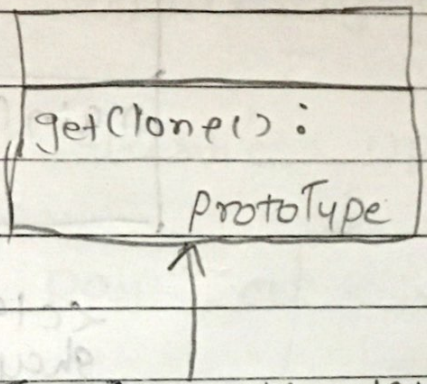
18/7.

* DP.

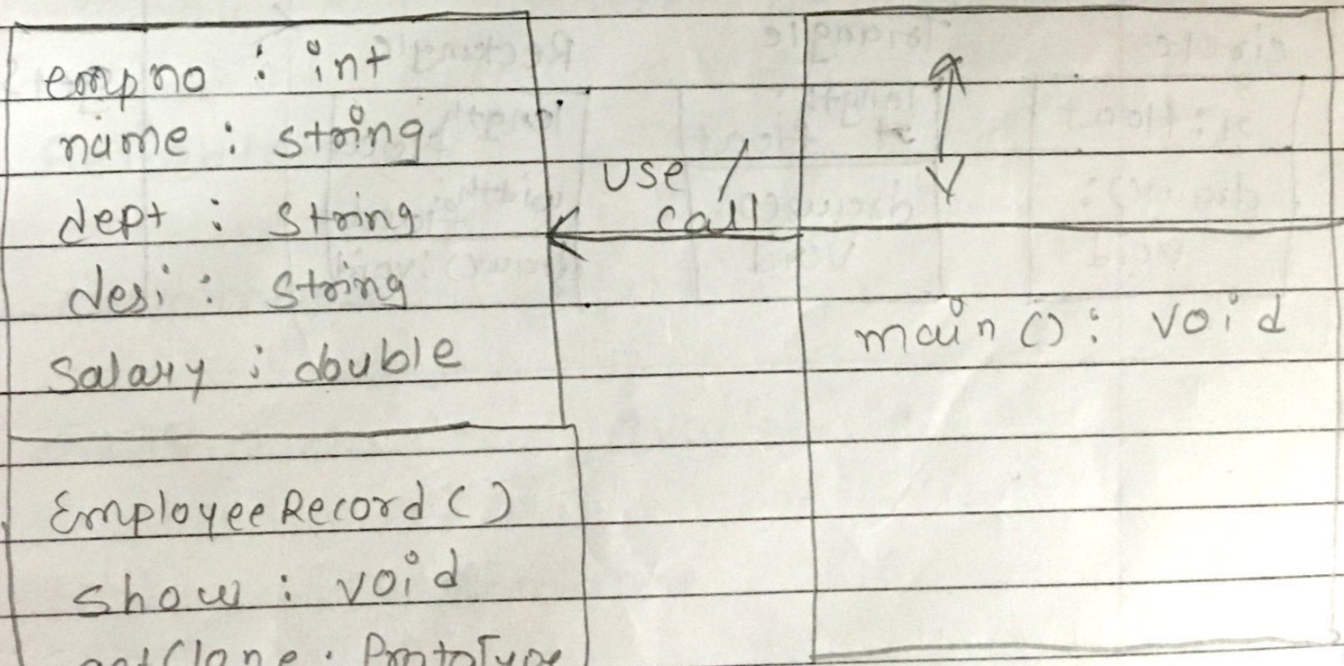
* Prototype :-

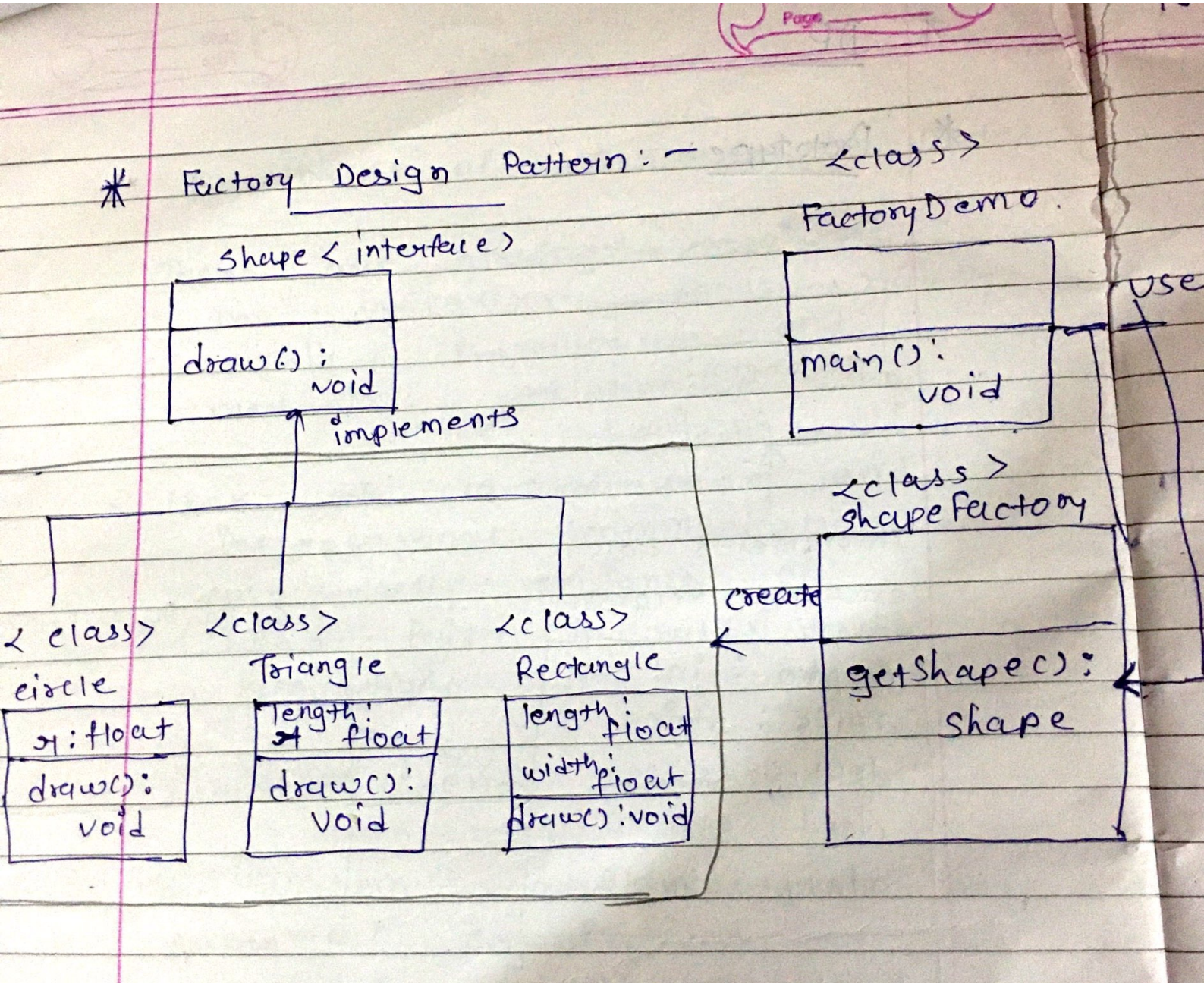
Prototype <interface>



Implements class
employee Records.

Ex of Prototype





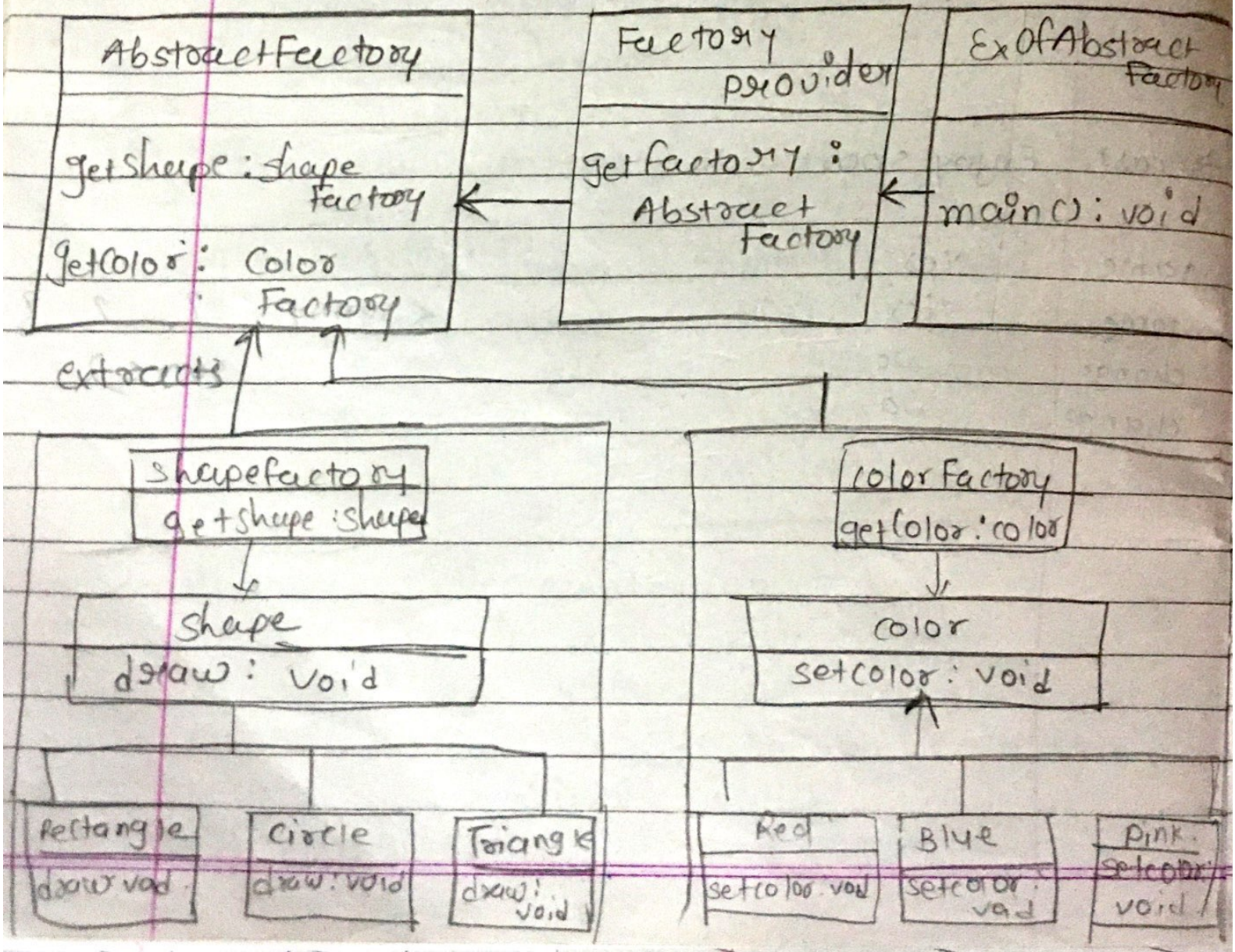
2417

DP

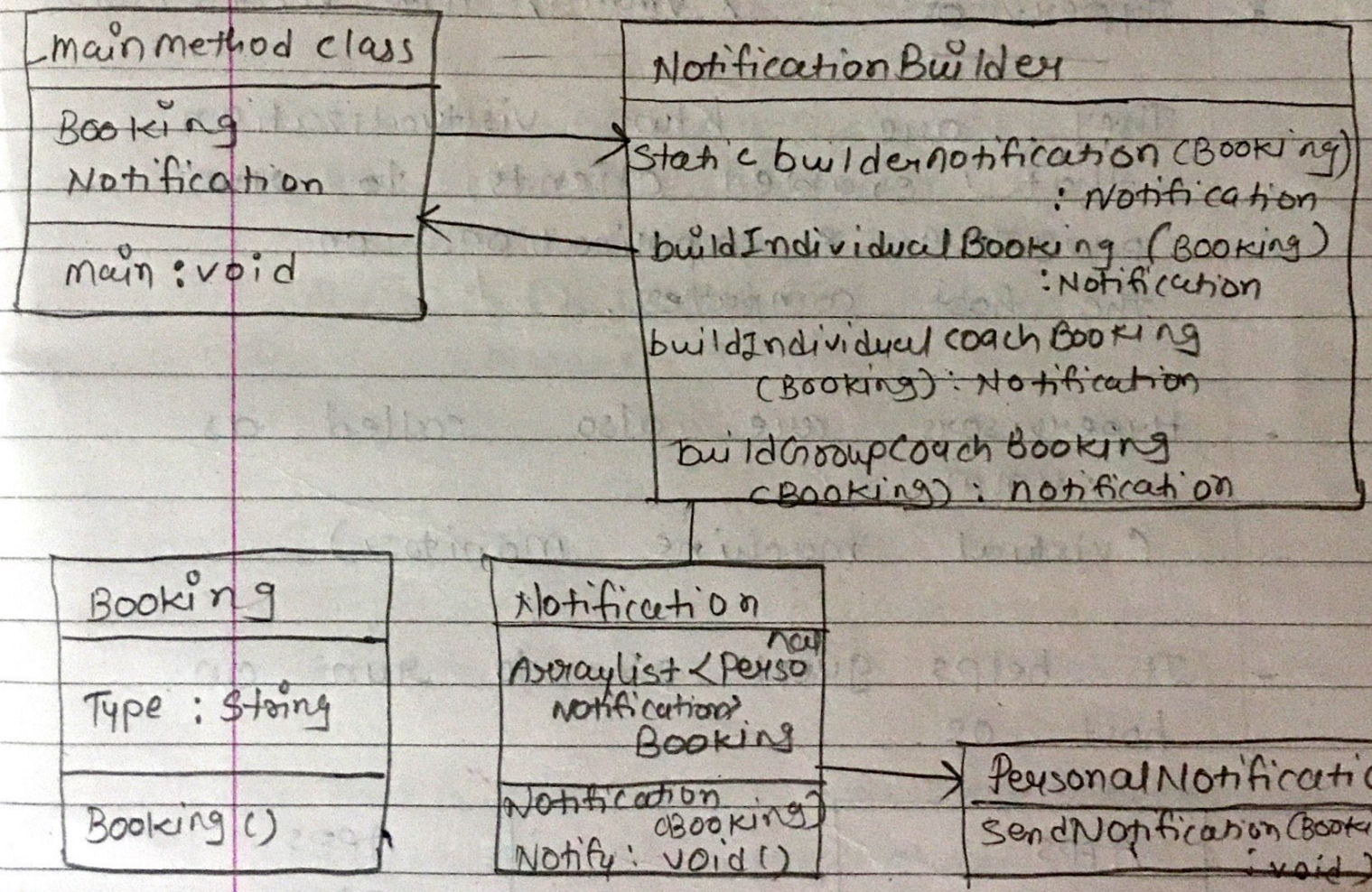
* Abstract Factory :-

- Factory of factory.
- want to create objects of related family.
- create libraries of language.

→ Factory Method.

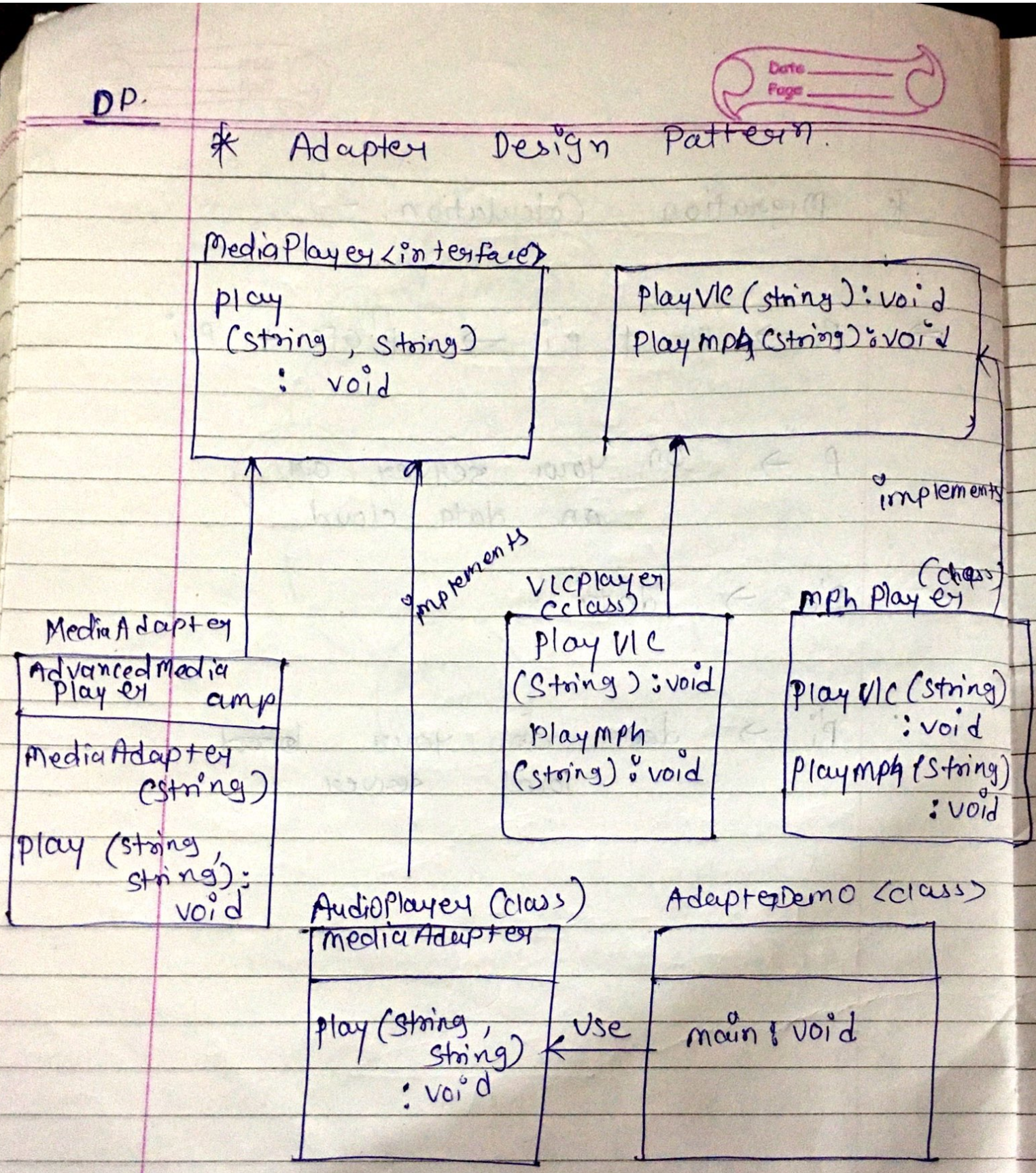


25/7. DP.



DP.

* Adapter Design Pattern.



18 DP.

* Media Player. (Adapter Pattern)

```
interface mediaPlayer
{
    void play(String audioType,
              String fileName);
}
interface AdvancedMediaPlayer
{
    void playVlc(String fileName);
    void playMp4(String fileName);
}
class VlcPlayer implements
                AdvancedMediaPlayer
{
    public void playVlc(String fileName)
    {
        System.out.println("media file
        Name for vlc is" + fileName);
    }
    public void playMP4(String
                        fileName)
    { }
}
```

```
class Mp4Player implements
                AdvancedMediaPlayer
{
    public void playVlc(String
                        fileName)
    { }

    public void playMp4(String fileName)
    {
        System.out.println("media
        file Name for mp4" + fileName);
    }

ass MediaAdapter implements
                MediaPlayer

    AdvancedMediaPlayer amp;
    MediaAdapter(String audioType)
    {
        if(audioType.equalsIgnore
                Case("vlc"))
            amp = new VlcPlayer();
```

```
{
    if (audioType.equalsIgnoreCase
            ("mp3"))
    System.out.println("playing
        mp3 file Name" + fileName);

    else if (audioType.equalsIgnoreCase
            ("vlc") || audioType.
        equalsIgnoreCase("mpu"))
    {
        mediaAdapter = new
            MediaAdapter(audioType);
        mediaAdapter.play(audioType,
            fileName);
    }
    else
        System.out.println("Invalid
            media player");
    }
}

class ExOfAdapter
```

{

public static void main
(String args[])

{

// mediaAdapter ma = new
MediaAdapter (args[0]);

AudioPlayer ap = new AudioPlayer();
ap.play("mp3", "beyond the
horizon.mp3");
ap.play("mp4", "alone.mp4");
ap.play("vlc", "far far away.
vlc")
ap.play("avi", "mind me.avi");

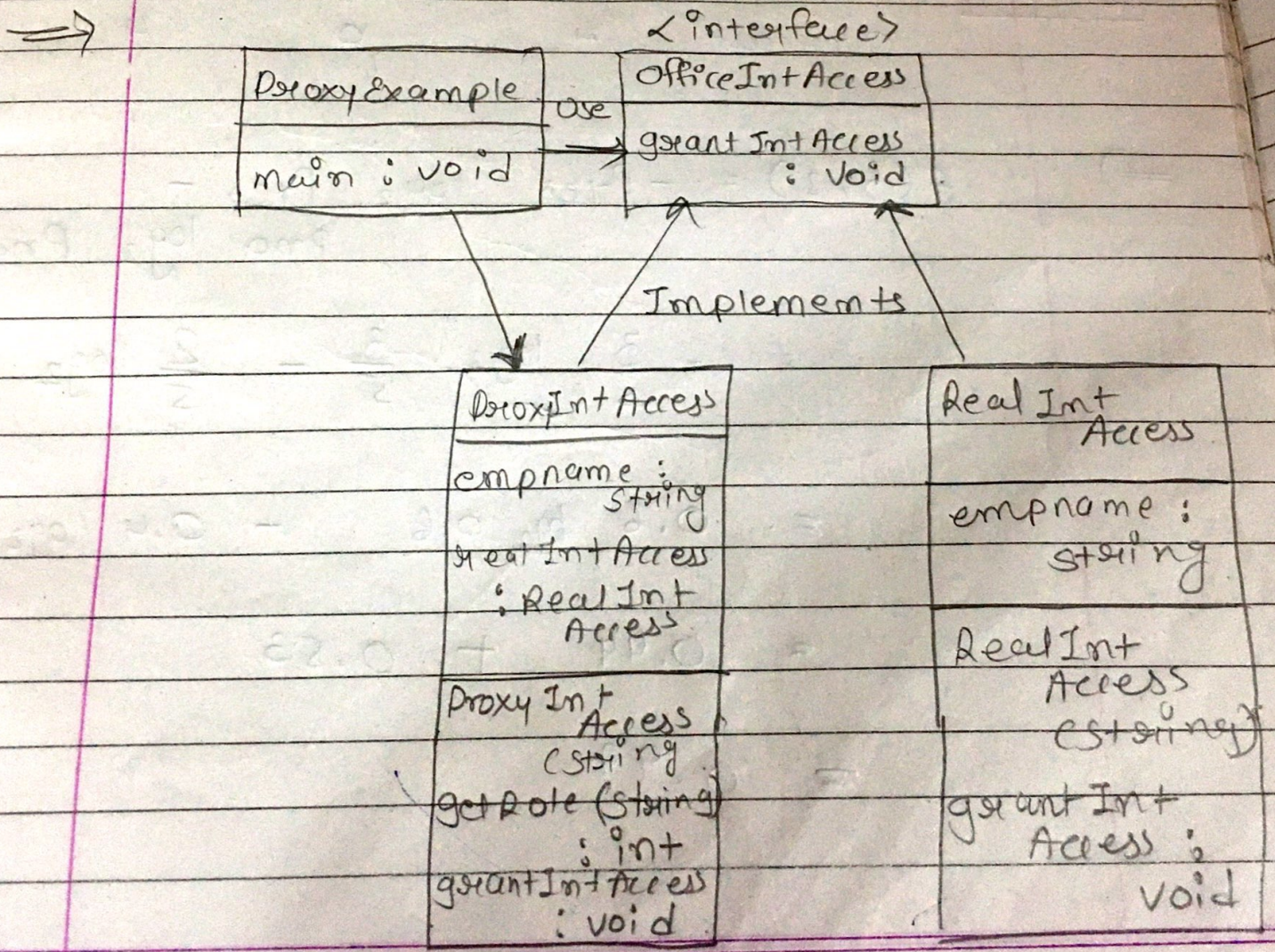
}

}

DP.

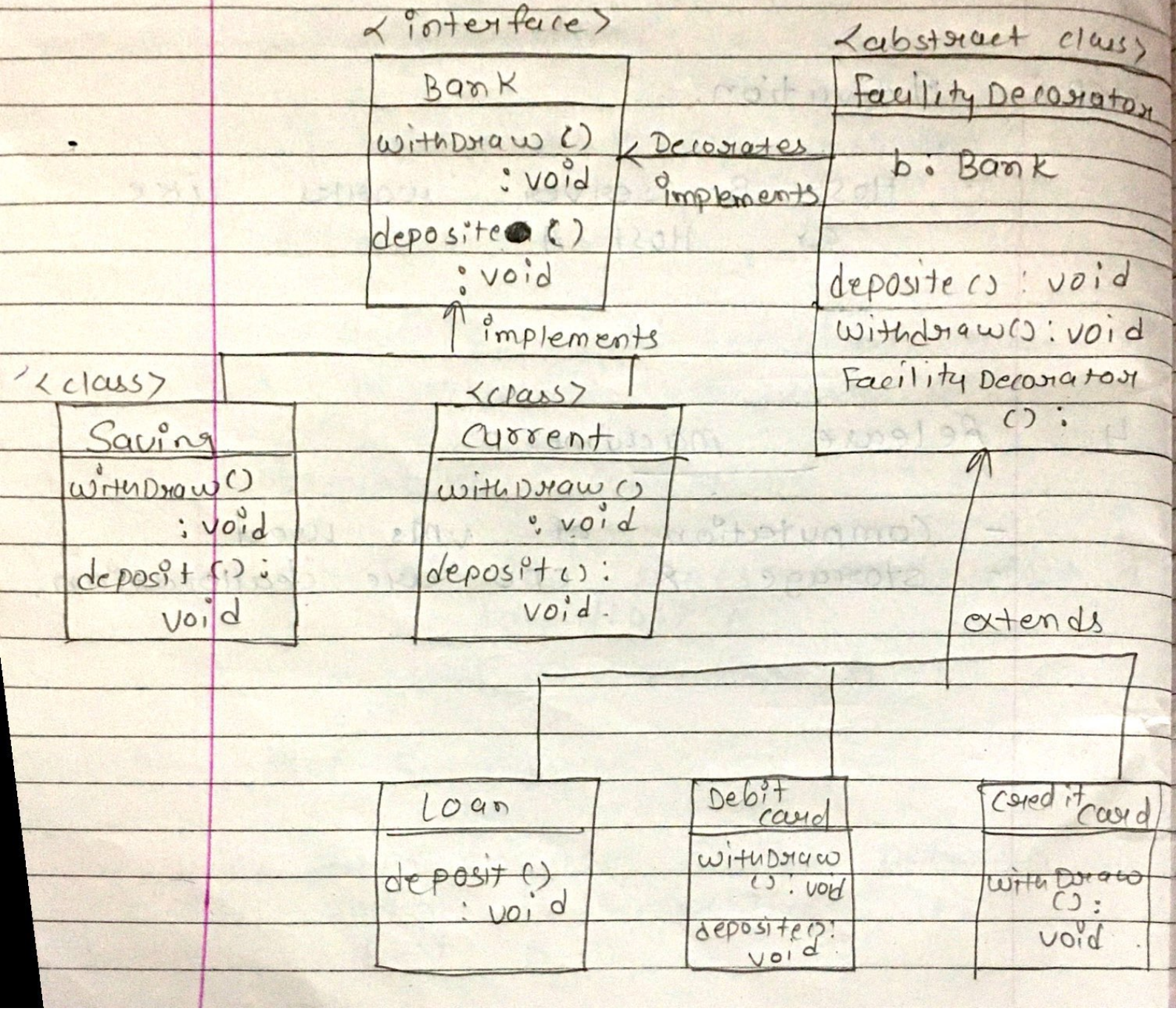
1. Proxy Design Pattern :-

- Remote
- Virtual
- Protection
- Smart.



21/8 DP.

* Decorated design pattern :-



23/8 DP

Facade Design Pattern

⇒ It is used when we have a complex system that we want to expose to clients in a simplified way to hide internal structure.