* Factory pattern are o singura clasa Factory, care returneaza diferite obiecte de la alte clase bazate pe if-uri ce decid ce obiect sa fie creat in dependenta de ce parametru oferim noi.
* In Abstract factory pattern, scapam de if-uri si avem o factory class pentru fiecare sub clasa. Apoi, vom avea o clasa Abstract Factory care va returna obiecte in dependeta de input folosind clasele factory.
* Abstract Fcatory se bazeaza pe aceea ce cream o interfata sau o clasa abstracta pentru Factory, si deci fiecare Factory de mai apoi va implementa aceasta interfata sau clasa abstracta. Deci, aici chiar cream mai multe factory
* Fiecare Factory lucreaza cu propriul obiect de construit, desi mereu returneaza un tip abstract, adia o interfata sau clasa abstracta, gen Computer, nu o implementare concreta.

public abstract class Computer {

public abstract String getRAM();

public abstract String getHDD();

public abstract String getCPU();

@Override

public String toString(){

return "RAM= "+this.getRAM()+", HDD="+this.getHDD()+", CPU="+this.getCPU();

}

}

public class PC extends Computer {

private String ram;

private String hdd;

private String cpu;

public PC(String ram, String hdd, String cpu){

this.ram=ram;

this.hdd=hdd;

this.cpu=cpu;

}

@Override

public String getRAM() {

return this.ram;

}

@Override

public String getHDD() {

return this.hdd;

}

@Override

public String getCPU() {

return this.cpu;

}

}

public class Server extends Computer {

private String ram;

private String hdd;

private String cpu;

public Server(String ram, String hdd, String cpu){

this.ram=ram;

this.hdd=hdd;

this.cpu=cpu;

}

@Override

public String getRAM() {

return this.ram;

}

@Override

public String getHDD() {

return this.hdd;

}

@Override

public String getCPU() {

return this.cpu;

}

}

public interface ComputerAbstractFactory {

public Computer createComputer();

}

public class PCFactory implements ComputerAbstractFactory {

private String ram;

private String hdd;

private String cpu;

public PCFactory(String ram, String hdd, String cpu){

this.ram=ram;

this.hdd=hdd;

this.cpu=cpu;

}

@Override

public Computer createComputer() {

return new PC(ram,hdd,cpu);

}

}

public class ServerFactory implements ComputerAbstractFactory {

private String ram;

private String hdd;

private String cpu;

public ServerFactory(String ram, String hdd, String cpu){

this.ram=ram;

this.hdd=hdd;

this.cpu=cpu;

}

@Override

public Computer createComputer() {

return new Server(ram,hdd,cpu);

}

}

public class ComputerFactory {

public static Computer getComputer(ComputerAbstractFactory factory){

return factory.createComputer();

}

}