Ein weiteres Beispiel für Objektinteraktion

Es geht immer noch darum, wie Objekte andere Objekte erzeugen und wie diese Objekte gegenseitig ihre Methoden aufrufen.

Bei der „Digitaluhr“ habt ihr selbst den Quelltext erstellt – im nächsten Beispiel sollt ihr mit Hilfe des Debuggers ein fertiges Programm aus mehreren Klassen verstehen. Der Debugger bietet die Möglichkeit, das Programm zeilenweise auszuführen und zeigt dabei die Werte der Variablen an.

Ein **Debugger** ist ein Softwarewerkzeug, mit dessen Hilfe die Ausführung eines Programms untersucht werden kann. Es kann benutzt werden, um Fehler (Bugs) zu finden.

Übung 1

Kopiere das Projekt Mail-System in dein Verzeichnis und öffne es. Erzeuge ein Objekt der Klasse MailServer. Erzeuge dann drei Objekte der Klasse MailClient. Dabei musst du den Namen der MailServer-Instanz, die du gerade erzeugt hast, als Parameter übergeben.

Du musst auch für jeden MailClient einen Benutzernamen angeben (z.B. Pippi, Annika, Tommi; jede MailClient-Instanz soll hier das Email-Programm eines Benutzers darstellen).

- Notiere, wie du Nachrichten von Pippi an Annika, Tommi an Pippi, usw. erzeugst und wie du die Nachrichten mit der jeweiligen MailClient-Instanz abrufst.
- Zeichne ein Klassendiagramm dieser Klasse und ein Objektdiagramm von der Situation unmittelbar nach der Erzeugung eines MailServers und dreier MailClients. Ergänze das Objektdiagramm so, dass es den Zustand nach dem Erzeugen einer Nachricht von Annika an Tommi zeigt.

Aufgabe 2

Öffne den Quelltext der Klasse Nachricht und notiere die Bedeutung des Schlüsselwortes „this“ im Konstruktor dieser Klasse.

Übung 3

Erzeuge ein neues MailSystem: ein MailServer und zwei MailClients, Harry und Sally. Lass Sally eine Nachricht an Harry senden – rufe die Nachricht von Harrys Client aber noch NICHT ab! Öffne nun den Quelltext der Klasse MailClient und setze einen Haltepunkt in der ersten Zeile der Methode, indem du auf Höhe der gewünschten Zeile auf den linken Randbereich klickst (da stehen in der Regel die Zeilennummern).

Rufe jetzt die Methode naechsteNachrichtAusgeben im MailClient von Harry auf. Klicke dann auf die Schaltfläche „Schritt“ im Debugger-Fenster. Notiere, was im Debugger zu beobachten ist! Begründe, auf welche Zeile die Markierung springen wird, wenn du nocheinmal auf „Schritt“ klickst.

Übung 4
Wiederhole Übung 3, klicke aber auf „Schritt hinein“, wenn du die Zeile nachricht.ausgeben(); erreichst. Notiere deine Beobachtung im Debugger.

Übung 5
Setze jetzt einen Haltepunkt in die erste Zeile der Methode sendeNachricht der Klasse MailClient. Rufe dann diese Methode auf und nutze „Schritt hinein“, um in den Konstruktor der Klasse Nachricht zu schreiten. Beobachte durch mehrfaches klicken auf „Schritt“, wie die Instanzvariablen initialisiert werden.

Übung 6
Benutze eine Kombination aus dem Lesen der Quelltexte, dem Ausführen von Methoden und der Verwendung des Debuggers um dich mit den Klassen Nachricht und MailClient vertraut zu machen (die Klasse MailServer lassen wir außen vor, da die hier verwendeten Java-Konstrukte noch nicht erklärt werden sollen – wer sich selbst einarbeiten möchte, kann das natürlich tun).
Erkläre schriftlich, wie die Klassen MailClient und Nachricht miteinander interagieren. Zeichne Objektdiagramme als Teil deiner Erklärung.

Übung 7
Füge eine Betreffzeile für eine Email in eine Nachricht aus dem Projekt Mail-System ein. Diese Betreffzeile soll auch auf die Konsole ausgegeben werden: passe dafür die Klasse MailClient entsprechend an.

Aufgabe 8
Welche Methoden der Klasse Nachricht und welche der Klasse MailServer werden in der Klasse MailClient aufgerufen? Beschreibe allgemein, wie Methoden anderer Klassen aufgerufen werden (externer Methodenaufruf).
Wie müsstest du den Aufruf der Methode ausgeben() der Klasse Nachricht im Konstruktor der Klasse Nachricht notieren (interner Methodenaufruf)?