1.4. Prototype Pattern

1. 1.4. Prototype Pattern

Prototip deseni genellikle sınıfın bir örneğine (prototip) sahip olduğumuzda kullanılır. Yalnızca prototipi kopyalayarak yeni nesneler oluşturmak isteriz. Bu noktada başvurulacak tasarım desenidir.

Örneğin bazı oyunlarda arka planda ağaçlar veya binalar mevcuttur. Karakter her hareket ettiğinde yeni ağaçlar veya binalar yaratmamız ve bunları ekranda oluşturmamız oldukça maliyetli bir işlem olacaktır.

Bu sebeple ilk önce ağacın bir örneği oluşturulur. Sonra bu örnekten (prototip) istenildiği ölçüde ağaç oluşturabilir ve konumları güncellenebilir. Nesne klonlamayı iki ana başlıkta inceleyebiliriz.

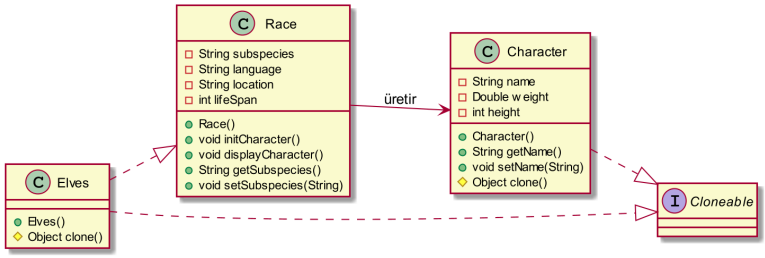

Figure 1. Prototype UML Diyagramı

1.1. 1.4.1 Shallow Copy

Bu klonlama tekniğinde yalnızca primitive değişkenler kopyalanır, object tipindeki değişkenlerin kopyalanma işlemi yapılmaz: Sınıf içerisindeki iç nesneler kopyalanmaz, "İç nesneler", orijinal nesne ile kopyası arasında paylaşılır.

Örneğimizde Race soyut sınıfı ve bazı değişkenleri görülmektedir. Ayrıca sınıf Character sınıfından bir nesneye de sahiptir.

Race.java.

```
public abstract class Race {

    private String subspecies;
```

```
    private String language;
    private String location;
    private int lifeSpan;

    Character charDetail;

    public abstract void initCharacter();

    public void displayRace() {
        System.out.println("");
        System.out.println("Subspecies: " + getSubspecies());
        System.out.println("Language(s): " + getLanguage());
        System.out.println("Location(s): " + getLocation());
        System.out.println("Average Life Span: " + getLifeSpan());
        System.out.println("-----CHAR------");
        System.out.println("Character Name: " +
getCharDetail().getName());
        System.out.println("---------------");
    }

    // Getter and setters...
```

Elves sınıfı, ırk sınıfından türeyen bir ırk türüdür. Elflere özel birtakım özellikleri **initCharacter()** metodu ile atanmaktadır.

Elf sınıfından bir nesne üretip karakter özelliklerini ayarlayacağız. Daha sonra bir elf ordusu kurmak adına bu özellikleri tanımlanan elf objesini, aynı adımları tekrardan yapmamak adına klonlayacağız.

Görüldüğü üzere Cloneable ara yüzünden clone() metodu override edilmiştir.

Elves.java.

```
public class Elves extends Race implements Cloneable{

 @Override
 public void initCharacter() {
  System.out.println("Now inside Elves::initCharacter()");

        // Set some predefined specs for Elves.
        setSubspecies("Quarter-elves");
        setLanguage("Common Speech");
        setLocation("Blue Mountains, Nilfgaardian Empire");
        setLifeSpan(350);
```

```
        // setChar default
        charDetail = new Character();
        charDetail.setName("Aen Elle");
 }

 @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }

}
```

Elf ırkından oluşan her bir birey için ise vücut bilgilerinin ayrıntılarının bulunduğu bir Character sınıfı mevcuttur.

Character.java.

```
public class Character {

 private String name;
 private double weight;
 private int height;

 // Getter and setters...
```

Örneği ve ardından çıktıyı inceleyelim.

MainApp.java.

```
public class MainApp {

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

  Elves elf = new Elves();
  elf.initCharacter();

  Elves clonedElf = (Elves) elf.clone();

  // SHALLOW COPY (cloning only primitives)
  clonedElf.setLanguage("New Elven Language");
  clonedElf.getCharDetail().setName("New Elves");

  System.out.println("Checking primitive data types: ");
  System.out.println("Original Elf Language: " + elf.getLanguage());
```

```
 System.out.println("Cloned Elf Language: " + clonedElf.getLanguage()
+ "\n");

 System.out.println("Checking objects: ");
 System.out.println("Original Elf Name: " +
elf.getCharDetail().getName());
 System.out.println("Cloned Elf Name: " +
clonedElf.getCharDetail().getName() + "\n");
 }
}
```

```
Checking primitive data types:
Original Elf Language: Common Speech
Cloned Elf Language: New Elven Language

Checking objects:
Original Elf Name: New Elves
Cloned Elf Name: New Elves
```

Görüldüğü üzere, primitive veri tipleri üzerinde yapılan değişiklikler iki nesnenin durumunu etkilemezken, nesne tipindeki referanslar, aynı adresi göstermekte ve birisi üzerindeki değişiklik, diğer nesneyi de etkilemektedir. **Shallow copy** yapıldığı taktirde bu durum gözlenmektedir.

1.2. 1.4.2 Deep Copy

Deep clone, çoğu durumda istenen davranıştır. Derin kopyada, orijinal nesneden bağımsız bir klon oluşturulur ve klonlanmış nesnede değişiklik yapmak orijinal nesneyi etkilememektedir.

Bunu gerçekleyebilmek için, **Elves** ve **Character** sınıflarındaki ilgili kısımlar aşağıdaki şekilde değiştirilmelidir.

Character sınıfı için de klon metodu override edilmelidir.

Character.java.

```
public class Character implements Cloneable {

 private String name;
 private double weight;
 private int height;
```

```
// Getter and setters...

@Override
   protected Object clone() throws CloneNotSupportedException {
       return super.clone();
   }
```

Elves sınıfı nesnesi, Character sınıfından bir nesne içermektedir. Dolayısıyla elf klonu için setCharDetail metoduna, Character sınıfından nesne, clon metodu çağırılarak set edilmelidir. Bu şekilde deep copy sağlanmış olur:

Elves.java.

```
   @Override
   protected Object clone() throws CloneNotSupportedException {
 Elves cloned = (Elves) super.clone();
 cloned.setCharDetail((Character)cloned.getCharDetail().clone());
       return cloned;
   }
```

Shallow copy için çalıştırdığımız kodu tekrardan çalıştırdığımızda, artık nesne referanslarının da birbirinden ayrıldığını görmüş oluruz:

MainApp.java.

```
public class MainApp {

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

  Elves elf = new Elves();
  elf.initCharacter();

  Elves clonedElf = (Elves) elf.clone();

  // DEEP COPY (also copies objects)
  clonedElf.setLanguage("New Elven Language");
  elf.getCharDetail().setName("Aen Elle");
  clonedElf.getCharDetail().setName("New Elves");

  System.out.println("Checking primitive data types: ");
  System.out.println("Original Elf Language: " + elf.getLanguage());
  System.out.println("Cloned Elf Language: " + clonedElf.getLanguage()
 + "\n");
```

```
  System.out.println("Checking objects: ");
  System.out.println("Original Elf Name: " +
elf.getCharDetail().getName());
  System.out.println("Cloned Elf Name: " +
clonedElf.getCharDetail().getName() + "\n");
  }
}
```

Çıktıda Character nesnesinin de her bir elf nesnesi için farklı bir referans olacak şekilde tutulduğu görülmektedir:

```
Checking primitive data types:
Original Elf Language: Common Speech
Cloned Elf Language: New Elven Language

Checking objects:
Original Elf Name: Aen Elle
Cloned Elf Name: New Elves
```