



Technische Universität Berlin

Software and Embedded Systems Engineering Group

Prof. Dr. Sabine Glesner

www.sese.tu-berlin.de    Sekr. TEL 12-4    Ernst-Reuter-Platz 7    10587 Berlin







## Softwaretechnik und Programmierparadigmen WiSe 2022/2023

Prof. Dr. Sabine Glesner  
Milko Monecke  
Simon Schwan

### Übungsblatt 7

#### Aufgabe 1: Metriken

Ihr habt den Code einer Funktion in JAVA erhalten und sollt dessen Qualität überprüfen. Die Funktion `createOrder(int customerID, int productID, Instant created)` soll im System einen Auftrag erstellen. Als erstes wird geprüft, ob die Kunden- und Kundinnennummer (`customerID`) und Warennummer (`productID`) im System vorhanden sind, und ob das Erstelldatum (`created`) gültig ist. Die Kunden und Kundinnen können außerdem für bestimmte Waren Rabatte sammeln, was vom System automatisch überprüft wird. Wurden alle Daten korrekt eingegeben, wird ein Auftrag im System erstellt.

- a) Bestimmt die LOC- und NCLOC-Werte der Implementierung. Was sagen sie über die Komplexität aus? 
- b) Erstellt zu der angegebenen Implementierung den Kontrollflussgraph. Das Werfen einer Exception kann dafür wie ein `return`-Statement behandelt werden. 
- c) Zählt alle möglichen Ausführungspfade des Programmes. Was sagt diese Zahl über die Komplexität aus? 
- d) Berechnet die Zyklomatische Komplexität nach McCabe. 

#### Aufgabe 2: Fragen zur Haskell Hausaufgabe

Am Ende der Übung habt ihr Zeit Fragen zur Haskell Hausaufgabe zu klären.

```

1  public Order createOrder(int customerID, int productID,
2      Instant created) throws ShopInputException {
3
4      // handle invalid inputs
5      if(!customers.containsKey(customerID)){
6          throw new ShopInputException(
7              ShopInputException.INVALID_CUST_NR);
8      }
9      if(!products.containsKey(productID)){
10         throw new ShopInputException(
11             ShopInputException.INVALID_PROD_NR);
12     }
13     if(created == null){
14         throw new ShopInputException(
15             ShopInputException.INVALID_DATE);
16     }
17
18     // get customer and product for given IDs
19     Customer c = customers.get(customerID);
20     Product p = products.get(productID);
21     // create new order
22     Order order = new Order(customerID, productID,
23         p.getPrice(), created);
24
25     if((c.isPriorityCustomer() & p.isDiscountable())
26         || isAnniversary(created)) {
27         order.setDiscount(true);
28         while(!c.getDiscount().isEmpty()) {
29             order.discount(c.getDiscount().removeFirst());
30         }
31     }
32     return order;
33 }

```

**Lösung:**

- a) Nach Definition aus der VL sind es 33 für LOC, und 27 für NCLOC. Für sich allein sagt dies nicht viel über die Komplexität. Interessanter ist es um z.B. erstmal grob Funktionen im Code nach ihrem Umfang zu sortieren, oder Projekte über ihren Verlauf oder den Umfang ihrer Komponenten zu bewerten.
- b) Siehe Abbildung 1. Zum Erstellen kann auch das Eclipse-Tool verwendet werden, das geht allerdings unter Umständen etwas anders mit den Statements um.
- c) Das Programm hat eine nichttriviale Schleife – und wie wir oben gelernt haben, wissen wir daher also nicht, wie viele Pfade es gibt (Vielleicht unendlich, wenn sie nicht terminiert). Das ist natürlich schade, denn diese Zahl wäre wirklich sehr interessant für verschiedene weitere Berechnungen wie z.B. die Worst-Case Execution Time (WCET).
- d) McCabe:  $\#edges - \#nodes + 2 = 21 - 16 + 2 = \mathbf{7}$  ( $\#conditions + 1 = 6 + 1 = 7$ )

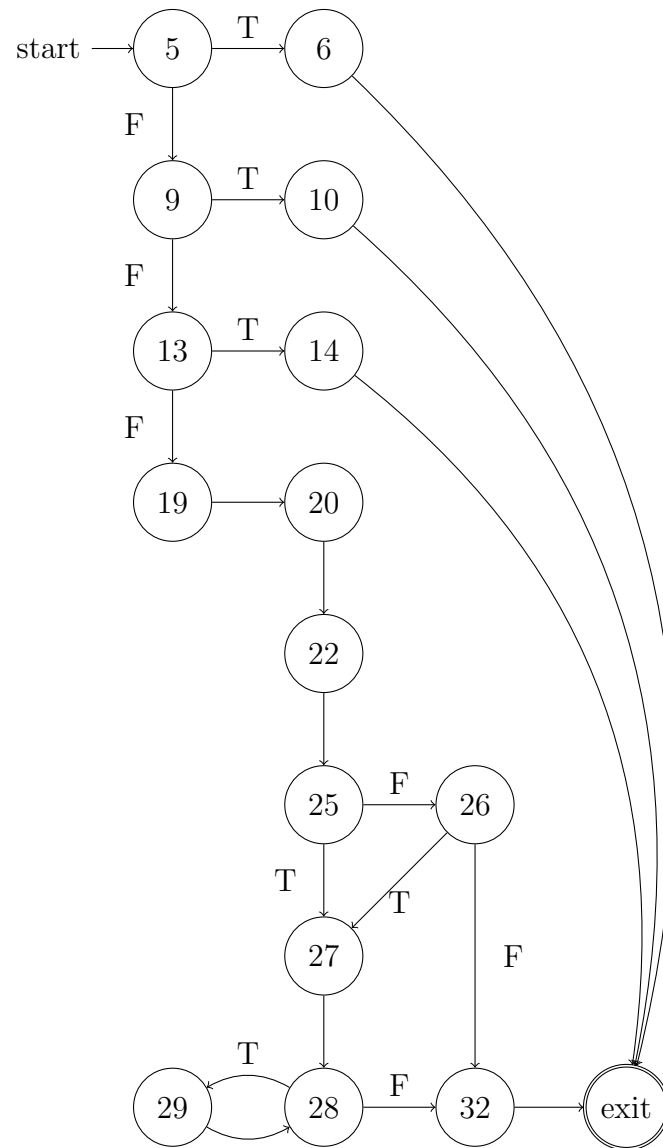


Abbildung 1: Kontrollflussgraph für `createOrder`.