PrototypePattern : 복잡한 인스턴스를 복사

* 생산 비용이 높은 인스턴스를 사용
* 종류가 너무 많아서 클래스로 정리되지 않는 경우
* 클래스로부터 인스턴스 생성이 어려운 경우

예제

package designpattern;

public class PrototypePattern {

public static void main(String[] args) throws CloneNotSupportedException {

// 축다요청 복사를하면 도형이 겹침으로 안겹치도록 살짝 이동 ㅛㅇ청

Circle circle1 = new Circle(1,1,3);

Circle circle2 = circle1.copy();

System.out.println(circle1.getX()+","+circle1.getY()+","+circle1.getR());

System.out.println(circle2.getX()+","+circle2.getY()+","+circle2.getR());

}

}

//java 에서는 명시적으로 써줘야

class Shape implements Cloneable{

private String id; // 모양을 id 를 통해서 만듬

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

}

// 꼭 extends 해줘야함

class Circle extends Shape{

private int x,y,r;

/\*\*

\* @param x

\* @param y

\* @param r

\*/

public Circle(int x, int y, int r) {

super();

this.x = x;

this.y = y;

this.r = r;

}

public Circle copy() throws CloneNotSupportedException{

Circle circle = (Circle)clone();

// 추가요청 사항 접수 circle 이 복잡한경우 복사가 힘들어서 clone 이 일아서해줌 한번씩만 더해주면됨

circle.x++;

circle.y++;

circle.r++;

return circle;

}

public int getR() {

return r;

}

public void setR(int r) {

this.r = r;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

}

깊은 복사와 얕은 복사 예시

package designpattern;

public class PrototypePattern2 {

public static void main(String[] args) throws CloneNotSupportedException {

// 깊은 복사와 얕은 복사

Cat na = new Cat();

na.setName("na"); // 나비설정

na.setAge(new Age(2012,3));

// 나비를 복사하여 새로운이름생성

Cat y = na;

y.setName("yo");

y.setAge(new Age(2013,2));

// 두개의 주소가 같아서 하나수정시 두개다 바뀜 ==> 앝은 복사

System.out.println(na.getName() + "," + na.getAge().getYear());

System.out.println(y.getName() + "," + y.getAge().getYear());

na = y.copy(); // 깊은 복사

na.setName("navi"); // 변견된것을 확인 가능

na.setAge(new Age(2012,3));

System.out.println(na.getName() + "," + na.getAge().getYear());

System.out.println(y.getName() + "," + y.getAge().getYear());

//깊은 복사가 잃어나지 않음 ==> 자동으로 새로운 주소값으로 참조함

y.setName("yo2");

y.getAge().setYear(2017);

y.getAge().setValue(7);

System.out.println(na.getName() + "," + na.getAge().getYear());

System.out.println(y.getName() + "," + y.getAge().getYear());

// 깊은복사를 위해서는 CAT 클래스에서 해야함

}

}

class Cat implements Cloneable {

private String name;

private Age age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Cat copy() throws CloneNotSupportedException {

Cat ret = (Cat) this.clone();

// main에서 마지막 예시의 깊은 복사를 위해 아래 추가하면 명시적 깊은복사 출력값달라짐

//ret.setAge(new Age(this.age.getYear(), this.age.getValue()));

return ret;

}

public Age getAge() {

return age;

}

public void setAge(Age age) {

this.age = age;

}

}

class Age {

private int year;

private int value;

public int getYear() {

return year;

}

public Age(int year, int value) {

super();

this.year = year;

this.value = value;

}

public void setYear(int year) {

this.year = year;

}

public int getValue() {

return value;

}

public void setValue(int value) {

this.value = value;

}

}