AbstractFactoryPattern(객체 생성의 가상화)

* 관련 있는 객체의 생성을 가상화 할 수 있다.
* 생성 부분의 가상화 /관련있는 객체

예시 1

package designpattern;

public class AbstractFactoryPattern {

public static void main(String[] args) {

BikeFactory factory = new Samfactory();

Body body = factory.craeteBody();

Wheel wheel = factory.createWheel();

System.out.println(body.getClass());

System.out.println(wheel.getClass());

BikeFactory factory1 =new GtBikeFactory();

Body body1 = factory1.craeteBody();

Wheel wheel1 = factory1.createWheel();

System.out.println(body1.getClass());

System.out.println(wheel1.getClass());

}

}

interface Body{

}

interface Wheel{

}

interface BikeFactory{

public Body craeteBody();

public Wheel createWheel();

}

class Samfactory implements BikeFactory{

@Override

public Body craeteBody() {

// TODO Auto-generated method stub

return new SamBody();

}

@Override

public Wheel createWheel() {

// TODO Auto-generated method stub

return new SamWheel();

}

}

class SamBody implements Body{

}

class SamWheel implements Wheel{

}

class GtBody implements Body{

}

class GtWhell implements Wheel{

}

class GtBikeFactory implements BikeFactory{

@Override

public Body craeteBody() {

// TODO Auto-generated method stub

return new GtBody();

}

@Override

public Wheel createWheel() {

// TODO Auto-generated method stub

return new GtWhell();

}

}

예시 2

-객체를 하나만 선언해도 여러개의 (함수에서 구현하여 아래 예시의경우 os 검사후 해당 os 클래스 리턴해주는방식) 클래스중 필요한 클래스를 자동으로 가져올 수 있음

package designpattern;

public class AbstractFactoryPattern\_2 {

public static void main(String[] args) {

GuiFac fac = new LinuxGuiFac();

TextArea text = fac.createTextArea();

Button button = fac.craeteButton();

System.out.println(text.getText() + button.getButton());

GuiFac win = new WindowFac();

TextArea text1 = win.createTextArea();

Button button1 = win.craeteButton();

System.out.println(text1.getText() + button1.getButton());

//맥도 같은 패턴으로 가능

// 사용자는 따로 소환안해도 되도록 만들기

GuiFac fac2 = FactoryInstence.getGuiFac();

}

}

interface GuiFac{

public Button craeteButton();

public TextArea createTextArea();

}

interface TextArea{

public String getText();

}

interface Button{

public String getButton();

}

// 패키지 Window

class WindowFac implements GuiFac{

@Override

public Button craeteButton() {

// TODO Auto-generated method stub

return new WindowButton();

}

@Override

public TextArea createTextArea() {

// TODO Auto-generated method stub

return new WindowTextAarea();

}

}

class WindowButton implements Button{

@Override

public String getButton() {

// TODO Auto-generated method stub

return "윈도우 버튼";

}

}

class WindowTextAarea implements TextArea{

@Override

public String getText() {

// TODO Auto-generated method stub

return "윈도우 텍스트";

}

}

// 패키지 Linux

class LinuxGuiFac implements GuiFac{

@Override

public Button craeteButton() {

//리눅스 소스

return new LinuxButton();

}

@Override

public TextArea createTextArea() {

// TODO Auto-generated method stub

return new LinuxTextArea();

}

}

class LinuxButton implements Button{

@Override

public String getButton() {

// TODO Auto-generated method stub

return "리눅스 버튼";

}

}

class LinuxTextArea implements TextArea{

@Override

public String getText() {

// TODO Auto-generated method stub

return "리눅스 text";

}

}

// 패키지 Mac .... 두개와 같은 방식으로 사용간으

// 콘크리트 패키지 외부에 유출되면안됨

class FactoryInstence {

public static GuiFac getGuiFac(){

switch(0){

case 0 : return new WindowFac();

case 1 : return new LinuxGuiFac();

//case 2 : return Mac factory ...

}//구현된 내용을 알 수 없음

return null;

}

}