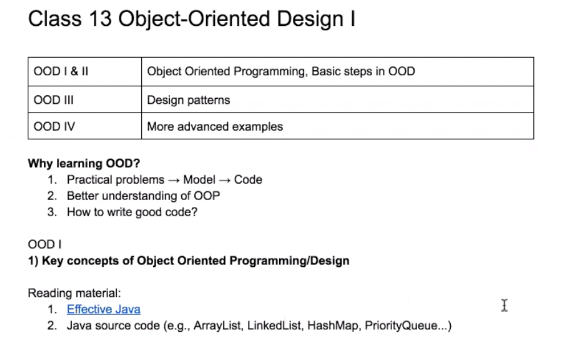
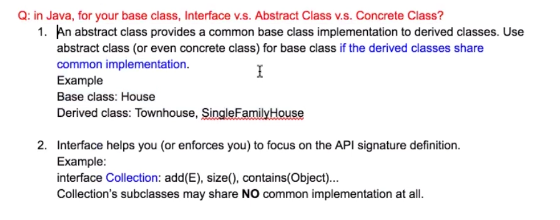
OOD Class 1

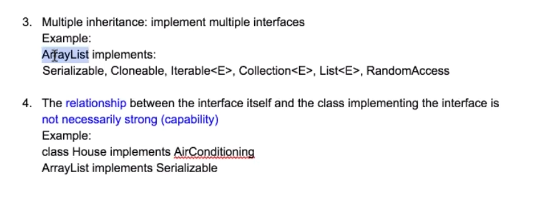


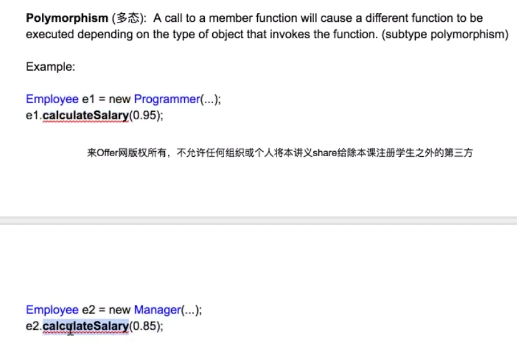


interface - act as

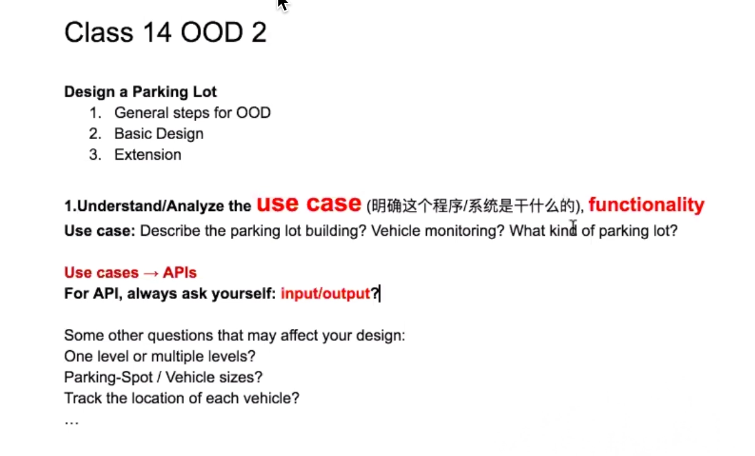
abstract - is a

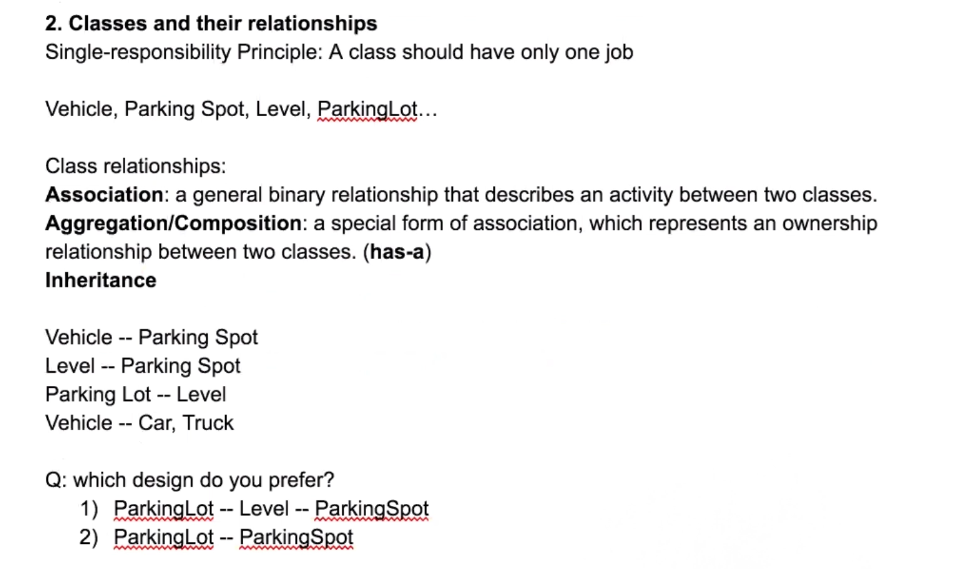


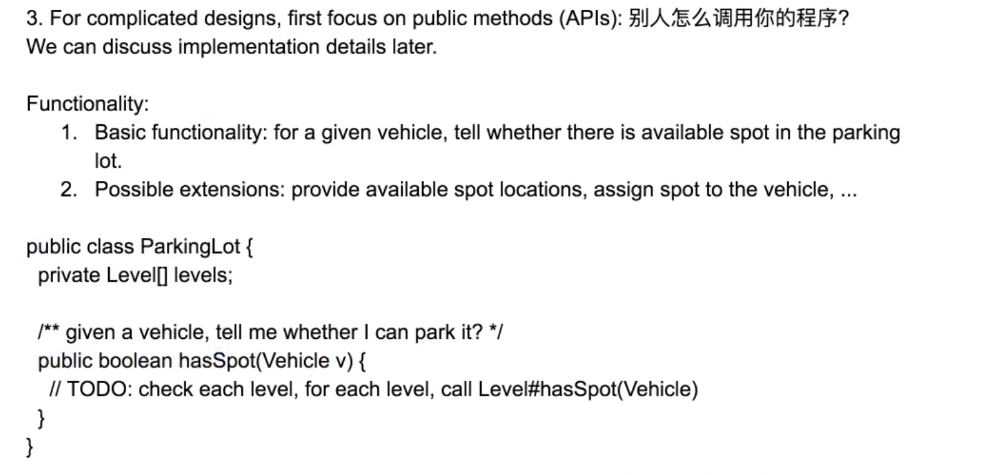


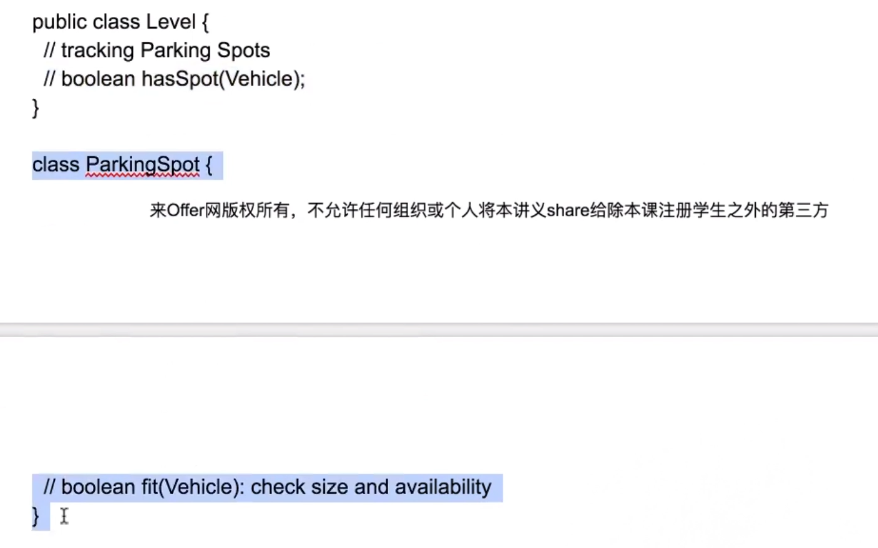


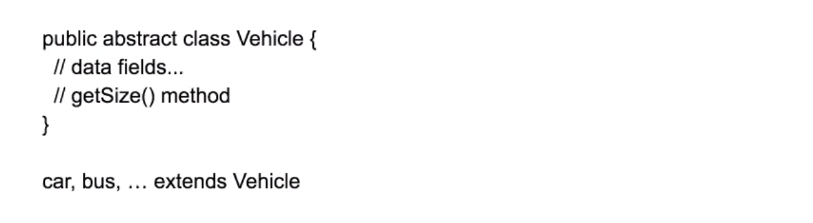
OOD Class 2

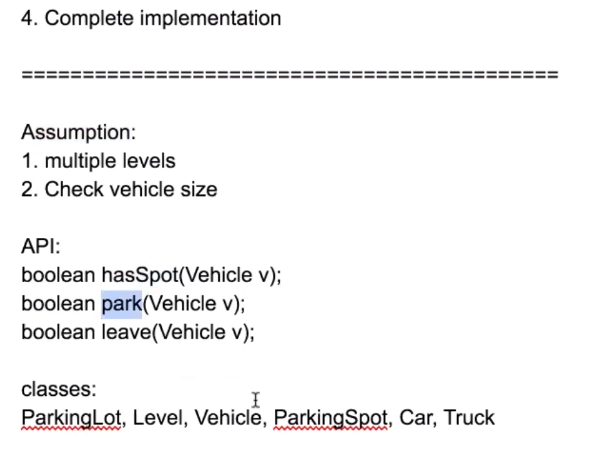


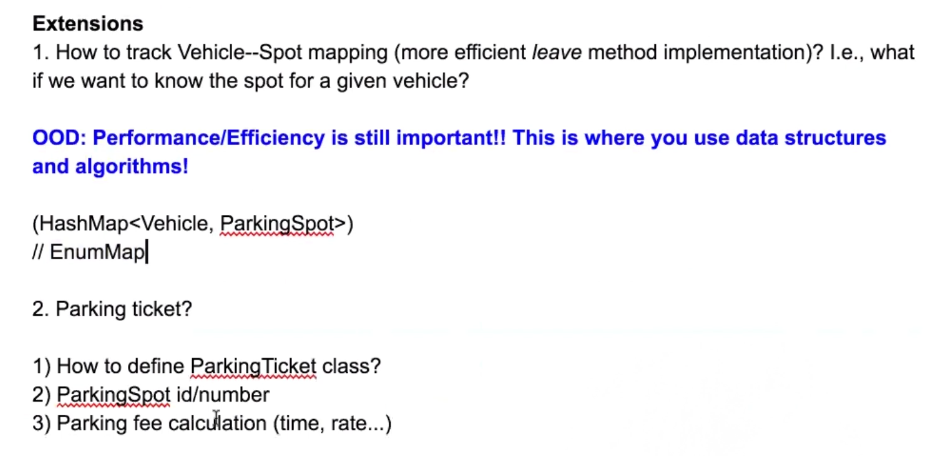




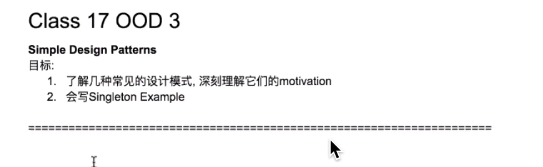








OOD Class 3



今天老师讲OOD 3 主要是Design Pattern

今天主要是自己写一个Singleton Example

如果有k个fields作为构造器的参数, 那么一共有2^k种组合

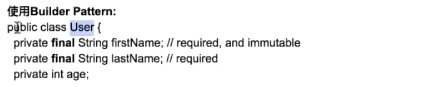
那么你就必须构建很多个constructor 所以并不好

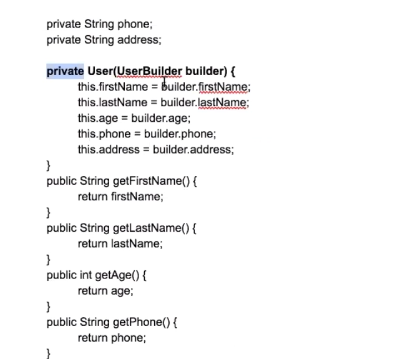
也可以设定一个简单的constructor 但是设定很多set函数 慢慢去设定

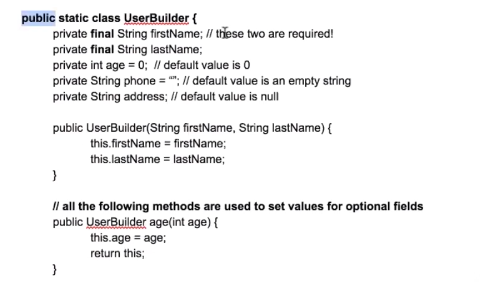
但是这个是有问题的 违反了encapsulation的原则 相当于一泡屎一次性不好好拉完

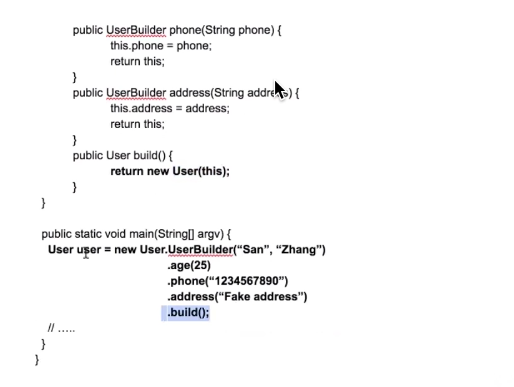
这里的nested class必须要设置成static的 原因如下 nested class是用于构建外部的class的 如果不是static class 那么内部的class还需要new 一个外部的class 循环依赖

================Builder Pattern=================









=================Factory Pattern=================

## Factory Pattern

+ create an interface

```

public interface Shape{

void draw();

}

```

+ create concrete classes

```

public class Rectangle implements Shape{

@Override

public void draw(){

System.out.println("Inside Rectangle");

}

}

```

```

public class Circle implements Shape{

@Override

public void draw(){

System.out.println("Inside Circle");

}

}

```

+ create factory to generate concrete classes

```

public class ShapeFactory{

public Shape getShape(String type){

if(type == null){

return null;

}

if(shape.equalsIgnoreCase("Rectangle")){

return new Rectangle();

}

if(shape.equalsIgnoreCase("Circle")){

return new Circle();

}

return null

}

}

```

+ demo

```

public class Demo{

ShapeFactory factory = new ShapeFactory();

Shape circle = factory.getShape("Circle");

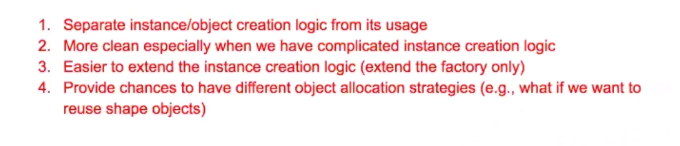
circle.draw();

Shape rectangle = factory.getShape("Rectangle");

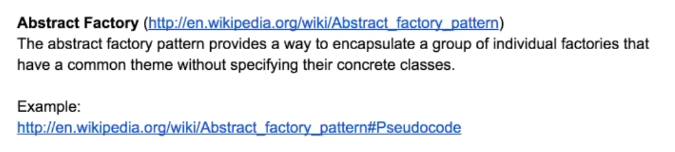
rectangle.draw();

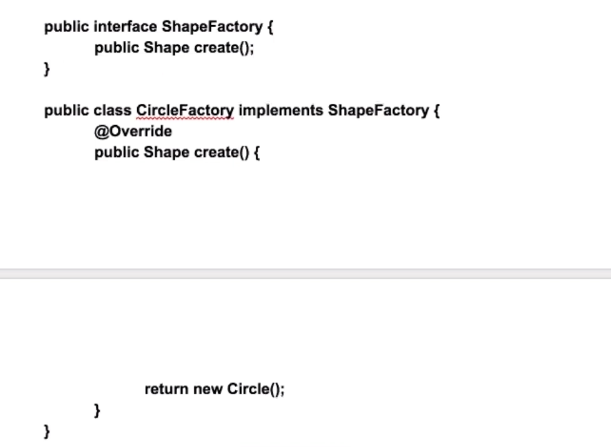
}

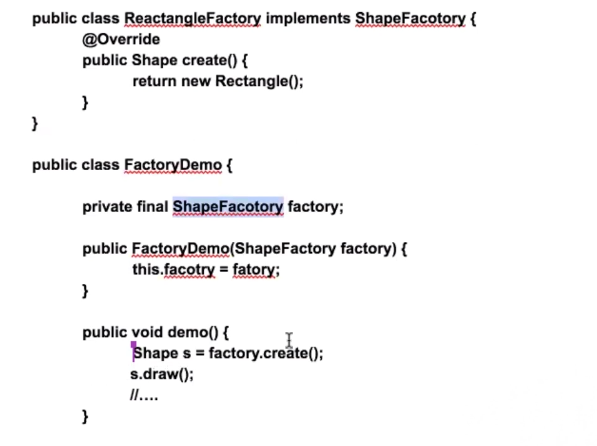
```



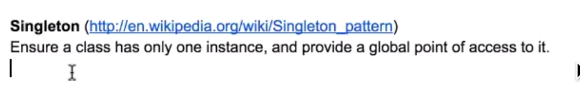
===============abstract factory pattern============







===================Singleton====================



+ create singleton class

```

public class SingleObject{

private static SingleObject ins = new SingleObject(); // 在加载类的时候就创建了

private SingleObject(){}

public static SingleObject getInstance(){

return ins;

}

public void show(){

System.out.println("Hello");

}

}

```

+ demo

```

public class Demo{

public static void main(String[] args){

SingleObject obj = SingleObject.getInstance();

obj.show();

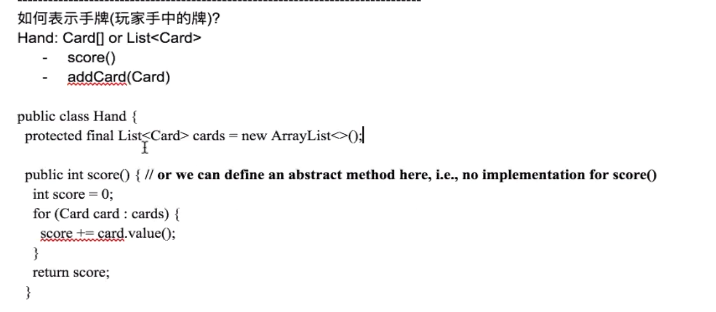
}

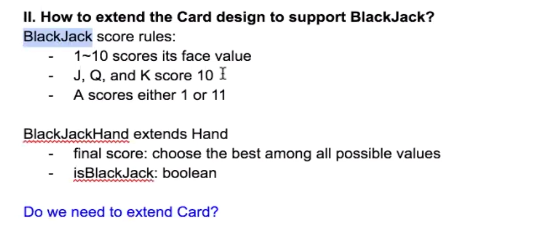
}

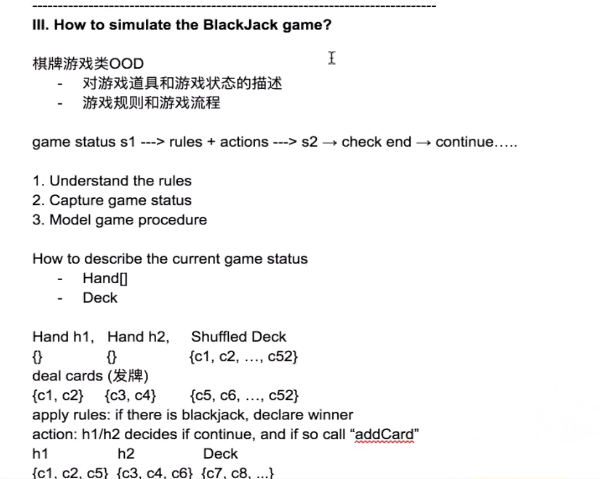
```

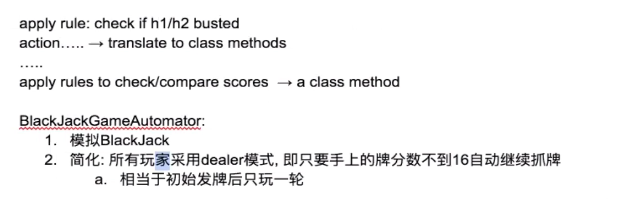
OOD Class 4

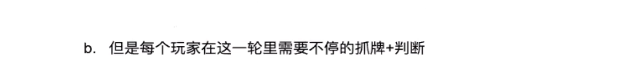


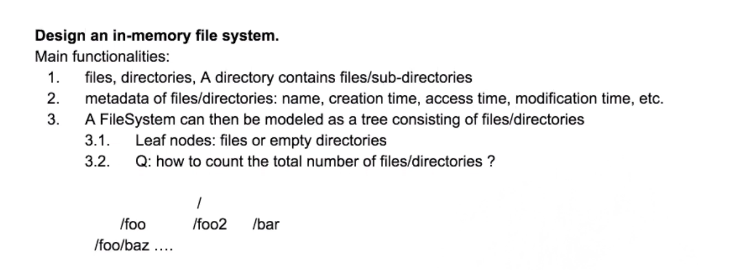




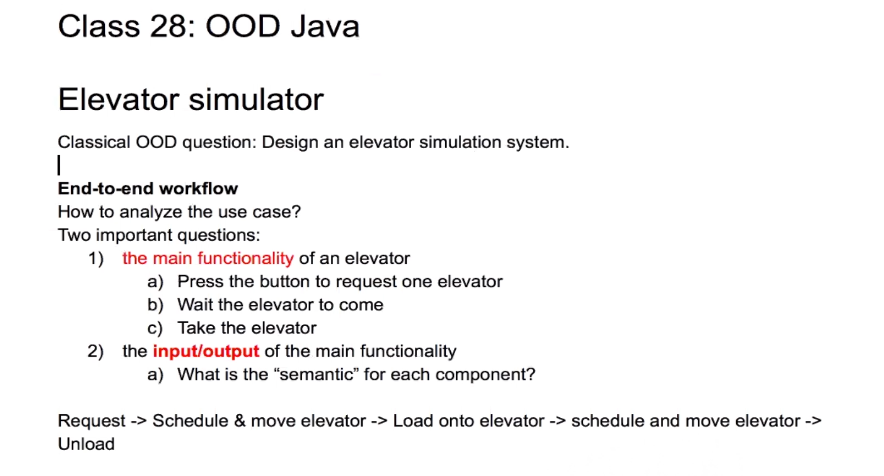


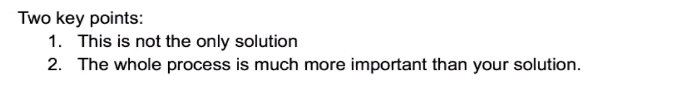






OOD Class 5 – Java Prac





1. Work Flows!!!

(2) Model “Things” in your program!!!

