Aufgabe 4.1 Polymorphie

a) Welche Vorteile bietet die Polymorphie bei der Objektorientierten Programmierung? Erläutern Sie die Vorteile an einem eigens gewählten Beispiel in Java.

Kindklassen können ihr eigenes Verhalten genauer ausspezifizieren, als ihre Elternklassen. Bestes Beispiel ist die **toString**-Methode von Java objects, die im Basis-Fall einen Hashcode ausgibt, jedoch häufig mit dem Inhalt von Objekt Attributen befüllt wird.

b) Nachfolgender Quellcode mit Java-Generics ist nicht kompillierbar. Was ist die Ursache? Was wäre eine mögliche Lösung ohne die Generics oder die Methoden zu entfernen?

```
public class GenericFail<S,T> {
    public void doAnything(S sValue){
        System.out.println("Doing anything with S");
    }

    public void doAnything(T sValue){
        System.out.println("Doing anything with T");
    }
}
```

Die Methoden doAnything haben die gleiche Signatur, da beide alle möglichen Objekte annehmen können. Der Kompiler kann nicht entscheiden, welche Methode aufgerufen werden soll. Das Problem kann durch unterschiedliche Herangehensweisen gelöst werden. Am einfachsten jedoch dadurch, dass min. 1 der beiden Generischen Typen in der Klassendeklaration mit dem Schlüsselwort **extends** eingeschränkt wird.