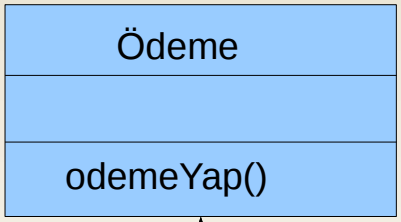
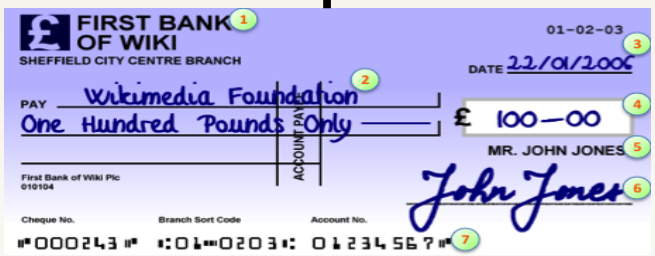
# Proxy Örüntüsü

# Proxy

- Bazı durumlarda **nesnelerin hemen yaratılması maliyetli** olabilir, yada o anda yaratılmaları **uygun olmayabilir**
- Ya da bazı nesnelere erişmeden evvel veya erişimden sonra **ilave bazı işlemler**in yapılması gerekebilir
- Asıl nesnenin yaratılmasını ihtiyaç anına kadar erteleyen, asıl nesneden önce veya sonra devreye giren, **asıl nesne yerine kullanılabilen bir nesne** yaratılır
- Vekil nesne asıl nesneye **erişimi dolaylı** hale getirir

# Proxy



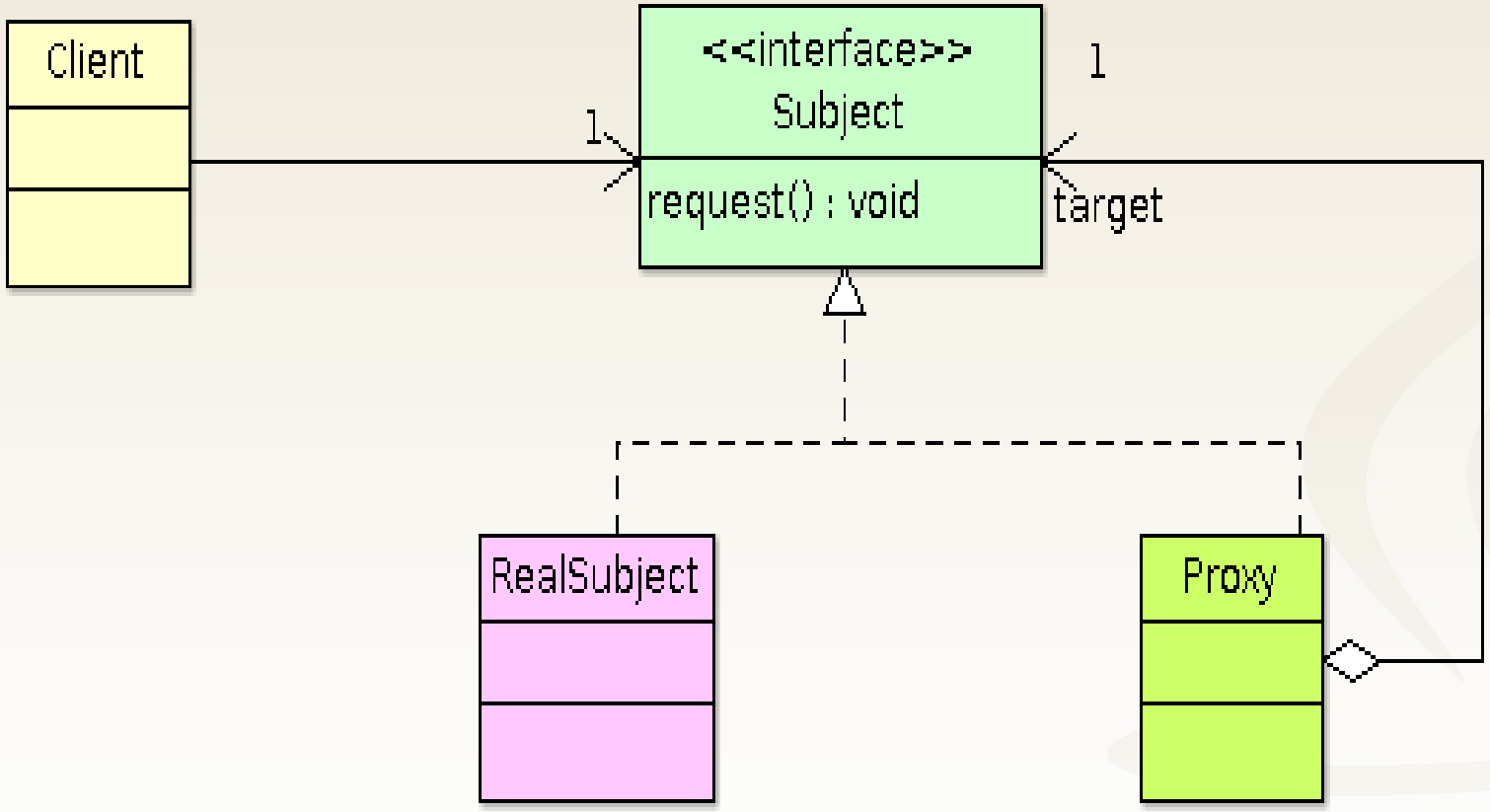




Asıl nesneye delegasyon


Çek: asıl nesneye ihtiyaç Duyulana kadar ödeme işlemlerinde
onun yerini alan Asıl nesne ile bir tutulan proxy nesne

Nakit para: Asıl nesne

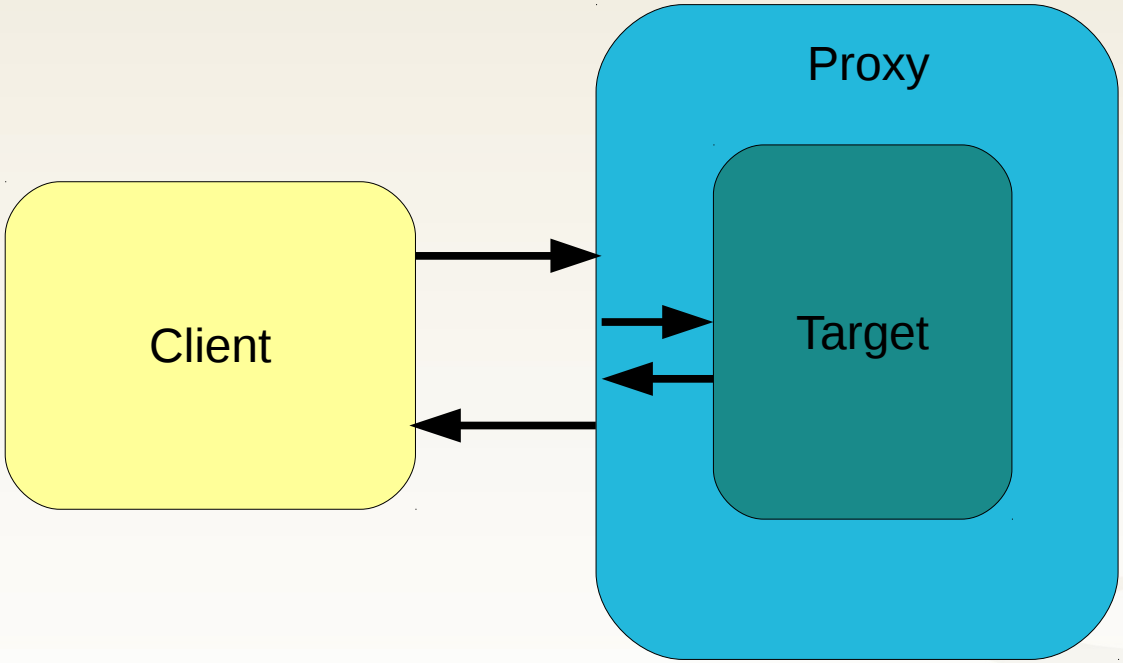
# Proxy Sınıf Diagramı



# Proxy

Proxy, target nesne ile aynı tipte olup, client ile target nesnenin arasına girer

Client proxy nesne ile konuştuğunun farkında değildir




Client'ın target nesne üzerindeki metot çağrıları öncelikle proxy nesneye erişir

Proxy metot çağrısından önce veya sonra bir takım işlemler gerçekleştirebilir

# Proxy Tipleri

- **Remote Proxy:** Farklı bir adres space'indeki nesnenin lokal represantasyonunu oluşturur
- **Virtual Proxy:** Yaratılması maliyetli nesneyi gerçekten ihtiyaç duyulduğunda yaratır
- **Protection Proxy:** Asıl nesneye erişimi denetler
- **Smart Reference:** Asıl nesneye erişim sağlamanın yanı sıra ilave işlemlerde gerçekleştirir

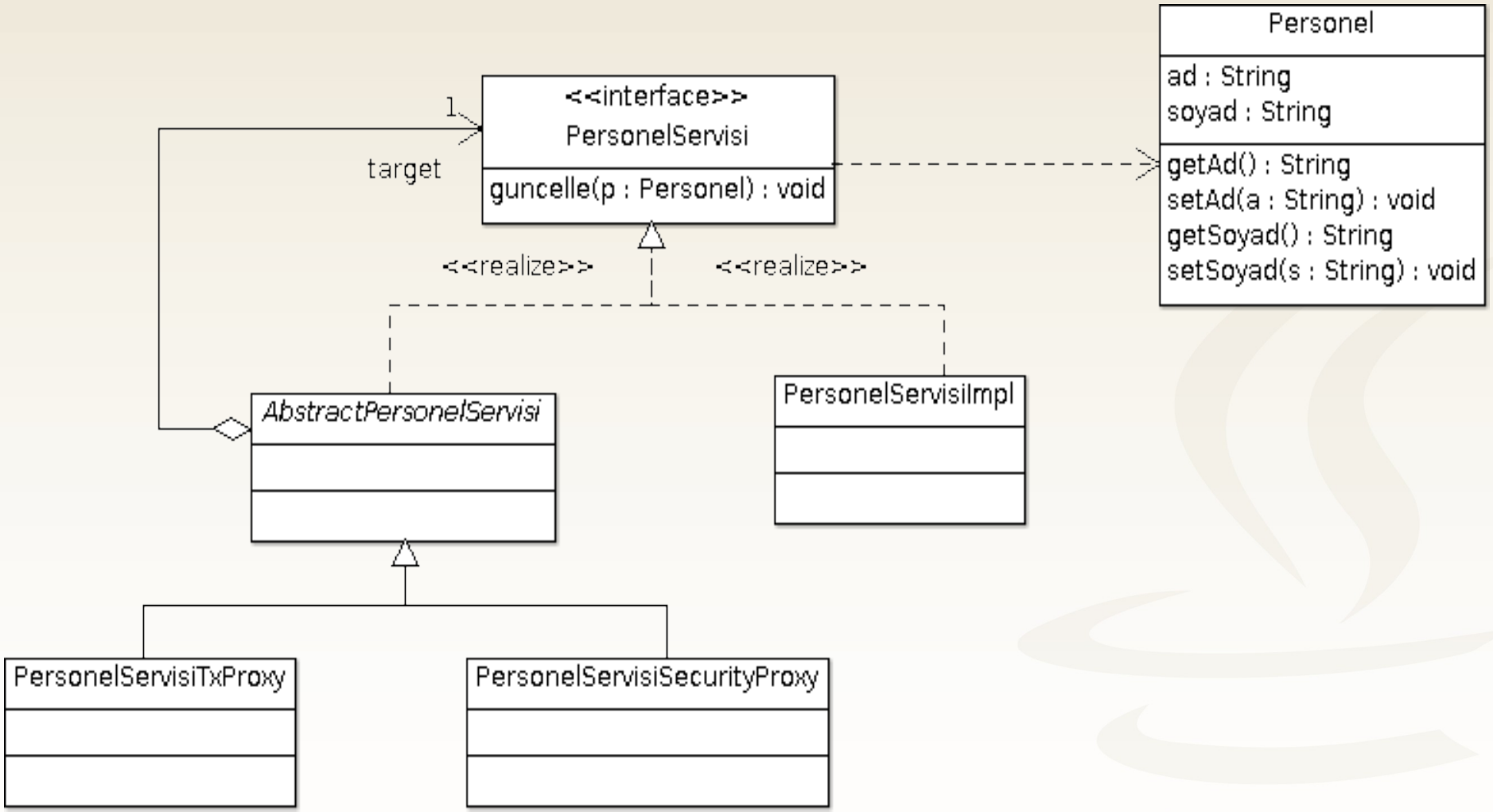
# Dinamik Proxy Oluşturma Yöntemleri

- **Interface Proxy**
  - Proxy sınıf üretmek için hedef nesnenin sahip olduğu arayüzlerden birisi kullanılır
  - JDK proxy olarak da bilinir, JDK API'sinde mevcuttur
- **Class Proxy**
  - Proxy sınıf hedef nesnenin ait olduğu sınıf extend edilerek yaratılır
  - CGLIB proxy olarak da bilinir, CGLIB, Javassist gibi kütüphaneler kullanılarak gerçekleştirilir

# LAB ÇALIŞMASI: Proxy

- Personel bilgilerini güncelleyen PersonelServisi isimli bir sınıf vardır
- Bu sınıf içerisinde personel güncellemesi yapılırken TX yönetiminin de yapılması istenmektedir
- Ayrıca personelin sadece kendi bilgilerini güncellemesi için de yetki kontrolü yapılmalıdır
- Transaction yönetimi ve yetkilendirme işlemlerinin istemci kodu tarafından bilinmesi istenmemektedir
- Bu davranışlar personel güncelleme davranışı üzerine sonradan konfigüratif ve uygulama geliştiricilerin isteğine bağılı biçimde eklenebilmelidir

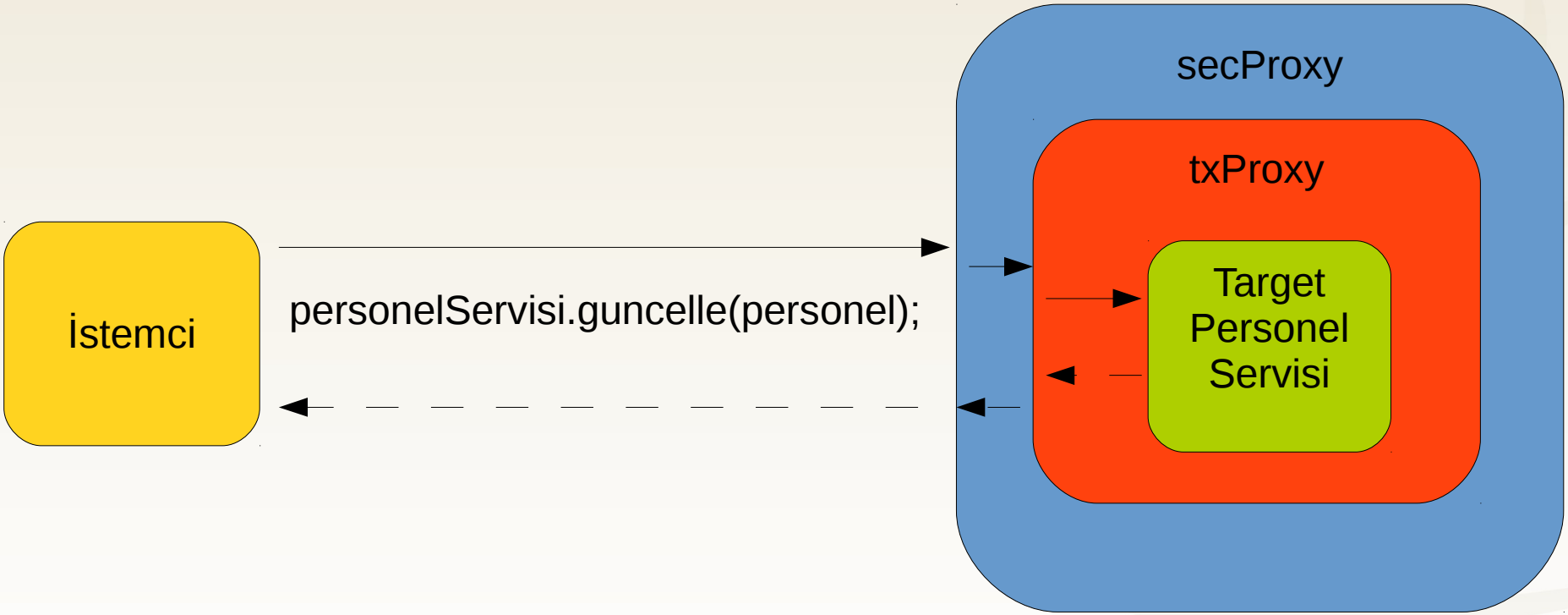
# LAB ÇALIŞMASI: Proxy



# Java ve Proxy

- JDK API'sinde arayüz tabanlı **dinamik proxy sınıf üretme kabiliyeti** mevcuttur
- Hedef nesnenin sahip olduğu **arayüz veya arayüzler** dinamik olarak üretilen bir **proxy sınıf tarafından implement** edilir
- Proxy sınıf çalışma zamanında **JDK Proxy API**'si tarafından üretilir
- Bu proxy sınıftan bir nesne de yine JDK Proxy API'Si üzerinden elde edilir

# Java ve Proxy



# Java ve Proxy

Java.lang.reflect paketindeki InvocationHandler arayüzü implement edilerek, proxy nesne içerisinde yürütülecek olan işlem invoke metodu içerisinde kodlanır

```
public class TxInvocationHandler implements InvocationHandler {

    private Object target;

    public TxInvocationHandler(Object target) {
        this.target = target;
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        try {
            System.out.println("tx begin");
            Object result = method.invoke(target, args);
            System.out.println("tx commit");
            return result;
        } catch(Exception ex) {
            System.out.println("tx rollback");
            throw new RuntimeException(ex);
        }
    }
}
```

# Java ve Proxy

```
public class SecInvocationHandler implements InvocationHandler {

    private Object target;

    public SecInvocationHandler(Object target) {
        this.target = target;
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        System.out.println("security check");
        Object result = method.invoke(target, args);
        return result;
    }
}
```

# Java ve Proxy

```
Class<?>[] interfaces = new Class[] { PersonelServisi.class };

ClassLoader classLoader = PersonelServisi.class.getClassLoader();
```

Proxy sınıfın implement edeceği arayüz veya arayüzler, proxy sınıfı yükleyecek ClassLoader belirlenir. Bu ClassLoader genellikle arayüzleri yükleyen sınıf olabilir.

```
PersonelServisi target = new PersonelServisiImpl();
```

Asıl işin delege edileceği hedef nesne yaratılır

```
InvocationHandler txInvocationHandler = new TxInvocationHandler(target);

Object txProxy = Proxy.newProxyInstance(classLoader, interfaces,txInvocationHandler);
```

Belirlenen arayüz(ler), classLoader ve proxy nesnenin içerisinde yürütülecek olan kod (invocationHandler) ile java.lang.reflect paketindeki Proxy sınıfı kullanılarak Proxy instance yaratılır. Bu işlem sırasında proxy sınıf da dinamik olarak yaratılmaktadır.

# Java ve Proxy

```
InvocationHandler secInvocationHandler = new SecInvocationHandler(txProxy);

Object secProxy = Proxy.newProxyInstance(classLoader, interfaces,secInvocationHandler);
```

İstenilen derinlikte proxy zinciri oluşturulabilir.

```
PersonelServisi personelServisi = (PersonelServisi) secProxy;

personelServisi.guncelle(new Personel());
```

Zincirde oluşturulan en son proxy nesne hangi arayüz ile çalışılacak ise ona downcast edilerek kullanılabilir.

# Proxy Örüntüsünün Sonuçları

JAVA Eğitimleri

- İlave kabiliyetlerin veya her durumda işletilmesi uygun olmayan **davranışların tek bir sınıf içerisinde birikmesi**nin önüne geçilir
- İlave davranışlar farklı tipte nesnelerle beraber de kullanılabilir
- Böylece bu **davranışların yeniden kullanılabilirliği** mümkün hale gelir

# İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri