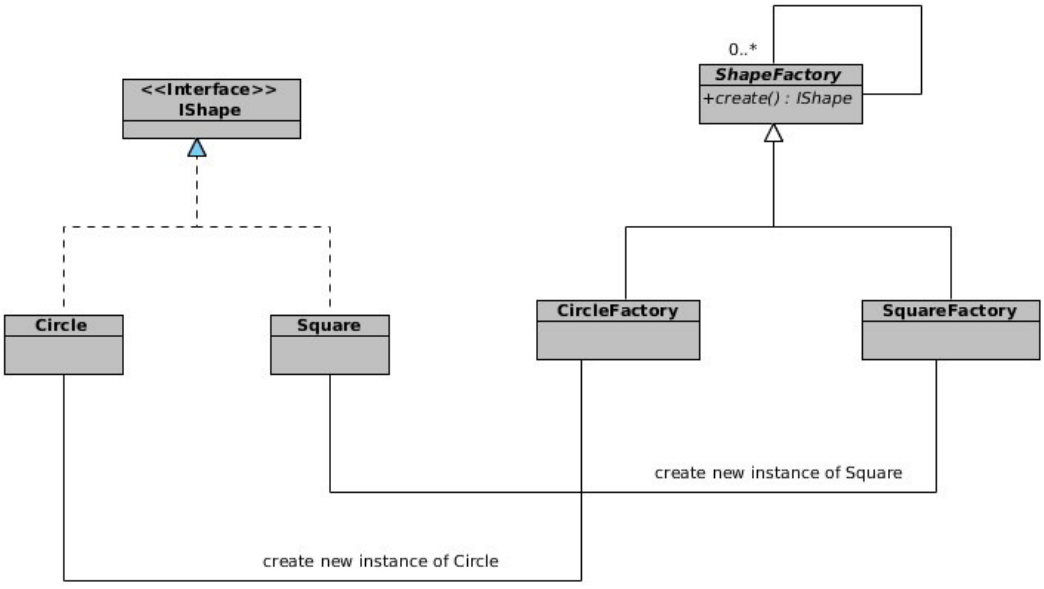
Software design patterns: 2010 – 2011

(exercise 2 – Method Factory, Prototype, Singelton)

Задача 1. В в папка singelton има два класа : **Country** и **CountrySIngeltonTest.** В тестовия клас можем веднага да забележим, че създаваме два пъти инстанция на класа **Country**, представляваща България. Т.к. няма две държави България, нека с помощта на Singelton шаблона да създадем нов статичен метод, с параметър от тип String, в класа **Country**, който ако няма вече създадена инстанция на желаната страна**,** да създаде такава и да я запази в подходяща структура (вие изберете каква). В противен случай: да ни върне вече създадената инстанция. Също така нека ограничим и броя създадени инстанции на класа до **195** – броят на страните в световен мащаб.

Задача 2. В папка factoryMethodPolymorphic е създадена система, имплементираща шаблона за проектиране Factory Method за създаване на фигури: **Circle** & **Square**. За всяка фигура има конкретна фабрика, която я създава.
Имайки предвид сорс кода и отразяващата го UML графика, добавете бизнес логиката, необходима за създаване и на Triangle (**Triangle** & **TriangleFactory** класове)



Задача 3. В папка pizzaCompany имаме полузавършена пицария. Ако разгледаме предадения сорс код и UML диаграмата, ще забележим, че отново с помощта на Builder шаблона приготвяме различни видове пици. Това, което му лиспва, е **CookDirector**, който използва конкретна инстанция на super class на **PizzaBuilderAbstract** и с негова помощ 'строи' поръчаната пица. От нас се изисква да довършим по този начин системата (пицарията), че в него да използваме:

→ Singelton
→ Method Factory
→ Prototype
→ Builder : или по точно неговото довършване, т.к. ни лиспва **CookDirector**.

За тази цел :

3.1. Извършете следните стъпки, използвайки сегашния сорс код

3.1.1. Създайте клас **CookDirecotor**

3.1.2. Намправете го Singelton (не ни е нужен повече от един директор, използващ строителите за направата на пици в системата)

3.1.3. Добавете Factory метод:

public PizzaBuilderAbstract createPizzaBuilderFactory(PizzaBuilderTypes builderType)

- връщащ ни инстанция на наследяващ **PizzaBuilderAbstract** клас в зависимост от подадения тип enumeration.

3.1.4. Използвайки java.util.Clonnable интерфейса, направете Pizza prototype : т.е. да връща копие от себе си

3.1.5. Тествайте така конструираната система, следвайки стъпките в **FMBuilderSingeltonPrototypeTest** класа

3.2. (Дискусия) Помислете за подобряване на сегашната структура companyPizza. От архитектурна гледна точка, дали ако извършим следния refactoring, постигаме по-силна кохезия (по-висока свързаност на отговорностите):

→ добавим всичките съществуващи строители на пици (за момента **HawaiianPizzaBuilder** и **SpicyPizzaBuilder**) като static final property в класа **CookDirector**

→ имплементираме нов метод: **Pizza createPizza**(**PizzaBuilderTypes builderType**) , който според типа използва конккретния Pizza Builder (един от двата, създаден като static final property) и връща **копие** на създадената от него пица (т.к. в самото стартиране на програмата се инициалират строителите и се създава и инстанция на съответната пица и не искаме да даваме една и съща пица на клиентие, задаваме копие от нея като вече сме имплементирали Prototype шаблона)