Answer your solutions in your favorite language, Korean or English. But you should use English for all the technical terms. If your writing is unreadable, your solution will be considered wrong. Please do not give me questions. You are on your own interpreting and answering exam questions. (한국어나 영어 중 편한 언어로 답하시면 되나, 모든 전문 용어는 영어로 써야 합니다. 문장이나 글씨가 읽을 수 없는 수준이면 틀린 것으로 간주됩니다. 주어진 문제들에 관하여 스스로 해석하고 답해 주시길 바랍니다. 질문 받지 않습니다.)

* 1. Compare and explain the difference between prototype pattern and flyweight pattern. You may use code snippets as necessary.

prototype pattern은 새로운 instance를 생성하는데 비용이 많이 드는 점을 해결하기 위해 기존의 instance를 copy하여 수정하여 사용하는 반면, flyweight pattern은 기존의 instance를 공유할 수 있어 재사용성이 높습니다.

<prototype pattern> < flyweight pattern>

병, 화면, 모니터, 음식이(가) 표시된 사진

자동 생성된 설명 스크린샷, 앉아있는, 테이블, 전화이(가) 표시된 사진

자동 생성된 설명

스크린샷, 앉아있는, 테이블, 전화이(가) 표시된 사진

자동 생성된 설명

* 1. Briefly explain the role of the “register” and “create” in the code below.

public class Manager {

private HashMap showcase = new HashMap();

public void register(String name, Product proto) {

showcase.put(name, proto);

}

public Product create(String protoname) {

Product p = (Product)showcase.get(protoname);

return p.createClone();

}

}

Register function은 각 frame object(Product proto)를 showcase(HashMap)에 저장하여 instance를 생성하는 역할을 수행합니다.

Create function은 register를 통해 생성된 instance를 copy하여 새로운 Product로 생성하는 역할을 수행합니다.

* 1. Explain in detail the role of “pool” and “getBigChar” of the following code.

import java.util.HashMap;

public class BigCharFactory {

private HashMap pool = new HashMap();

private static BigCharFactory singleton = new BigCharFactory();

private BigCharFactory() { }

public static BigCharFactory getInstance() { return singleton; }

public synchronized BigChar getBigChar(char charname) {

BigChar bc = (BigChar)pool.get("" + charname);

if (bc == null)

{

bc = new BigChar(charname);

pool.put("" + charname, bc);

}

return bc;

}

}

Pool은 getBigChar의 HashMap으로 BicChar의 Instance들을 저장하는데 사용됩니다.

getBigChar은 bc에 pool안의 charname을 가져오도록 설정하여 만약 bc가 null일 경우 새로운 BigChar를 pool에 넣도록 하고 return하여 주며, 아닐 경우에는 pool에서 불러온 BigChar를 return해주는 function입니다.

2-1. Compare the visitor pattern and the chain of responsibility pattern based on the following criteria. You may use short code snippets for your explanation.

• the relationship and logic between a server object and a client object

• maintainability of the code

visitor pattern은 데이터 구조와 연산을 분리하였기에 구조와 요소를 변경하지 않고 새로운 연산을 추가할 수 있습니다. 즉, 새로운 visitor(server)를 추가하면 됩니다. 이러한 점으로 인해, maintainability of the code가 chain of responsibility pattern에 비해 높습니다.

병, 검은색, 하얀색, 빨간색이(가) 표시된 사진

자동 생성된 설명

반면, chain of responsibility pattern은 여러 개의 server들이 특정 순서로 연쇄적으로 연결되어 있으며 client에서 request가 들어오게 되면, 요청을 수행할 수 있는 server인지 확인후, 다음 연결 server로 요청을 전달하며 이 과정을 수행할 수 있는 server에 도달하기까지 계속합니다.

컴퓨터, 노트북, 검은색, 앉아있는이(가) 표시된 사진

자동 생성된 설명

2-2. Explain in detail the role of “this” in the code below:

class This implements Element {

public void accept( Visitor v )

{

v.visit( this );

}

}

this는 v.visit function이 여러 개의 server에서 공유되는 같은 type parameter인 Visitor interface에서 polymorphic function으로 사용되기 위해 사용되는 parameter 변수입니다. This, That, TheOther과 같은 각 서버를 나타냅니다.

3-1. Explain some common facts of the observer pattern and the MVC pattern.

observer pattern과 MVC pattern 둘다 Behavioral Design Pattern에 속하며 MVC의 Model과 Observer의 Observable, MVC의 View와 Observer으 Observer이 유사한 functionality를 가지고 있습니다. 두 패턴 다 1:다수 관계를 형성할 수 있으며 이는 같은 data로 Multiple format의 output들을 생성할 수 있게 됩니다. Observer pattern은 MVC pattern에서 사용되기에 두 패턴 모두 Model과 View사이, Observable과 Observer 사이에 loose한 coupling관계가 생성되어 있습니다.

3-2. Explain the logic of the underlined statement of the following example code snippet.

public class HexaObserver extends Observer

{

public HexaObserver(Subject subject)

{

this.subject = subject;

this.subject.attach(this);

…

HexaObserver와 같은 각자의 observer들을 Observer list에 더하여 input에 따른 received response들을 출력하기 위해 사용됨

그리기이(가) 표시된 사진

자동 생성된 설명