The Last Quiz (50 minutes exam)

Name: 곽영혜 Student ID: 21700034 Phone #: 010-6604-8628

Please make your answer with key words briefly. If you use unnecessarily lengthy words, you will get serious penalty. (답안은 keyword 를 이용하여 간결하게 쓰시길 바랍니다. 불필요하게 길게 쓰면, 감점 당합니다.)

Please do not write what you do not understand. (이해하지 못하는 내용은 쓰지 마세요.) 한국어가 모국어인 학생들은 우리말로 답안 작성하길 권합니다.

You are given 50 minutes. (50분 동안에 답안 작성하시길 바랍니다.)

1-1. What is the criteria to use specific design pattern among such similar patterns as builder, abstract factory, factory method, prototype, and façade patterns. (builder, abstract factory, factory method, prototype, façade pattern 과 같은 비슷한 pattern 들을 구분해서 사용하는 기준에 대하여 기술해 보세요.)

디자인 패턴은 사용 목적에 따라 'Creational(생성)', 'Structural(구조)', 'Behavioral(행위)' Pattern으로 나눌 수 있으며 범위에 따라 Class를 대상으로 하는가 객체를 대상으로 하는가로 나눌 수 있습니다.

builder, abstract factory, factory method, prototype pattern들은 Creational pattern에 속하며 객체 생성 매커니즘을 담당합니다.

Builder Pattern은 표현에서의 복잡한 객체의 구성하여 같은 구성 프로세스가 다른 표현을 생성할 수 있게 합니다.

Abstract factory Pattern은 연관되어 있거나 독립적인 객체를 생성하는 interface를 객체의 구체적인 class지정 없이 제공하는 pattern입니다.

Factory method Pattern은 class가 subclass에 대한 인스턴스화를 지연시킬 수 있습니다.

Prototype Pattern은 Prototype instance를 사용하여 만들 개체의 종류를 지정하고 이를 복제하여 새 개체를 만드는 pattern입니다.

façade pattern은Structural Pattern에 속하며 일반적인 작업을 쉽게 사용할 수 있도록 기존 interface의 단순화된 interface를 생성합니다.

1-2. How do they can be used together if it is possible? (1-1번의 비슷한 pattern 들을 같이 사용할 수 있다면, 어떻게 사용하면 되는가?)

객체들의 관계를 tree 구조로 구성하여 부분-전체 계층을 표현하여 단일 객체와 복합 객체를 모두 동일하게 사용할 수 있도록 하면 된다.

1-3. Explain the differences between builder pattern and factory pattern.

builder pattern은 복잡한 객체를 단계별로 구성하는데 중점을 두고 단일 객체 생성에 연관이 있는 반면 factory pattern은 공통 interface를 공유하는 제품 객체들을 생성하는데 사용됩니다.

1-4 Explain the differences between facade pattern and builder pattern.

facade pattern은 객체 모델에서 호출을 단순화하는 방법이며 일련의 작업을 수행 할 때마다 많은 양의 코드를 작성할 필요가 없습니다. 작업을 수행하기 위해 여러 객체와 메소드를 호출해야하는 것을 단순화하기 위해 사용됩니다.

builder pattern은 class 또는 class 세트의 구성 논리를 단순화하는 데 사용되는 일종의 facade pattern입니다. 그 목적은 대상 객체를 구성하는 방법을 명확하게 하고 명확한 느낌을 주는 명확한 방법을 제공하는 것입니다.

2. Answer each question for the given program code.

(1) What is the role of SyncOb object?

(2) When is the block below of the given source code working? Explain its role. catch(InterruptedException exc) {

System.out.println("Interrupted.");

}

(3) Explain the main program logic of this program.

class SyncOb {

boolean ready = false;

synchronized void waitFor() {

String thrdName = Thread.currentThread().getName(); System.out.println(thrdName + " is waiting. \n");

try {

while(!ready) wait();

}

catch(InterruptedException exc) {

System.out.println("Interrupted.");

}

System.out.println(thrdName + " starts to proceed. \n");

}

synchronized void goAhead() {

String thrdName = Thread.currentThread().getName(); System.out.println("\n" + thrdName + " thread lets Working Thread resume execution.\n");

ready = true; notify();

}

}

class MyThread implements Runnable {

SyncOb syncOb; MyThread(String name, SyncOb so) {

syncOb = so; new Thread(this, name).start();

}

public void run() {

syncOb.waitFor();

}

}

class ThreadComDemoExam {

public static void main(String args[]) {

try {

SyncOb sObj = new SyncOb();

new MyThread("Working Thread", sObj);

for(int i=0; i < 10; i++) {

Thread.sleep(250); System.out.print(".");

}

System.out.println();

sObj.goAhead();

}

catch(InterruptedException exc) {

System.out.println("Main thread interrupted.");

}

}

}