands of the user
- It has 4 sub-classes: Tank, Submarine, FighterJet & WarShip
- Status attribute indicates if it is available or taken(0=available and 1=taken)

Tank, WarShip, FighterJet, Submarine

Tank
+mass:float +width:float +height:float +operationalRange:float
+Tank() +Tank(string,double,int,float,float, float,float) +printVehicle():void

WarShip
+name:string +aircraftCapacity:int +operationalRange:float
+Warship() +Warship(string,double,int,string, int,float) +printVehicle():void

FighterJet
+name:string +wingspan:float +maxSpeed:float
+ FighterJet() + FighterJet(string,double,int ,string,float,float) +printVehicle():void

Submarine
+testDepth:int
+Submarine() +Submarine(string,double,int, int) +printVehicle():void

- The above classes are used to create objects of vehicle type using factory design pattern.
- aircraftCapacity in WarShip tells us how many aircrafts it can carry

Artillery

Artillery
+name:string +inServiceDate:string

+manufacturer:striing +status: bool

+Artillery()
+Artillery(string, string, string)
+virtual printArtillery():void=0
+create(ArtilleryType type): *Artillery

- This is an Abstract class where we have a pure virtual function “printArtillery()”
- It creates the type of artillery as per demands of the factory(ArmedForce)
- It has 4 sub-classes: Tank, Submarine, FighterJet & WarShip
- Status attribute indicates if it is available or taken(0=available and 1=taken)

Gun , Canon , Grenad

Gun
+calibre:string +rateOfFire:int +magazineSize:int
+Gun() +Gun(string, string, string, string, int , int) +printArtillery ():void +reload():void

Grenade
+mass:float +diameter:float

+Grenade()
+Grenade(string,string,string,float ,float)
+printArtillery ():void

Canon
+crew:int +rateOfFire:int +maximumFiringRange:float

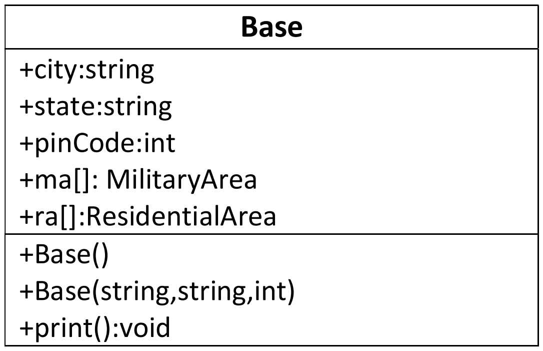
+Canon()
+Canon(string,string,string,int,int ,float)
+printArtillery():void

- The above classes are used to create objects of artillery type using factory design pattern
- Attribute magazineSize in Gun refers to number of bullets in the gun which will keep getting decremented each time the gun is fired.

Validation:

➔ Verify magazineSize in Gun should be positive value. If it becomes negative display appropriate error message.

Base

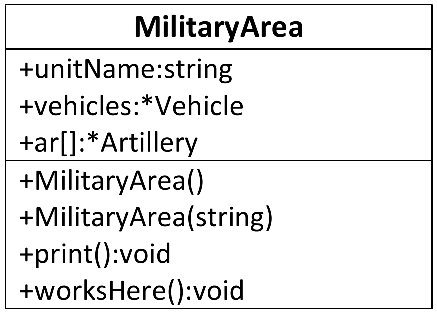


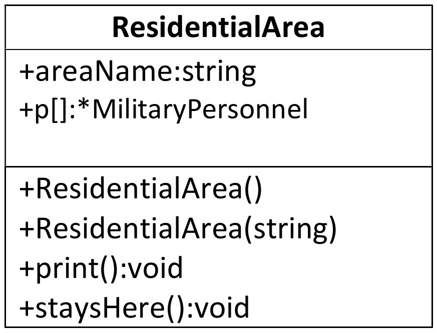
- This class describes bases all over the country

Validation

- The zip-code should be valid otherwise appropriate error message should be shown.
- A base has military areas and residential areas.

ResidentialArea , MilitaryArea





- The above classes aggregate from base class
- MilitaryArea has array of vehicles and artillery and ResidentialArea has array of military personnel

Person

Person
+fullName:string +dob:string +height:float +bloodGrp:string

+Person()
+Person(string,string,float,string)
+printPerson():void
+registers():void

- It is base class for military personnel
- A person can register for the recruitment process and if he is selected he gets inducted into the armed force.

MilitaryPersonnel

MilitaryPersonnel
+serviceNo:string +rank:string +enrollmentDate:string +service:string
+MilitaryPersonnel() +MilitaryPersonnel(string,string,float, string,string,string,string,string) +printPerson():void +assignVehicle(Vehicle):void +assignArtillery(Artillery):void

- This class inherits from Person class and uses the artillery and vehicles
- They reside in residential areas

Validation

- The serviceNo should be valid otherwise appropriate error message should be shown

Exception class:-

errorException
+errorNo:int +errorMsg:string
+errorException(int,string) +printError():void

- This class is used to handle all the exception cases and negative cases throughout the code.

7. Main Function

In main function we will create a single object of Army, AirForce and Navy. According to the user input we will create a Vehicle. The Base, MilitaryArea and ResidentialArea will be read from file. Array of MilitaryPersonnel is created as well as array of Vehicle is created. Using Army, Navy and AirForce objects any of the functions of ArmedForces can be called. Any errors are handled through the instantiation of the object of errorException class.

8. Use of Standard Design Patterns

Factory design pattern and singleton pattern are used for the above application.

Factory method pattern(creational pattern):-

• Definition:- The Factory Method pattern is a design pattern used to define a runtime interface for creating an object. It's called a factory because it creates various types of objects without necessarily knowing what kind of object it creates or how to create it.

• Usage:- Factory method is suitable for this scenario because vehicle objects are to be created as per user demand, so to create the objects of the required vehicle type during the run time interface it becomes easier. Similarly artillery are to be created as per the requirements of the armed forces.

Singleton Pattern(creational pattern)

• Definition:- This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

Usage:- Singleton pattern is suitable for this scenario because there can be only one navy,army and airforce in the country