Prototype Design Pattern in Java

MAY 2, 2018 BY PANKAJ — 18 COMMENTS

Prototype design pattern is one of the Creational Design pattern, so it provides a mechanism of object creation.

Prototype Design Pattern

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`

```
package com.journaldev.design.prototype;

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");
```

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`

```
package com.journaldev.design.test;

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);
```

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.

« PREVIOUS
Builder Design Pattern in Java

NEXT »
Adapter Design Pattern in Java

About Pankaj
If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: DESIGN PATTERNS

Comments

Ying Zhang says
JUNE 8, 2018 AT 8:24 AM
The Clone method still use new to correct object. it does not meet the design pattern requirement.

"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."
It do save time to get data from DB.
Is below clone method better?
@Override
protected Object clone() throws CloneNotSupportedException {
Object ee = (Employees)super.clone();
List tmp = new ArrayList();
((Employees) ee).setEmpList(tmp);
for(String s: this.empList)
tmp.add(s);
return ee;
}
Reply

Amine KOUIS says
NOVEMBER 12, 2017 AT 6:53 AM
Thanks! Very useful
Reply

Ganesh AC says
SEPTEMBER 27, 2017 AT 7:12 AM
Hi Pankaj, Could you please provide JDK example as well for prototype design pattern?
Reply

Arun SIngh says
SEPTEMBER 26, 2017 AT 11:27 AM
Thanks, very nice expl
Reply

Tahmid says
DECEMBER 20, 2016 AT 9:23 AM
Dear Pankaj,
I was experimenting with the code you provided here. Found something really weird that I couldn't understand. Please check the following method.

@Override
public Object clone() throws CloneNotSupportedException {
return new PlayersList(empList);
}
Instead of returning the 'temp' ArrayList, I returned the 'empList'. output like this:
emps HashMap : [Pankaj, Raj, David, Lisa]
empsNew HashMap : [Raj, David, Lisa, John]
empsNew1 HashMap : [Raj, David, Lisa, John]
I think there is something that I have failed to understand. An explanation would be of great help.
Thanks a lot for these nice, useful and easy to implement tutorials.
Reply

Tahmid says
DECEMBER 20, 2016 AT 10:09 AM
Just got the answer. . .
The clone generally shares state with the object being cloned. If that state is mutable, you don't have two independent objects. If you modify one, the other changes as well. And all of a sudden, you get random behavior.
– Josh Bloch
Reply

Pankaj says
APRIL 26, 2018 AT 10:14 AM
Yes, you need to understand concept of shallow copy and deep copy.
I hope you got it now.
Reply

salil says
DECEMBER 12, 2016 AT 9:40 PM
what is use of implementing Cloneable interface in this example and suppose i removed cloneable interface then what will happened.it will clone object,i tried to remove cloneable interface but same output came,please explain
Reply

sharad chandra says
FEBRUARY 28, 2017 AT 3:23 AM

If you look at the javadoc of Cloneable, it states that Invoking Object's clone method on an instance that does not implement the Cloneable interface results in the exception CloneNotSupportedException being thrown.
In the above example, clone() method was overridden to show Deep copy(here new ArrayList is created)
Removing of Cloneable interface did not impacted your code because, clone() method worked as normal method.
But if you comment the clone() method from Employees class and try to use Object's clone() method, it will throw exception.
Reply

Sandeep says
SEPTEMBER 28, 2016 AT 12:26 AM
I got a doubt after implementing this design pattern. When I update database using changed list lets say List list = empsNew.getEmpList();
list.add("John"); whith this list and I will get list1 like List list1 = empsNew1.getEmpList(); then I will get old list. So isn't it have data inconsistency issue?
Reply

Sashi says
SEPTEMBER 13, 2016 AT 6:41 AM
Hi Pankaj,
i can very well call the below 2 lines of code with creating a new object /calling a db
emps.getEmpList().add("John");
emps.getEmpList().remove("Pankaj");
I havent understood the exact thing.. can u please explain.
Reply

Sashi says
SEPTEMBER 13, 2016 AT 6:42 AM
sry it is without creating a new object**
Reply

Kunal Bansal says

OCTOBER 27, 2015 AT 2:18 AM

what would happen if I would add one employee("John") in the list (list) and save it to the database, and from (list1) i want to remove the "John" but it would say it does not exist?
can you please elaborate more onhere can we use this

Reply

DEBARATI MAJUMDER says

AUGUST 13, 2017 AT 6:45 AM

Hi Kunal,
I feel – addition/removal of employees and saving into database are not real objective here. This is just for an example.
The main objective of prototype design is – we can use the clone of an existing object instead of creating a new one. Now what you will do with this objects (original and clone) depends on your requirements.

Reply

Supriya says

OCTOBER 14, 2015 AT 5:11 AM

Your tutorials arereally wonderful and easy to understand.I was wondering where can we use prototype pattern in real time.

Reply

Vinod says

DECEMBER 17, 2014 AT 9:58 AM

Program says List.. but it is printing as hashmap.. how it is possible?
System.out.println("emps List: "+emps.getEmpList());
System.out.println("empsNew List: "+list);
System.out.println("empsNew1 List: "+list1

Reply

Amit N says

APRIL 22, 2014 AT 4:56 AM

Hi Pankaj,
Nice explaination.
One correction, it is written as 'Notice that the clone method is overridden to provide a deep copy of the employees list.' But clone method created shallow copy and not deep copy.

Reply

kaushal says

JULY 24, 2014 AT 12:52 AM

Hi, By default clone method created shallow copy but here we want deep copy. We have to provide the definition for deep copy in clone method by our self.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

GET IT ON Google Play

DESIGN PATTERNS TUTORIAL

RECOMMENDED TUTORIALS

- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial