Prototype ve Object Pool

1. Prototype ve Object Pool Design Pattern

1.1. 1.Prototype Tasarım Kalıbı

Bir sistemde kullanılan nesnelerin nasıl oluşturulduğu,birleştirildiğinden bağımsız bir yapı olması gerektiğinde prototip tasarım kalıbı kullanılmaktadır.

> Makul sayıdaki prototipi yüklemek ve sınıfı her seferinde uygun durumlar için tekrar oluşturmak yerine bunları klonlamak daha uygun olarak düşünülebilir.

Bu kodda iki farklı kopyalama çeşidi kullanılmıştır.

- **Shallow Copy:** Oluşturulan iki, bellekte aynı yeri gösterirse bu sığ kopyalamadır.
- **Deep Copy:** Nesnelerin içerikleri aynı olup bellekte gösterdikleri yer farklı olursa bu kopyalama işlemine derin kopyalama denir.

```
public class Main {

public static void main(String[] args) throws
CloneNotSupportedException {
 // TODO Auto-generated method stub

 Adres adres=new Adres("Bura" , "Ora" , 07);
 Sirket sirket= new Sirket("Istanbul" , "Turkiye" , adres);
 Calisan calisan1= new Calisan( 1 , "Ayse" , sirket , adres);
 Calisan calisan2= (Calisan) calisan1.clone();


   System.out.println("calisan1 -->" + calisan1 );
   System.out.println("calisan2 -->" + calisan2 );

   // Shallow Clone'da calisan2'nin sehrini değiştirirsem calisan1 de
değişecek.
   calisan2.getSirket().setSehir("Kocaeli");
   System.out.println("calisan1 -->" + calisan1 );
   System.out.println("calisan2 -->" + calisan2 );
```

```
    // Deep Clone'da calisan2'nin adresini değiştirirsem calisan1 aynı
 kalacak.
    calisan2.getAdres().setSokak("BuraDeğil");
    System.out.println("calisan1 -->" + calisan1 );
    System.out.println("calisan2 -->" + calisan2 );
 }}
```

```
public class Calisan implements Cloneable{

 private int calisanID;
 private String calisanAd;
 private Sirket sirket;
 private Adres adres;

 public Calisan(int calisanID, String calisanAd, Sirket sirket, Adres
 adres) {
  super();
  this.calisanID = calisanID;
  this.calisanAd = calisanAd;
  this.sirket = sirket;
  this.adres= adres;
 }

 @Override
 protected Object clone() throws CloneNotSupportedException {
  // TODO Auto-generated method stub
  Calisan calisan = (Calisan) super.clone();
  calisan.setAdres((Adres)adres.clone());
  return calisan;
 }

 public int getCalisanID() {
  return calisanID;
 }
 public void setCalisanID(int calisanID) {
  this.calisanID = calisanID;
 }
 public String getCalisanAd() {
  return calisanAd;
 }
 public void setCalisanAd(String calisanAd) {
  this.calisanAd = calisanAd;
 }
 public Sirket getSirket() {
  return sirket;
```

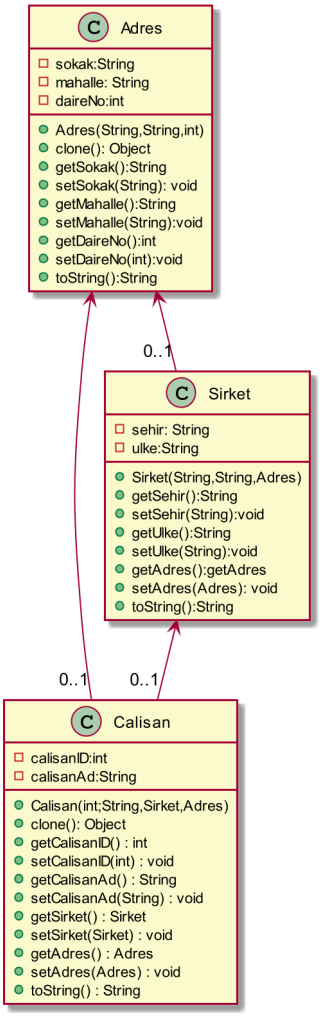
```
 }
 public void setSirket(Sirket sirket) {
  this.sirket = sirket;
 }
 public Adres getAdres() {
  return adres;
 }


 public void setAdres(Adres adres) {
  this.adres = adres;
 }

 @Override
 public String toString() {
  // TODO Auto-generated method stub
  return "Calisan [calisanID=" + calisanID + ", calisanAd="+ calisanAd
    + ", sirket=" + sirket + ", adres=" + adres+ "]";
 }}
```

```
public class Sirket implements Cloneable {

 private String sehir;
 private String ulke;
 private Adres adres;

 public Sirket(String sehir, String ulke, Adres adres) {
  super();
  this.sehir = sehir;
  this.ulke = ulke;
  this.adres = adres;
 }

 public String getSehir() {
  return sehir;
 }
 public void setSehir(String sehir) {
  this.sehir = sehir;
 }
 public String getUlke() {
  return ulke;
 }
 public void setUlke(String ulke) {
  this.ulke = ulke;
 }
```

```
 public Adres getAdres() {
  return adres;
 }
 public void setAdres(Adres adres) {
  this.adres = adres;
 }
 @Override
 public String toString() {
  // TODO Auto-generated method stub
  return "Sirket [ sehir= " + sehir + ", ulke="+ ulke +
      "]" ;
 }}
```

```
public class Adres implements Cloneable{

 private String sokak;
 private String mahalle;
 private int daireNo;


 public Adres(String sokak, String mahalle, int daireNo) {
  super();
  this.sokak = sokak;
  this.mahalle = mahalle;
  this.daireNo = daireNo;
 }

 @Override
 protected Object clone() throws CloneNotSupportedException {
  // TODO Auto-generated method stub
  return super.clone();
 }

 public String getSokak() {
  return sokak;
 }

 public void setSokak(String sokak) {
  this.sokak = sokak;
 }


 public String getMahalle() {
  return mahalle;
 }
```

```
public void setMahalle(String mahalle) {
 this.mahalle = mahalle;
}

public int getDaireNo() {
 return daireNo;
}
public void setDaireNo(int daireNo) {
 this.daireNo = daireNo;
}
@Override
public String toString() {
 return "Adres [sokak=" + sokak + ", mahalle=" + mahalle + ", daireNo="
+ daireNo + "]";
}}
```



1.2. Object Pool Tasarım Kalıbı

İstenilen nesnelerin sürekli olarak üretilmesi yerine, başlangıçta bir havuzu oluşturulur ve bu havuz nesneler ile doldurulur.

Bu tasarım deseninde object pool sınıfına singleton tasarım deseni uygulanır. Singleton tasarım deseni, uygulanan nesnenin bellekte tek bir kopyasının olmasını, istenildiği durumlarda aynı instance gönderilmesini sağlar. Bu tasarım desenini object pool sınıfına uygulayarak, nesne havuzunun tek bir kopyasının olmasını sağlar.

```
public class MainApp {

     public static void main(String[] args) {
         ReusablePool pool = ReusablePool.getInstance();

         for ( int count = 0; count < ReusablePool.DEFAULT_POOL_SIZE +
 1; ++count ) {
             ReusableVisitor visitor = pool.acquireVisitor();
             if ( visitor == null ) {
                 System.out.println( count + ". Kapasite Dolu !" );
                 break;
             }

             System.out.println( count + ". Ziyaretci " +
 visitor.hashCode() );
             //pool.releaseVisitor(visitor);
         }

         for ( int count = 0; count < ReusablePool.DEFAULT_POOL_SIZE +
 1; ++count ) {
             ReusableVisitor visitor = pool.acquireVisitor();
             if ( visitor == null ) {
                 System.out.println( count + ". Kapasite Dolu ! " );
                 break;
             }

             System.out.println( count + ". Ziyaretci " +
 visitor.hashCode() );
             //pool.releaseVisitor(visitor);
         }} }
```

AcquireVisitor() : Bu metod ile objenin oluşturulmasını ya da hazırda olan objeyi bize vermesini bekliyoruz.

```
import java.util.ArrayList;

public class ReusablePool {
    public static final int DEFAULT_POOL_SIZE = 10; ❶

    private ArrayList< ReusableVisitor > reusables;
    private int maxPoolSize = DEFAULT_POOL_SIZE;
```

```
    protected static ReusablePool instance = null;

    protected ReusablePool()
    {
        reusables = new ArrayList< ReusableVisitor >();
    }

    public int getMaxPoolSize() {
        return maxPoolSize;
    }

    public void setMaxPoolSize(int maxPoolSize) {
        this.maxPoolSize = maxPoolSize;
    }


    public ReusableVisitor acquireVisitor()
    {


        for ( ReusableVisitor visitor : reusables) { ❷
            if ( !visitor.isInUse() ) {
                visitor.setInUse(true);

                return visitor;
            }
        }
          ❸

        if ( reusables.size() >= getMaxPoolSize() ) {

            return null;
        }
        ReusableVisitor visitor = new ReusableVisitor();
        visitor.setInUse(true);
        reusables.add(visitor);

        return visitor;
    }

    public void releaseVisitor( ReusableVisitor subject )
    {  ❹

        int idx  = reusables.indexOf( subject );
        if ( idx == -1 ) {
```

```
        return;
        }
        ReusableVisitor visitor = reusables.get( idx );
        visitor.setInUse(false);

    }


    public static ReusablePool getInstance() ❺
    {
        if ( instance == null ) {
            instance = new ReusablePool();
        }
    return instance;
    }}
```

❶ Havuzun kapasitesi belirlendi.

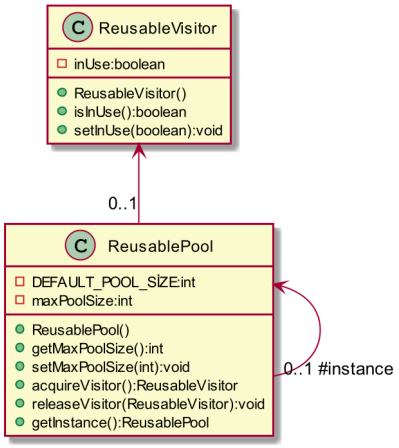
❷ Önceden var olan yeniden kullanılabilir bir nesneyi arar

❸ Yeni oluşturur ve havuza ekler.

❹ ReleaseVisitor() : Bu metod ile objeyi havuza iade ediyoruz.

❺ Singleton kalıbı uygulandı.

```
public class ReusableVisitor {

    private boolean inUse = false;

    public boolean isInUse() {
        return inUse;
    }
    public void setInUse(boolean inUse) {
        this.inUse = inUse;
    }}
```


ReusableVisitor
inUse:boolean
ReusableVisitor()
isInUse():boolean
setInUse(boolean):void
0..1
ReusablePool
DEFAULT_POOL_SİZE:int
maxPoolSize:int
ReusablePool()
getMaxPoolSize():int
setMaxPoolSize(int):void
acquireVisitor():ReusableVisitor
releaseVisitor(ReusableVisitor):void
getInstance():ReusablePool
0..1 #instance