Factory

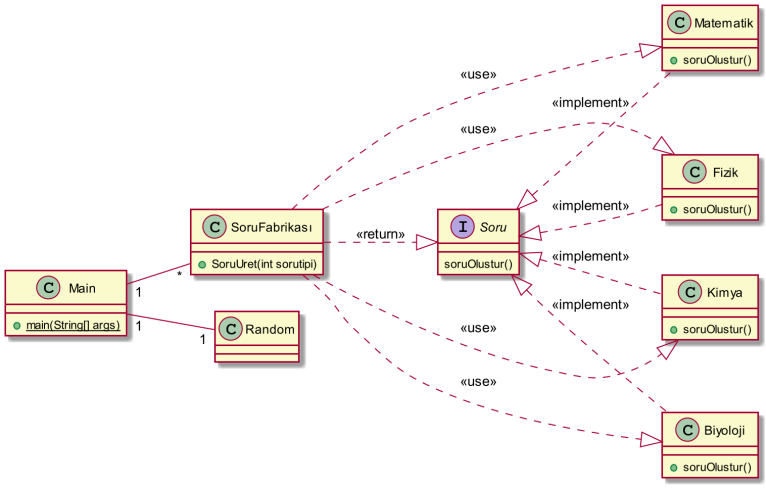

Figure 1. Factory

1. Factory

```
Factory tasarım kalıbı aynı (abstract yada interface) superclasstan
 implement(extend) edilmiş bir çok sınıfın bir factory sınıfındaki
 public methoda gönderilen parametre aracılığıyla hangi tipte geri
 dönüş alacağının seçilebilmesi işlemidir. Kod karmaşıklığını ve
 anlaşılırlığını yönetmek açısından avantajlar sağlar
```

```
public interface Soru { ❶

    void soruOlustur(); ❷
}
```

❶ Soru isminde bir interface oluşturuluyor

❷ Kalıtım sağlanan sınıflarda override edilmek üzere fonksiyon belirtiliyor.

```
public class Fizik implements Soru { ❶

    @Override
    public void soruOlustur() { ❷
        System.out.println("Fizik sorusu oluşturuldu.");
```

```
    }
}
```

❶ İnterface sınıf implement edilmiştir.

❷ İnterfacede belirtilen fonksiyonlar override edilmiştir.

```
public class SoruFabrikası {

    public Soru SoruUret(int sorutipi){    ❶

        if (sorutipi == 1)
            new Matematik().soruOlustur();  ❷
        else if (sorutipi == 2)
            new Fizik().soruOlustur();
        else if (sorutipi == 3)
            new Kimya().soruOlustur();
        else if (sorutipi == 4)
            new Biyoloji().soruOlustur();

        return null;
    }
}
```

❶ Aldığı parametre doğrultusunda soru tipinde geri dönüş yapan public bir fonksiyon.

❷ Girilen parametreye göre belirli tiplerde nesne dönderir.

```
public class Main {

    public static void main(String[] args) {

        SoruFabrikası sorufabrikası=new SoruFabrikası();  ❶
        Random rand = new Random(); ❷

        for (int i=0;i<10;i++){
            sorufabrikası.SoruUret(rand.nextInt(3)+1); ❸
        }

    }
}
```

❶ SoruFabrikası sınıfından bir nesne üretir.

❷ Random sınıfından rand isminde bir nesne üretir.

❸ 10 adet rastgele seçilen bir tipte soru getirir.