Java Cloning - Copy Constructor versus Cloning (https://programmingmitra.blogspot.in/2016/05/java-cloning-copy-constructor-versus-Object-clone-or-cloning)

posted by Naresh Joshi (https://plus.google.com/108302142446482002391) | on January 10, 2017 | 2 comments (https://programmingmitra.blogspot.in/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html#comment-form)

In my previous article Java Cloning and Types of Cloning (Shallow and Deep) in Details with Example (https://programmingmitra.blogspot.in/2016/11/Java-Cloning-Types-of-Cloning-Shallow-Deep-in-Details-with-Example.html), I have discussed Java Cloning in details and answered questions about how we can use cloning to copy objects in Java, what are two different types of cloning (Shallow & Deep) and how we can implement both of them, if you haven't read it please go ahead.

In order to implement cloning, we need configure our classes to follow below steps

CONNECT WITH US

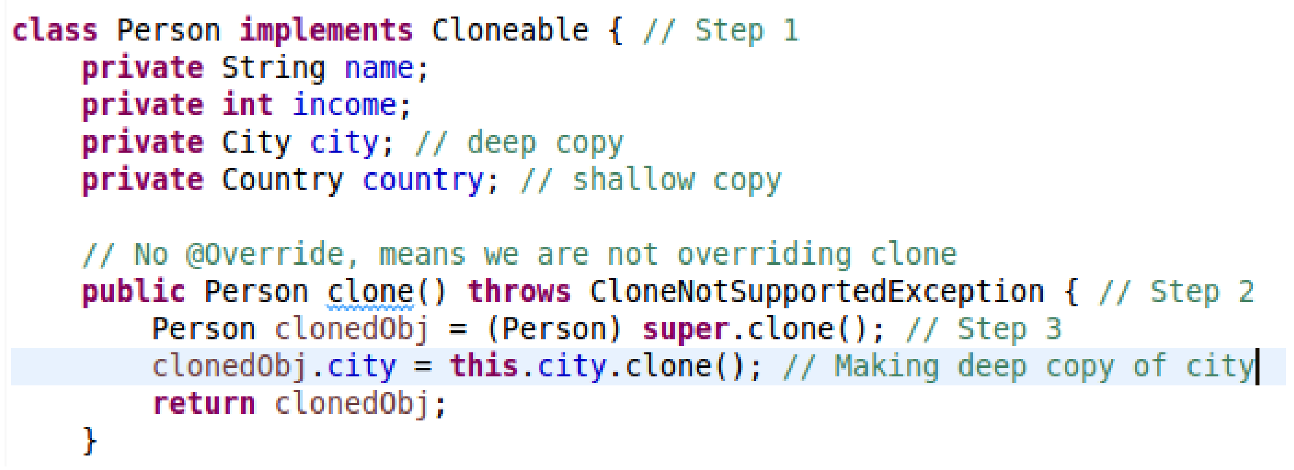
FACEBOOK (HTTP://WWW.FACEBOOK.COM/PROGRAMMINGMITRA)

GOOGLE (HTTP://PLUS.GOOGLE.COM/B/103534373840551473813/COMMUNITIES/106845742620040840023)

LINKEDIN (HTTPS://WWW.LINKEDIN.COM/IN/NARESH-JOSHI-89493255?TRK=NAV_RESPONSIVE_TAB_PROFILE_PIC)

POPULAR POSTS

- Implement Cloneable interface in our class or its superclass or interface,
- Define clone() method which should handle CloneNotSupportedException (either throw or log),
- And in most cases from our clone() method we call the clone() method of the superclass.



```
class Person implements Cloneable { // Step 1
    private String name;
    private int income;
    private City city; // deep copy
    private Country country; // shallow copy

    // No @Override, means we are not overriding clone
    public Person clone() throws CloneNotSupportedException { // Step 2
        Person clonedObj = (Person) super.clone(); // Step 3
        clonedObj.city = this.city.clone(); // Making deep copy of city
        return clonedObj;
    }
```

(//2.bp.blogspot.com/-Mmj1Gv6Y6mI/WHUGporMneI/AAAAAAAAKXQ/a3ztA69ucFEkWwknE4BFGqu_IiK7lmqdwCK4B/s1600/Java%2BCloning.png)

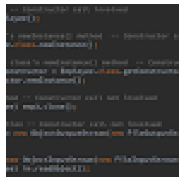
And super.clone() will call its super.clone() and chain will continue until call will reach to clone() method of the Object class which will create a field by field mem copy of our object and return it back.

Like everything Cloning also comes with its advantages and disadvantages. However, Java cloning is more famous its design issues but still, it is the most common and popular cloning strategy present today.

Advantages of Object.clone()

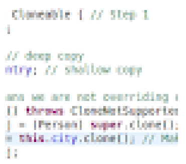
(https://programmingmitra.blogspot.in/2017/02/automatic-spring-data-jpa-auditing-saving-CreatedBy-createddate-lastmodifiedby-lastmodifieddate-automatically.html)

Spring Data JPA Auditing: Saving CreatedBy, CreatedDate, LastModifiedBy, LastModifiedDate automatically (https://programmingmitra.blogspot.in/2017/02/automatic-spring-data-jpa-auditing-saving-CreatedBy-createddate-lastmodifiedby-lastmodifieddate-automatically.html)



(https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html)

5 Different ways to create objects in Java with Example (https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html)



(https://programmingmitra.blogspot.in/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html)

Java Cloning - Copy Constructor versus Cloning (https://programmingmitra.blogspot.in/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html)

Object.clone() have many design issues but it is still the popular and easiest way of copying objects, Some advantages of using clone() are

- Cloning requires very less line of code, just an abstract class with 4 or 5 line long clone() method but we will need to override it if we need deep cloning.
- It is the easiest way of copying object specially if we are applying it to an already developed or an old project. We just need to define a parent class, implement Cloneable in it, provide the definition of clone() method and we are ready every child of our parent will get the cloning feature.
- We should use clone to copy arrays because that’s generally the fastest way to do it.
- As of release 1.5, calling clone on an array returns an array whose compile-time type is the same as that of the array being cloned which clearly means calling clone on arrays do not require type casting.

Disadvantages of Object.clone()

Below are some cons due to which many developers don't use Object.clone()

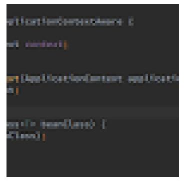
- Using Object.clone() method requires us to add lots of syntax to our code like implement Cloneable interface, define clone() method and handle CloneNotSupportedException and finally call to Object.clone() and cast it our object.
- Cloneable interface lacks clone() method, actually, Cloneable is a marker interface and doesn’t have any method in it and still, we need to implement it just to tell JVM that we can perform clone() on our object.
- Object.clone() is protected so we have to provide our own clone() and indirectly call Object.clone() from it.
- We don’t have any control over object construction because Object.clone() doesn’t invoke any constructor.

(https://programmingmitra.blogspot.in/2017/02/automatic-jpa-auditing-persisting-audit-logs-automatically-using-entityListeners.html)

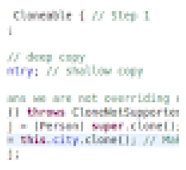
JPA Auditing: Persisting Audit Logs Automatically using EntityListeners (https://programmingmitra.blogspot.in/2017/02/automatic-jpa-auditing-persisting-audit-logs-automatically-using-entityListeners.html)

(https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html)

Creating objects through Reflection in Java with Example (https://programmingmitra.blogspot.in/2016/05/creating-objects-through-reflection-in-java-with-example.html)


(https://programmingmitra.blogspot.in/2017/03/AutoWiring-Spring-Beans-Into-Classes-Not-Managed-By-Spring-Like-JPA-Entity-Listeners.html)

AutoWiring Spring Beans Into Classes Not Managed By Spring Like JPA Entity Listeners (https://programmingmitra.blogspot.in/2017/03/AutoWiring-Spring-Beans-Into-Classes-Not-Managed-By-Spring-Like-JPA-Entity-Listeners.html)


(https://programmingmitra.blogspot.in/2016/11/Java-Cloning-Types-of-Cloning-Shallow-Deep-in-Details-with-Example.html)

- If we are writing clone method in a child class e.g. Person then all of its superclasses should define clone() method in them or inherit it from another parent class otherwise super.clone() chain will fail.
- Object.clone() support only shallow copy so reference fields of our newly cloned object will still hold objects which fields of our original object was holding. In order to overcome this, we need to implement clone() in every class whose reference our class is holding and then call their clone them separately in our clone() method like in below example.
- We can not manipulate final fields in Object.clone() because final fields can only be changed through constructors. In our case, if we want every Person objects to be unique by id we will get the duplicate object if we use Object.clone() because Object.clone() will not call the constructor and final final id field can't be modified from Person.clone().

```
class City implements Cloneable {
    private final int id;
    private String name;
    public City clone() throws CloneNotSupportedException {
    return (City) super.clone();
    }
}

class Person implements Cloneable {
    public Person clone() throws CloneNotSupportedException {
        Person clonedObj = (Person) super.clone();
        clonedObj.name = new String(this.name);
        clonedObj.city = this.city.clone();
        return clonedObj;
    }
}
```

Because of above design issues with Object.clone() developers always prefer other ways to copy objects like using

- BeanUtils.cloneBean(object) creates a shallow clone similar to Object.clone().

Java Cloning and Types of Cloning (Shallow and Deep) in Details with Example (https://programmingmitra.blogspot.in/2016/11/Java-Cloning-Types-of-Cloning-Shallow-Deep-in-Details-with-Example.html)

(https://programmingmitra.blogspot.in/2017/05/how-does-jvm-handle-method-overriding-internally.html)

How Does JVM Handle Method Overloading and Overriding Internally (https://programmingmitra.blogspot.in/2017/05/how-does-jvm-handle-method-overriding-internally.html)

(https://programmingmitra.blogspot.in/2016/02/embedding-github-code-into-your-blogger.html)

Embedding Github code into your Blogger blog (https://programmingmitra.blogspot.in/2016/02/embedding-github-code-into-your-blogger.html)

(https://programmingmitra.blogspot.in/2016/05/java-build-tools-comparisons-ant-vs.html)

Java Build Tools Comparisons: Ant vs Maven vs Gradle (https://programmingmitra.blogspot.in/2016/05/java-build-tools-comparisons-ant-vs.html)

CATEGORIES

- SerializationUtils.clone(object) creates a deep clone. (i.e. the whole properties graph is cloned, not only the first level), but all classes must implement Serializable.
- Java Deep Cloning Library (https://code.google.com/p/cloning/) offers deep cloning without the need to implement Serializable.

All these options require the use of some external library plus these libraries will also be using Serialization or Copy Constructors or Reflection internally to copy our object. So if you don't want to go with above options or want to write our own code to copy the object then you can use

1. **Serialization**
2. **Copy Constructors**

Serialization

I have discussed in 5 Different ways to create objects in Java with Example (https://programmingmitra.blogspot.in/2016/05/different-ways-to-create-objects-in-java-with-example.html) how we can create a new object using serialization. Similarly, we can also use serialization to copy an object by first Serialising it and again deserializing it like it is done below or we can also use other APIs like JAXB which supports serialization.

Ant (https://programmingmitra.blogspot.in/search/label/Ant)

Blog Writing (https://programmingmitra.blogspot.in/search/label/Blog%20Writing)

Build Tools (https://programmingmitra.blogspot.in/search/label/Build%20Tools)

Gist (https://programmingmitra.blogspot.in/search/label/Gist)

Github (https://programmingmitra.blogspot.in/search/label/Github)

Gradle (https://programmingmitra.blogspot.in/search/label/Gradle)

How in Java (https://programmingmitra.blogspot.in/search/label/How%20in%20Java)

Interview Q&A (https://programmingmitra.blogspot.in/search/label/Interview%20Q%26A)

Java (https://programmingmitra.blogspot.in/search/label/Java)

Java 8 (https://programmingmitra.blogspot.in/search/label/Java%208)

Java Cloning (https://programmingmitra.blogspot.in/search/label/Java%20Cloning)

JPA (https://programmingmitra.blogspot.in/search/label/JPA)

```
public Person copy(Person original) {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("data.obj"));
        ObjectInputStream in = new ObjectInputStream(new FileInputStream("data.obj"))) {

        // Serialization
        out.writeObject(original);

        //De Serialization
        return (Person) in.readObject();
    } catch (Exception e) {
        throw new RuntimeException(e); // log exception or throw it by your own way
    }
}
```

But serialization is not solving any problem because we will still not be able to modify the final fields, still don't have any control on object construction, still needs to implement Serializable which is again similar to Cloneable. Plus serialization process is slower then Object.clone().

Copy Constructors

This method copying object is most popular between developer community it overcomes every design issue of Object.clone() and provides better control over object construction

```
public Person(Person original) {
    this.id = original.id + 1;
    this.name = new String(original.name);
    this.city = new City(original.city);
}
```

Advantages of copy constructors over Object.clone()

Copy constructors are better than Object.clone() because they

Lombok (https://programmingmitra.blogspot.in/search/label/Lombok)
Maven (https://programmingmitra.blogspot.in/search/label/Maven)
Reflection API (https://programmingmitra.blogspot.in/search/label/Reflection%20API)
Solr (https://programmingmitra.blogspot.in/search/label/Solr)
Spring (https://programmingmitra.blogspot.in/search/label/Spring)
Spring Boot (https://programmingmitra.blogspot.in/search/label/Spring%20Boot)
Spring Data (https://programmingmitra.blogspot.in/search/label/Spring%20Data)
What in Java (https://programmingmitra.blogspot.in/search/label/What%20in%20Java)
Why in Java (https://programmingmitra.blogspot.in/search/label/Why%20in%20Java)

- Don’t force us to implement any interface or throw any exception but we can surely do it if it is required.
- Don’t require any casts.
- Don’t require us to depend on an unknown object creation mechanism.
- Don’t require parent class to follow any contract or implement anything.
- Allow us to modify final fields.
- Allow us to have complete control over object creation, we can write our initialization logic in it.

By using Copy constructors strategy, we can also create conversion constructors which can allow us to convert one object to another object e.g. ArrayList(Collection<? extends E> c) constructor generates an ArrayList from any Collection object and copy all items from Collection object to newly created ArrayList object.

Naresh Joshi

()

BLOG ARCHIVE

RELATED POSTS

(https://programmingmitra.blogspot.com/2016/10/why-a-java-class-can-not-be-private-or-protected.html)

Why an outer Java class can't be private or protected (https://programmingmitra.blogspot.com/2016/10/why-a-java-class-can-not-be-private-or-protected.html)

(https://programmingmitra.blogspot.com/2016/06/why-java-is-purely-object-oriented-or-why-not.html)

Why Java is Purely Object Oriented Language Or Why Not (https://programmingmitra.blogspot.com/2016/06/why-java-is-purely-object-oriented-or-why-not.html)

(https://programmingmitra.blogspot.com/2016/06/java-lambda-expression-explained-with-example.html)

Java Lambda Expression Explained with Example (https://programmingmitra.blogspot.com/2016/06/java-lambda-expression-explained-with-example.html)

2017/01/Project-Lombok-The-Boilerplate-Code-Extractor.html)

Java Cloning - Even Copy Constructors Are Not Suff... (https://programmingmitra.blogspot.in/2017/01/java-cloning-why-copy-constructors-are-not-sufficient-or-good.html)

Java Cloning - Copy Constructor versus Cloning (https://programmingmitra.blogspot.in/2017/01/Java-cloning-copy-constructor-versus-Object-clone-or-cloning.html)

► 2016 (https://programmingmitra.blogspot.in/2016/) (16)

SUBMIT A QUERY OR SUGGESTION

Name	
Email *	
Message *	

Send

« Newer Post (https://programmingmitra.blogspot.in/2017/01/java-cloning-why-copy-constructors-are-not-sufficient-or-good.html)

Older Post (https://programmingmitra.blogspot.in/2016/11/Java-Cloning-Types-of-Cloning-Shallow-Deep-in-Details-with-Example.html) »

2 Comments :

1. **YouLoseBellyFat (Https://Www.Blogger.Com/Profile/18175607118884749966)** 1

   APRIL 08, 2017 6:47 PM (HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2017/01/JAVA-CLONING-COPY-CONSTRUCTOR-VERSUS-OBJECT-CLONE-OR-CLONING.HTML?SHOWCOMMENT=1491657429976#C8774234096226393513)

   applets for java (http://java.happycodings.com/applets/) for all people

   Reply

2. **For IT The (Https://Www.Blogger.Com/Profile/01234222908168002434)** 2

   NOVEMBER 13, 2017 10:53 AM (HTTPS://PROGRAMMINGMITRA.BLOGSPOT.COM/2017/01/JAVA-CLONING-COPY-CONSTRUCTOR-VERSUS-OBJECT-CLONE-OR-CLONING.HTML?SHOWCOMMENT=1510550616012#C830203209865835816)

Hi, Great.. Tutorial is just awesome..It is really helpful for a newbie like me.. I am a regular follower of your blog. Really very informative post you shared here. Kindly keep blogging. If anyone wants to become a Java developer learn from Java Training in Chennai (http://wisentechnologies.com/it-courses/java-training.aspx). or learn thru Java Online Training in India (http://wisenitsolutions.com/IT-Courses/Java-Training) . Nowadays Java has tons of job opportunities on various vertical industry.

Reply

(https://www.blogger.com/comment-iframe.g?blogID=3832771837877502009&postID=4453765617221832626&blogspotRpcToken=9850920)