BuilderPattern : 복잡한 단계가 필요한 인스턴스 생성을 빌더 패턴을 통해서 구현할 수 있다.

* 서브 클래스에게 복잡한 다계를 거쳐야 샹성되는 개체의 구현을 넘겨주는 패턴
* 많은 인자를 가진 객체의 생성을 도움받아 생성

예제 2가지

**교과서 대로의 예제**

package designpattern;

public class BuilderPattern {

public static void main(String[] args) {

Computer computer = new Computer("i7","16G","256SSD");

System.out.println(computer.toString());

//인자 값이 많아질 경우 출력 힘들고 복잡해짐

// ==> 생성을 다른 객체로 넘김

ComputerFactory factory = new ComputerFactory();

factory.setPrint(new LgGramBlueprint());

factory.make();

// 설계 도를 만들어서 가지고옴

Computer computer4 = factory.getComputer();

System.out.println(computer.toString());

}

}

class Computer {

private String cpu;

private String ram;

private String storage;

public Computer(String cpu, String ram, String storage) {

super();

this.cpu = cpu;

this.ram = ram;

this.storage = storage;

}

public String getCpu() {

return cpu;

}

public void setCpu(String cpu) {

this.cpu = cpu;

}

public String getRam() {

return ram;

}

public void setRam(String ram) {

this.ram = ram;

}

public String getStorage() {

return storage;

}

public void setStorage(String storage) {

this.storage = storage;

}

@Override

public String toString() {

// TODO Auto-generated method stub

return cpu + "," + ram + "," + storage;

}

}

abstract class BluePrint {

abstract public void setCpu();

abstract public void setRam();

abstract public void setStorage();

abstract Computer getComputer();

}

class LgGramBlueprint extends BluePrint {

private Computer computer;

public LgGramBlueprint() {

computer = new Computer("default", "default", "default");

}

@Override

public void setCpu() {

computer.setRam("i7");

}

@Override

public void setRam() {

computer.setRam("8G");

}

@Override

public void setStorage() {

computer.setStorage("246SSD");

}

@Override

Computer getComputer() {

// TODO Auto-generated method stub

return computer;

}

}

class ComputerFactory{

private BluePrint print;

// 방법2 여기서 인자값을 private String cpu ram 만들고 직접 값을넣어준후

public void setPrint(BluePrint print) {

this.print = print;

}

public void make(){

print.setCpu();

print.setRam();

print.setStorage();

}

public Computer getComputer(){

// 여기에서 new로 만들어서 리턴해도됨

return print.getComputer();

}

}

**실무에서의 예제 - 많은 인자값이 있을시**

package designpattern;

public class BuilderPattern2 {

public static void main(String[] args) {

// Computer computer = new Computer("","",""); // 값의 순서를 잘못 입력했을 경우의

// 문제도잇음 ==> 값이많을대는 해깔림

Computer computer2 = ComputerBuilder

//.start()

//.setCpu("i7")

.startWithCpu("i7") // 위 2개와는 달리 시작과 동시에 cpu정보를 넣어 줄 수도 잇음

.setRam("8G") // 초기화하는 명칭을 알 수 있음

.setStorage("128G SSD")

.build();

System.out.println(computer2.toString());

}

}

class Computer {

private String cpu;

private String ram;

private String storage;

/\*

\* @param cpu 입니다.

\*

\* @param rma 입니다.

\*

\* @param storage 입니다.

\*/

public Computer(String cpu, String ram, String storage) { // 변수값이 엄청 많을대

super();

this.cpu = cpu;

this.ram = ram;

this.storage = storage;

}

public String getCpu() {

return cpu;

}

public void setCpu(String cpu) {

this.cpu = cpu;

}

public String getRam() {

return ram;

}

public void setRam(String ram) {

this.ram = ram;

}

public String getStorage() {

return storage;

}

public void setStorage(String storage) {

this.storage = storage;

}

@Override

public String toString() {

// TODO Auto-generated method stub

return cpu + "," + ram + "," + storage;

}

}

class ComputerBuilder {

private Computer computer;

private ComputerBuilder() {

computer = new Computer("default", "default", "default");

}

public static ComputerBuilder start() {

return new ComputerBuilder();

}

public static ComputerBuilder startWithCpu(String cpu){

ComputerBuilder builder = new ComputerBuilder();

builder.computer.setCpu(cpu);

return builder;

}

// 시작과 동시에 cpu 를 넣어 줄수도 있음

public ComputerBuilder setCpu(String string) {

computer.setCpu(string);

return this;

}

public ComputerBuilder setRam(String string) {

computer.setRam(string);

return this;

}

public Computer build() {

return this.computer;

}

public ComputerBuilder setStorage(String storage){

computer.setStorage(storage);

return this;

}

}