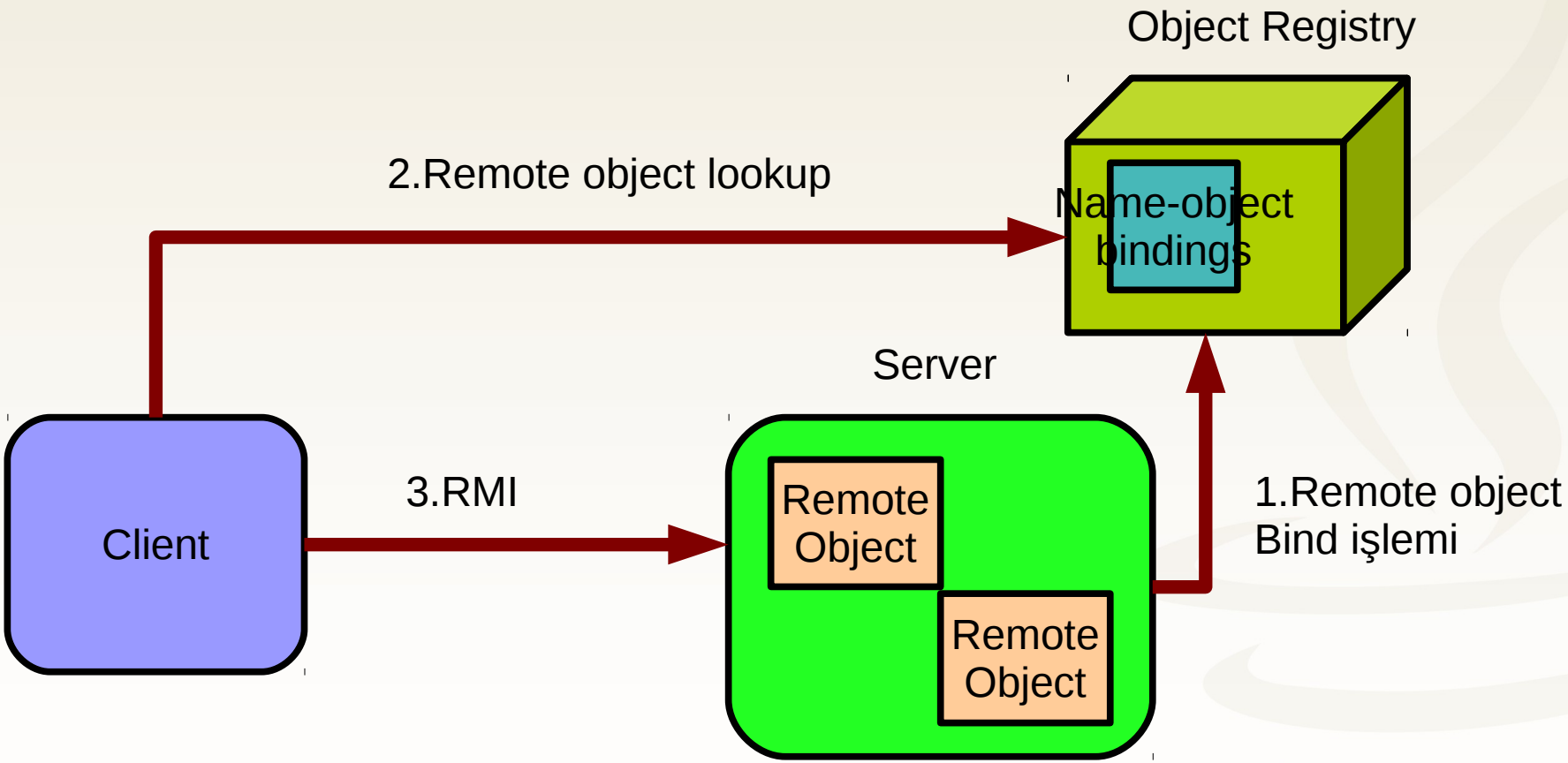
Remote Method Invocation (RMI)

harezmi bilişim çözümleri

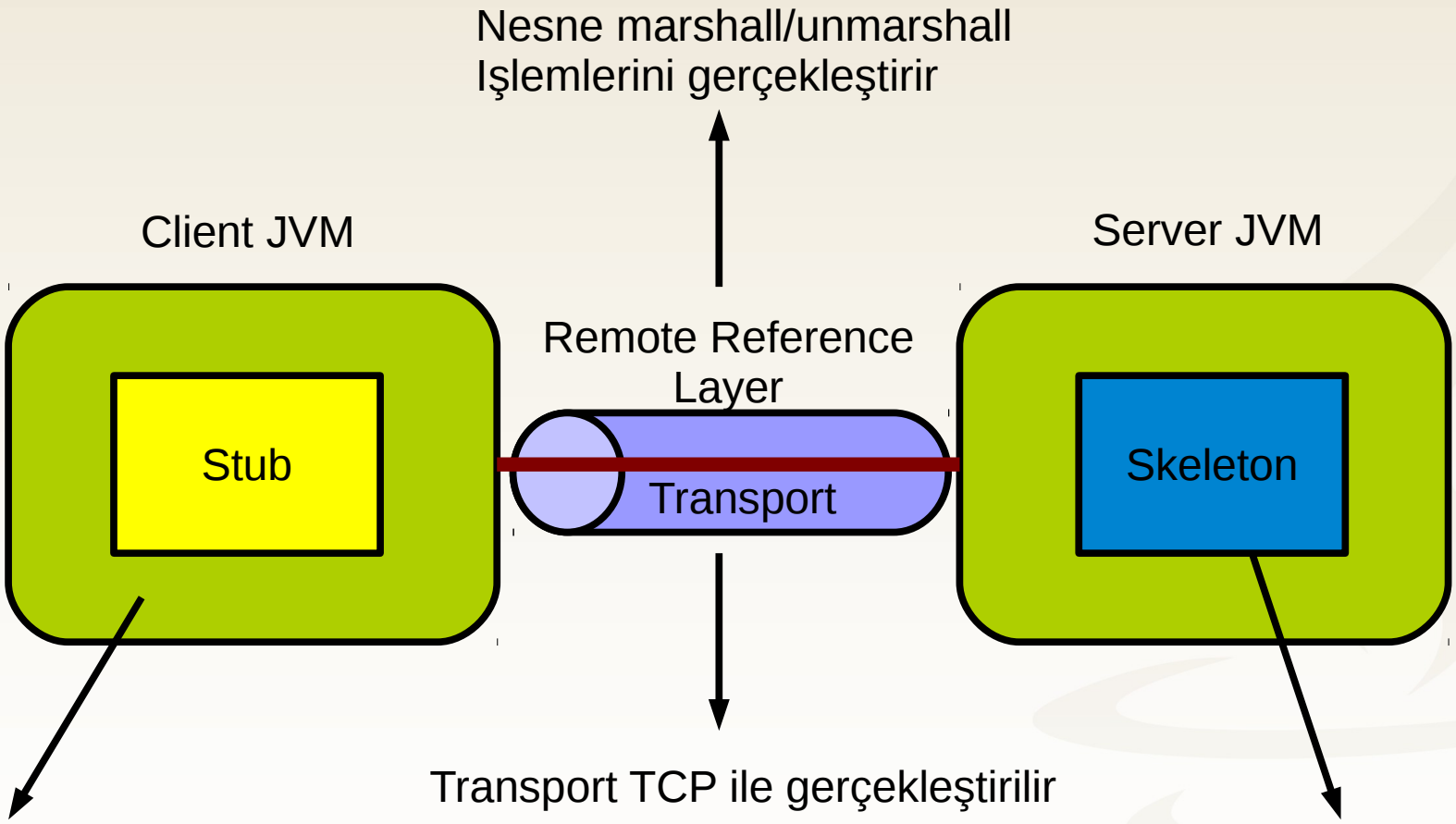
RMI Nedir?

- **Farklı bir JVM'de çalışan** Java nesnesinin metotlarına erişim imkanı sunar
- Java uygulamalarının arasında **uzaktan iletişim** kurmalarını sağlar
- Object oriented **remote procedure call** (RPC) mekanizması olarak tanımlanabilir
- RMI sayesinde server'daki nesne, aynı arayüz ile **sanki client'ın JVM'inde çalışıyormuş gibi** kullanılabilir
- Remote nesne **RMI işleminden habersiz**dir

RMI Mimarisi



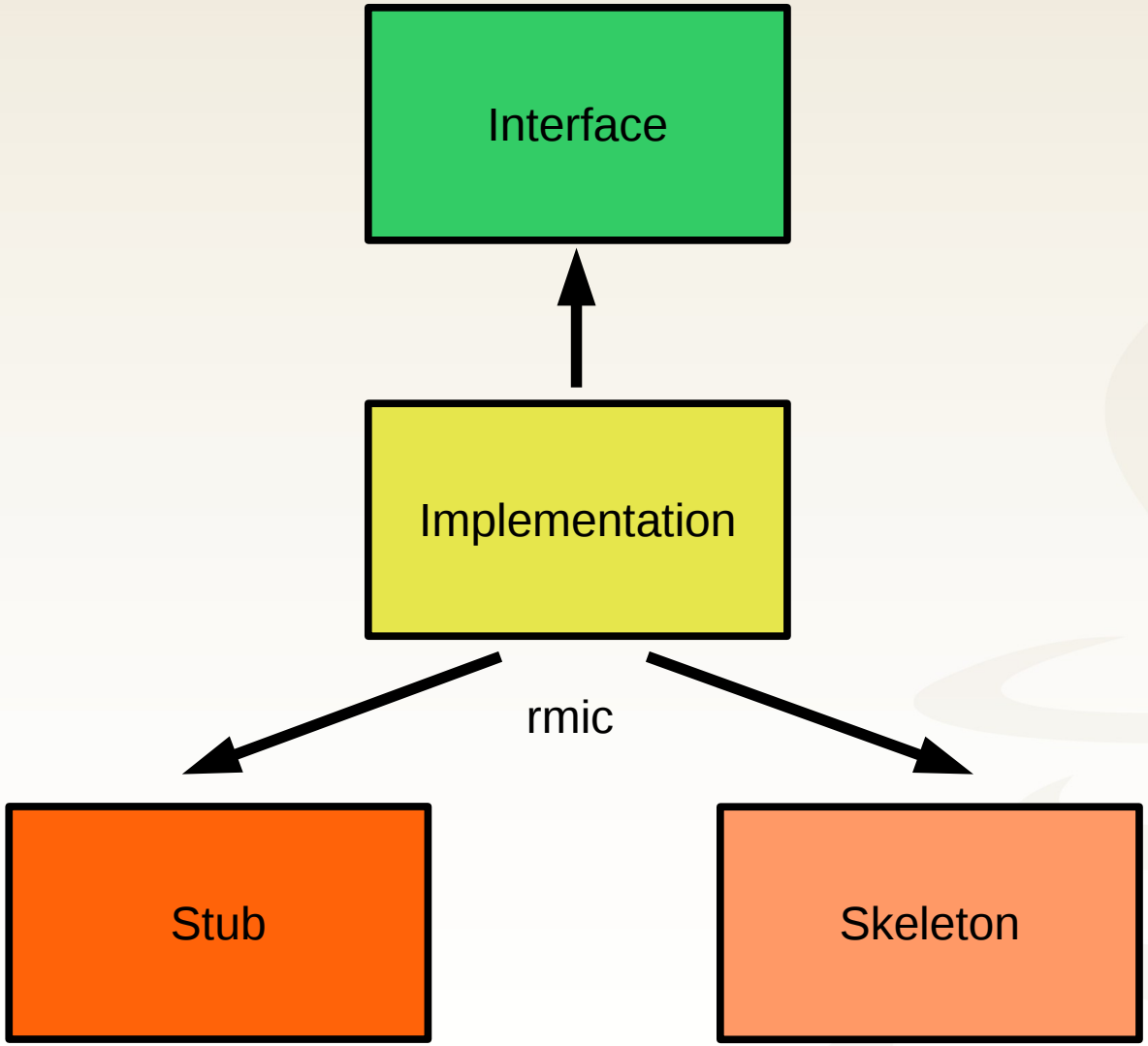
Stub ve Skeleton Nesneler



Stub, client tarafından object registry'deki
Remote object'e lookup işlemini gerçekleştirir

Skeleton, server tarafındaki asıl
Nesnenin object registry içerisine
Bind edilmesi işlemlerini gerçekleştirir
Asıl nesneye erişim sağlar

Stub ve Skeleton Oluşturulması



Stub ve Skeleton Oluşturulması

- Java **1.2**'den itibaren **Skeleton** oluşturulmasına gerek yoktur
- Java **1.5**'den itibaren **Stub** oluşturulmasına gerek yoktur
- Java **dinamik proxy yöntemi** ile runtime'da stub üretmektedir

RMI ve Sınıflar

- **Remote**
  - Client ve instance **aynı veya farklı address space**'lerde bulunabilir
  - Farklı address space'de iken remote nesneye erişim "**object handle**" vasıtası ile olur
  - Eğer remote nesne RMI metoduna input param veya return değer ise **"object handle" kopyalanır**
- **Serializable**
  - **Instance değeri** bir address space'den başka bir address space'e kopyalanır
  - Eğer serializable nesne, RMI metoduna input param veya return değer ise serialization ile **değeri kopyalanır**

RMI Örneği – Server Tarafı

```
public interface Hello extends Remote {
    public String hello(String name) throws RemoteException;
}
```

RMI ile erişilebilecek metotlar bir interface ile tanımlanır, bu sayede client tarafında sadece bu interface'in olması yeterlidir

```
public class HelloImpl extends UnicastRemoteObject implements
Hello {
```

UnicastRemoteObject HelloImpl sınıfının remote client'lar tarafından erişilebilir hale gelmesini sağlar. Bu sınıfı extend etmeden de sınıfın remote client'lar tarafından erişilebilir kılınması mümkündür.

```
    protected HelloImpl() throws RemoteException {
        super();
    }

    public String hello(String name) throws RemoteException {
        return "Hello " + name;
    }
}
```

RMI Örneği – Server Tarafı

```
Registry registry = LocateRegistry.createRegistry(1198);
```

Öncelikle RMI registry elde edilmelidir, yeni bir registry yaratılabilir veya mevcut bir registry kullanılabilir

```
HelloImpl hello = new HelloImpl();
```
→ Daha sonra remote instance oluşturulur

```
//UnicastRemoteObject.exportObject(hello, 1198);
```

Eğer remote instance UnicastRemoteObject sınıfını extend etmez ise bu şekilde export edilmesi gerekir

```
Naming.bind("//localhost:1198/hello", hello);

//registry.bind("hello", hello);
```

Son olarak da remote Instance'ın belirli bir isimde Registry'ye bind edilmesi sağlanır

RMI Örneği – Client Tarafı

```
Registry registry = LocateRegistry.getRegistry("localhost", 1198);
```

Öncelikle RMI registry'nin elde edilmesi gerekir

```
Hello remoteObject = (Hello) registry.lookup("hello");
```

Ardından remote nesneye ismi ile lookup yapılır

```
System.out.println(remoteObject.hello("kenan"));
```

RMI Örneği

- **UnicastRemoteObject.exportObject(Remote)** metodu kullanılırsa **rmic ile Stub sınıfının oluşturulması** gerekir
- Çünkü üretilen proxy bu metot ile dönen **RemoteStub** sınıfına cast edilemez

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri