```
7     }
8      public static void main(String [] args) {
9         Hose h1 = new Hose();
10        Jeans j1 = new Jeans();
11        GenPaar<Hose> p1 = new GenPaar<Hose> (h1,j1);
12        System.out.println(linksGleichRechts(p1));
13        System.out.println(links(p1));
14    }
15 }
```

haben wir zwei generische Methoden definiert, denen jeweils eine Referenz auf ein Objekt der typisierten Klasse `GenPaar` als Parameter übergeben werden kann. Die Methode `linksGleichRechts` vergleicht die beiden im `GenPaar`-Objekt referenzierten Objekte und gibt einen entsprechenden Booleschen Wert zurück. Die Methode `links` besitzt auch einen typisierten Rückgabewert und gibt beim Aufruf die Referenz auf das Objekt zurück, die mit der `GenPaar`-Instanzmethode `getL` ausgelesen werden kann.
Beim Aufruf der generischen Methoden ermittelt der Compiler den tatsächlichen Typ, der für den Platzhalter `T` einzusetzen ist, aus dem Typ der Objekte, die als aktuelle Parameter übergeben werden.
Nachdem wir nun die Notation der generischen Methoden kennen gelernt haben, stellt sich die Frage, wann wir Wildcards in der Parameterliste einer Methode verwenden und wann die oben beschriebene generische Deklaration der Methoden zur Anwendung kommen sollte. Unsere oben implementierte Methode `linksGleichRechts` könnte nämlich auch folgendermaßen aussehen:

```
public static boolean linksGleichRechts (GenPaar<?> x){
  return x.getL().equals(x.getR());
}
```

Durch die nun benutzte Wildcard-Notation haben wir keinerlei Funktionalität eingebüßt. Die Methode

```
public static <T> T links (GenPaar<T> x){
  return x.getL();
}
```

lässt sich jedoch nicht durch eine alternative Version ohne Typ-Parameter ersetzen. Die Äbhängigkeit des Rückgabetyps der Methode von der Typisierung ihres Parameters lässt sich nur mit Hilfe einer generischen Methode realisieren.
Zum Schluss sei noch erwähnt, dass es laut obiger Syntax auch möglich ist, verschiedene Typ-Parameter zu verwenden und gegenseitige Abhängigkeiten zwischen den Typisierungen der Parameter selbst und dem Rückgabewert der Methode durch entsprechende Einschränkungen der Typ-Parameter vorzunehmen. Die Methode `linksPaar` in der Klasse

```
1 public class GenPaarTest5 {
2   public static <T,S extends T> GenPaar<T> linksPaar
3                                       (GenPaar<T> x, GenPaar<S> y){
4     return new GenPaar<T>(x.getL(),y.getL());
5   }
6   public static void main(String [] args) {
```