**Prototype Pattern**

**Use Case:**

Let's say we want to create an exact copy of an object. Usually, we create a new instance of the same class and go through all fields of the object and copy it into a newly created object members, few drawbacks with this approach are

* Creating a new instance of the class is a costly operation
* Some of the object fields might be private and may not be accessible from outside
* We must know the object's all fields to copy into the new instance

**Prototype Pattern**

prototype Pattern comes under the creational design Pattern. This pattern is

* basically, the creation of a new instance by cloning an existing instance.
* Used when object creation is a time-consuming and costly operation

**Implementation**

The Prototype pattern delegates the cloning process to the actual objects that are being cloned.

* One of the best available ways to create an object from existing objects is the clone() method.
* An object that supports cloning is called a *prototype*

Example:

package org.designpattern.prototype;  
  
public interface Animal extends Cloneable{  
 public Animal makecopy();  
}

package org.designpattern.prototype;

public class Sheep implements Animal {  
  
 int j =1;  
  
 public Sheep(int j) {  
 this.j = j;  
 System.*out*.println("Sheep is made");  
 }  
  
 public int getJ() {  
 return j;  
 }  
  
 public void setJ(int j) {  
 this.j = j;  
 }  
  
 @Override  
 public Animal makecopy() {  
 System.*out*.println("Sheep is made");  
 Sheep sheepobj = null;  
 try {  
 sheepobj = (Sheep) super.clone();  
 } catch (CloneNotSupportedException e) {  
 throw new RuntimeException(e);  
 }  
 return sheepobj;  
 }  
 @Override  
 public String toString() {  
 return "Sheep{" +  
 "j=" + j +  
 '}';  
 }  
}

1. **Static block initialization**

Static block initialization implementation is similar to eager initialization, except that an instance of the class is created in the static block that provides the option for exception handling.

package org.designpattern.prototype;  
  
public class CloneFactory {  
 public Animal getClone (Animal animalSample){

return animalSample.makecopy();

}

}

package org.designpattern.prototype;  
  
public class TestingCloning {  
  
 public static void main(String[] args){  
 CloneFactory animalMaker = new CloneFactory();  
 Sheep sally = new Sheep(2);  
 Sheep clonedSheep = (Sheep) animalMaker.getClone(sally);  
  
 System.*out*.println(sally);  
  
 System.*out*.println(clonedSheep);  
 System.*out*.println("sally hash" + System.*identityHashCode*(sally));  
 System.*out*.println("sally hash" + System.*identityHashCode*(clonedSheep));  
  
 }  
}