	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 3**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Jangle Parth **Roll no: 22BCP083**			

Objective: To familiarize students with standard Creational design patterns.

Experiment: Explain the builder design pattern and write a program using any object-oriented programming language to demonstrate the working of builder design pattern.

Theory:

Imagine a Scenario where you need different types of tanks and each tank is composed of Many Elements like manhole , sprayball, airvent... . Now It is not necessary that a Company Has concern with all the parameters like he wants only diff manhole rest all things are required to be standard then also he has to enter every configuration to solve this problem Builder Design Pattern is used . And it may happen that two company may make same types of tank means they have all components same Just the implementation of All might be different

In builder Pattern, Separate the construction of a complex object from its representation so that the same construction process can create different representations. It is used to construct a complex object step by step and the final step will return the object. The process of constructing an object should be generic so that it can be used to create different representations of the same object.

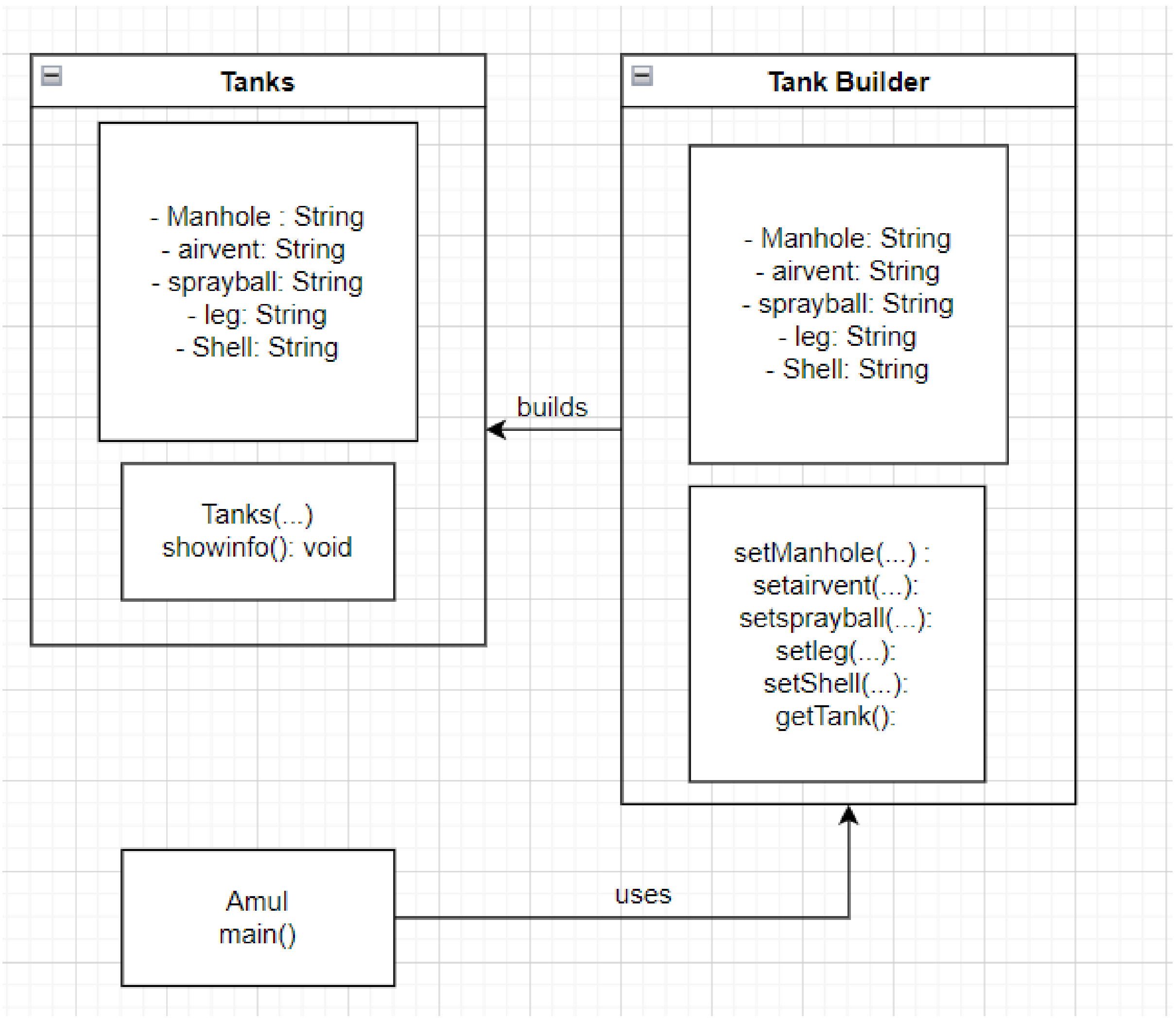
In Factory Patten,

Problem Statement Explanation:

We have a Class Tank which is the Product of the Company it has attributes as Manhole , airvent ,shell ,sprayball ,leg etc. it has a method show info which

Tells about all the parameters . Then we have a TankBuilder class Which is Responsible to set all the objects it has methods like Set Manhole , Set airvent… etc. and al Last a Method get Tank which return a Tank object with all the parameters provided.

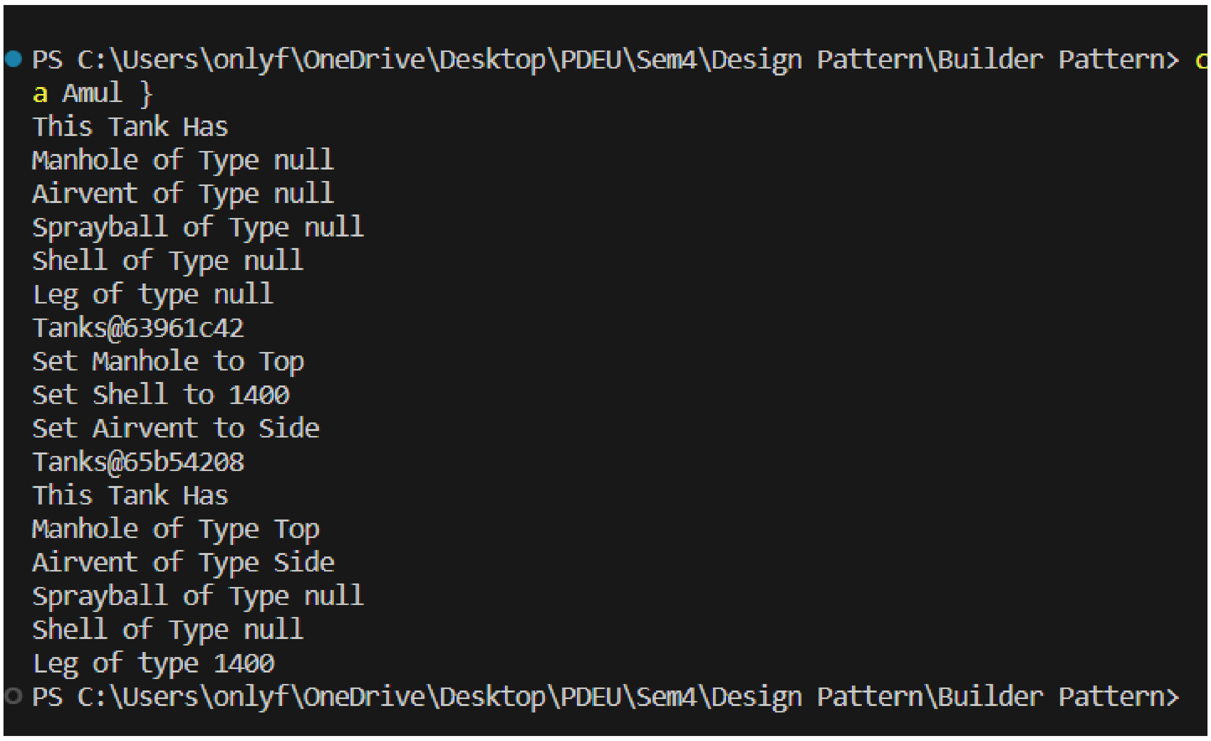
Flowchart Explanation:



Code:

```
class Tanks {
    String Manhole;
    String airvent;
    String sprayball;
    String leg;
    String Shell;

    public Tanks(String pManhole, String pairvent, String pspryaball, String
pleg, String pShell) {
        this.Manhole = pManhole;
        this.Shell = pShell;
        this.airvent = pairvent;
        this.leg = pleg;
        this.sprayball = pspryaball;
    }

    public void showinfo() {
        System.out.println("This Tank Has");
        System.out.println("Manhole of Type " + Manhole);
        System.out.println("Airvent of Type " + airvent);
        System.out.println("Spraybaall of Type " + sprayball);
        System.out.println("Shell of Type " + Shell);
        System.out.println("Leg of type " + leg);
    }

}

class TankBuilder {

    String Manhole;
    String airvent;
    String sprayball;
    String leg;
    String Shell;

    public TankBuilder setManhole(String pManhole) {
        System.out.println("Set Manhole to " + pManhole);
        this.Manhole = pManhole;
        return this;
    }

    public TankBuilder setairvent(String pairvent) {
        System.out.println("Set Airvent to " + pairvent);
        this.airvent = pairvent;
        return this;
    }

    public TankBuilder setsprayball(String psprayball) {
        System.out.println("Set Sprayball to " + psprayball);
        this.sprayball = psprayball;
        return this;
    }
    public TankBuilder setleg(String pleg) {
        System.out.println("Set leg to " + pleg);
        this.leg = pleg;
        return this;
    }

    public TankBuilder setShell(String pShell) {
        System.out.println("Set Shell to " + pShell);
        this.Shell = pShell;
        return this;
    }

    public Tanks getTank() {
        return new Tanks(Manhole, airvent, sprayball, Shell, leg);
    }

}

public class Amul {
    public static void main(String[] args) {
        TankBuilder atpl = new TankBuilder();
        Tanks Storage = atpl.getTank();
        Storage.showinfo();
        System.out.println(Storage);
        Tanks HMST =
atpl.setManhole("Top").setShell("1400").setairvent("Side").getTank();
        System.out.println(HMST);
        HMST.showinfo();

    }

}
```

Output:



```
PS C:\Users\onlyf\OneDrive\Desktop\PDEU\Sem4\Design Pattern\Builder Pattern> c
a Amul }
This Tank Has
Manhole of Type null
Airvent of Type null
Sprayball of Type null
Shell of Type null
Leg of type null
Tanks@63961c42
Set Manhole to Top
Set Shell to 1400
Set Airvent to Side
Tanks@65b54208
This Tank Has
Manhole of Type Top
Airvent of Type Side
Sprayball of Type null
Shell of Type null
Leg of type 1400
PS C:\Users\onlyf\OneDrive\Desktop\PDEU\Sem4\Design Pattern\Builder Pattern>
```