# Week 5 Exercise 1 – Creational Design Patterns

Name (individual):

**Objectives:**

* Understand and further apply how each of the creational pattern works.

All 3 exercises are adapted from the following sources

- Java Design Patterns: A Hands-On Experience with Real-World Examples

- Design Patterns reference by SourceMaking

For reviewing of the **points for class participations**, please develop the codes for the questions in this document. Place this document in your **individual exercise GitHub repo <e.g., exercises-2020-21t1-xxxxx> -> submission -> week 5 -> class-exercises -> CS301\_Week5\_Ex1\_<your name>.docx**

Please follow the skeleton codes so that the maven tests can be executed. You can review the test codes in **designpatterns\src\test\java** folder. Try to do it yourself and refrain from finding answers online. The coverage of these test cases are not 100% and just meant to validate your answers to a certain extent. If you have better test cases, feel free to suggest by raising an issue within your exercises repo.

When you use apache maven to test, there is a POM (Project Object Model) file that is being used. An introduction is available at

* <https://maven.apache.org/guides/introduction/introduction-to-the-pom.html>
* <https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html>
* <https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>

For the following exercises,

* Our POM file is pom.xml located in the root directory of designpatterns
* All our java source codes are in **src\main\java directory**
* All our test codes are in **src\test\java directory**
* Using JUnit for running the tests.

## Question 1 - Singleton

Refer to the **Captain.java** given. This piece of code return unique Captain instances and not a singleton.

1. Navigate to the **designpatterns\src\main\java** folder
2. Compile with **javac itsa\patterns\Captain.java**
3. Run **java itsa.patterns.Captain.** The execution will finished successfully but results is not correct. See the output.

Rewrite the code so that there is only one instance of the Captain object. You must following the following conventions as shown in the skeleton codes.

* The captain class is **Captain** with **method getCaptain**
* The test code will test initialize twice the Captain class and check if **Captain1 and Captain2 are referring to the same instance.**

You can also invoke a simple maven test script when ready

1. Navigate to the **designpatterns** folder
2. Run **apache-maven-3.6.1\bin\mvn clean**
3. Run **apache-maven-3.6.1\bin\mvn test -Dtest=CaptainTest**
4. Review the output.

## Question 2– Builder

Refer to the example given for **BuildPizza.java**

1. Navigate to the **designpatterns\src\main\java** folder
2. Compile with **javac itsa\patterns\BuildPizza.java**
3. Run **java itsa.patterns.BuildPizza**. The execution will finished successfully but the output is empty (e.g. “hawaiianPizza is”. You should get the pizza in a string e.g. **“crossmildpineapple” and “bakedhotpepperoni”.** This output is built by adding dough, sauce and topping.

Refactor the code based on the Builder pattern.

You must following the following conventions as shown in the skeleton codes.

* Use the given class **Pizza (product)**, **PizzaBuilder (abstract builder)**, **HawaiianPizzaBuilder** (concrete builder), **SpicyPizzaBuilder** (concrete builder) and **Waiter (director)** classes as skeleton codes to modify.
* The test code will test outputs of **“crossmildpineapple” for HawaiianPizzaBuilder** and **“bakedhotpepperoni” for SpicyPizzaBuilder.**

You can also invoke a simple maven test script when ready

1. Navigate to the designpatterns folder
2. Run **apache-maven-3.6.1\bin\mvn clean**
3. Run **apache-maven-3.6.1\bin\mvn test -Dtest=BuildPizzaTest**
4. Review the output.

## Question 3 – Simple Factory Method

Refer to the example given for **ImageDecoder.java.**

This piece of code is an example of an image decoder for gif images. However, the developer understands that there is a need of multiple image formats.

1. Navigate to the **designpatterns\src\main\java** folder
2. Compile with **javac itsa\patterns\ImageDecoder.java**
3. Run **java itsa.patterns.ImageDecoder sample.gif**. The execution will finished successfully.
4. However if you run **java itsa.patterns.ImageDecoder sample.jpg**, the execution will encounter an exception.

Extend the code to support at least two formats (e.g. sample.gif and sample.jpg) based on the **simple factory design pattern**. You must following the following conventions as shown in the skeleton codes.

* The factory class is **ImageDecoderFactory** with **method createImageReader**
* The interface class is **ImageReader** with **method** **getDecodeImage**
* The test code will test for **both “sample.gif” and “sample.jpg” inputs.**

You can also invoke a simple maven test script when ready

1. Navigate to the **designpatterns** folder
2. Run **apache-maven-3.6.1\bin\mvn clean**
3. Run **apache-maven-3.6.1\bin\mvn test -Dtest=ImageDecoderTest**
4. Review the output.