**Prototype Design Pattern**

[](https://cdn.journaldev.com/wp-content/uploads/2013/06/prototype-design-pattern.jpg)

Prototype design pattern is used when the Object creation is a costly affair and requires a lot of time and resources and you have a similar object already existing.

Prototype pattern provides a mechanism to copy the original object to a new object and then modify it according to our needs. Prototype design pattern uses java cloning to copy the object.

**Prototype Design Pattern Example**

It would be easy to understand prototype design pattern with an example. Suppose we have an Object that loads data from database. Now we need to modify this data in our program multiple times, so it’s not a good idea to create the Object using new keyword and load all the data again from database.

The better approach would be to clone the existing object into a new object and then do the data manipulation.

Prototype design pattern mandates that the Object which you are copying should provide the copying feature. It should not be done by any other class. However whether to use shallow or deep copy of the Object properties depends on the requirements and its a design decision.

Here is a sample program showing Prototype design pattern example in java.

Employees.java

import java.util.ArrayList;

import java.util.List;

public class Employees implements Cloneable{

private List<String> empList;

public Employees(){

empList = new ArrayList<String>();

}

public Employees(List<String> list){

this.empList=list;

}

public void loadData(){

//read all employees from database and put into the list

empList.add("Pankaj");

empList.add("Raj");

empList.add("David");

empList.add("Lisa");

}

public List<String> getEmpList() {

return empList;

}

@Override

public Object clone() throws CloneNotSupportedException{

List<String> temp = new ArrayList<String>();

for(String s : this.getEmpList()){

temp.add(s);

}

return new Employees(temp);

}

}

Notice that the clone method is overridden to provide a deep copy of the employees list.

Here is the prototype design pattern example test program that will show the benefit of prototype pattern.

PrototypePatternTest.java

import java.util.List;

import com.journaldev.design.prototype.Employees;

public class PrototypePatternTest {

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

Employees emps = new Employees();

emps.loadData();

//Use the clone method to get the Employee object

Employees empsNew = (Employees) emps.clone();

Employees empsNew1 = (Employees) emps.clone();

List<String> list = empsNew.getEmpList();

list.add("John");

List<String> list1 = empsNew1.getEmpList();

list1.remove("Pankaj");

System.out.println("emps List: "+emps.getEmpList());

System.out.println("empsNew List: "+list);

System.out.println("empsNew1 List: "+list1);

}

}

Output of the above prototype design pattern example program is:

emps List: [Pankaj, Raj, David, Lisa]

empsNew List: [Pankaj, Raj, David, Lisa, John]

empsNew1 List: [Raj, David, Lisa]

If the object cloning was not provided, we will have to make database call to fetch the employee list every time. Then do the manipulations that would have been resource and time consuming.

That’s all for prototype design pattern in java.