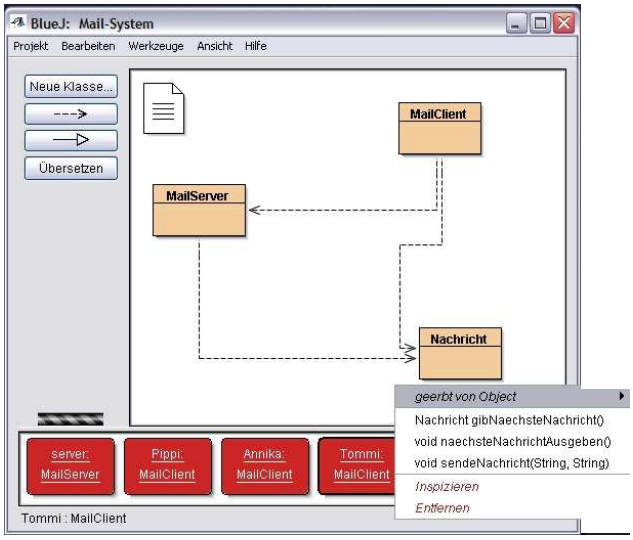
Ein weiteres Beispiel für Objektinteraktion



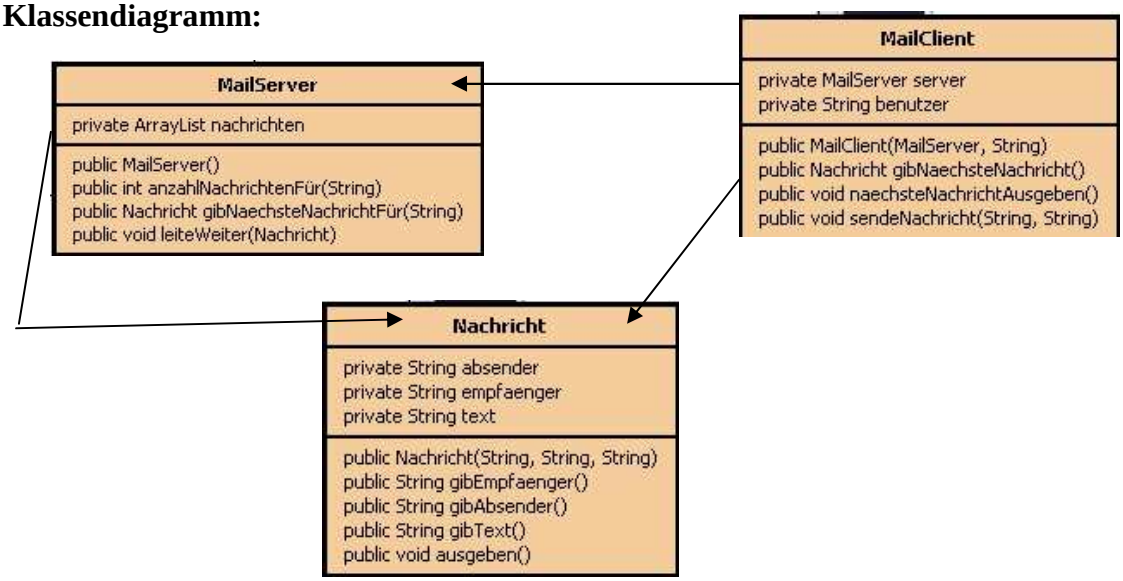
Ausgangssituation: Drei Klassen, für Objekte vom Typ MailServer, MailClient und Nachricht.
Vier Objekte: ein Mailserver („server“) und drei MailClients („Pippi“, „Annika“ und „Tommi“).

Um Nachrichten zu erzeugen, muss die Methode „sendeNachricht(String, String)“ des sendenden MailClient-Objekts aufgerufen werden. Der erste String-Parameter steht dabei für den Namen des Empfängers, der zweite für den Text der Nachricht.

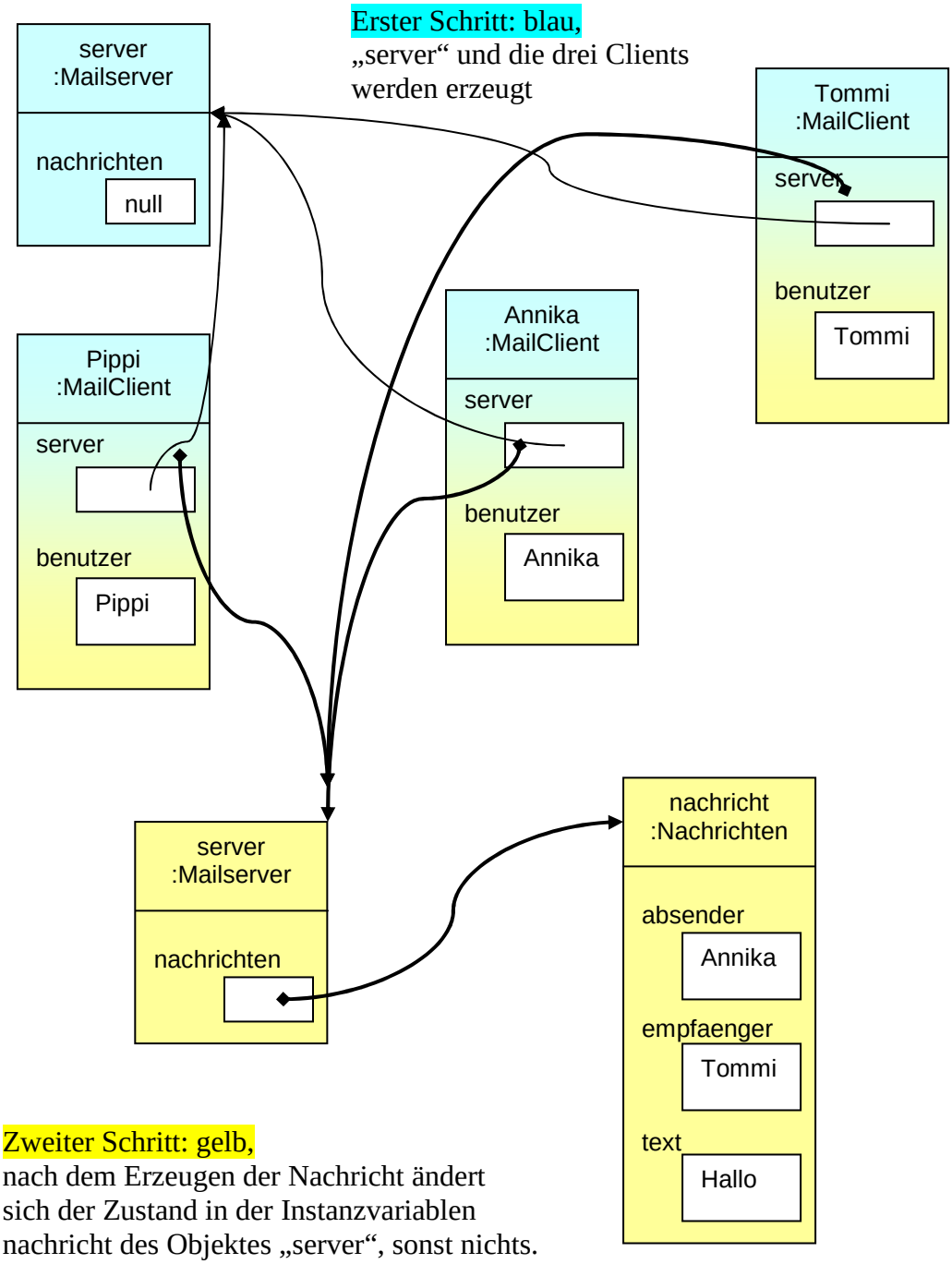
Diese Methode erzeugt ein neues Nachrichtenobjekt mit den Namen „nachricht“ und trägt die Referenz auf das Objekt in eine Liste des Servers ein.

Um Nachrichten abzurufen, hat der Client zwei Methoden. Mit „gibNaechsteNachricht()“ des Empfänger-Objekts wird dann die Referenz auf das Nachricht-Objekt vom Server abgefragt und zurückgegeben.
Als zweite Möglichkeit gibt es noch die Methode „naechsteNachrichtAusgeben()“. Hier wird zunächst auch die Referenz auf das gesuchte Nachrichtenobjekt vom Server abgefragt, dann jedoch der Inhalt der Instanzvariablen des Nachricht-Objektes auf die Konsole ausgegeben. Bei beiden Methoden wird immer zunächst die älteste Nachricht für den aufrufenden Benutzer ausgegeben, die Referenz darauf wird dann im server gelöscht, die Nachricht ist also kein zweites Mal abrufbar. Gleichzeitig wird der Zähler der gespeicherten Nachrichten reduziert.

Klassendiagramm:



Objektdiagramm:



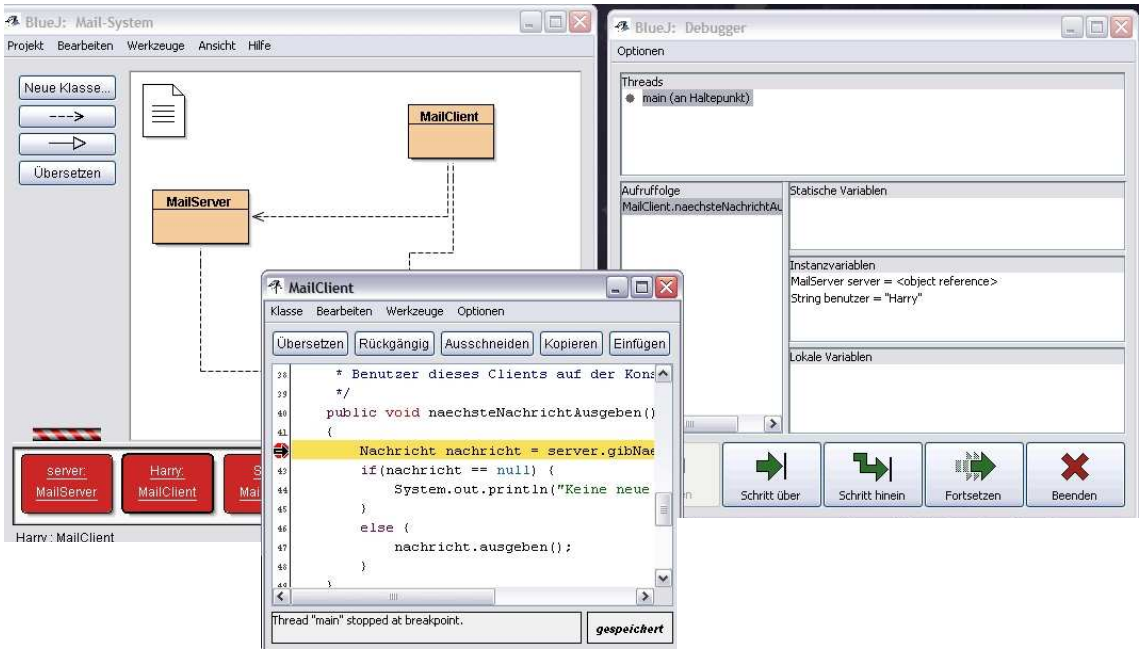
Erster Schritt: blau,
„server“ und die drei Clients werden erzeugt

Zweiter Schritt: gelb,
nach dem Erzeugen der Nachricht ändert sich der Zustand in der Instanzvariablen nachricht des Objektes „server“, sonst nichts. (Das Objekt wurde zur Verdeutlichung ein zweites Mal gezeichnet, existiert aber nur einmal.)

Aufgabe 2 Mit „`this`“ kann man verdeutlichen, welche von zwei gleichlautenden Variablen in einer Klasse gemeint ist. Im Beispiel haben die Instanzvariablen denselben Namen wie die Parameter des Konstruktors. Mit `this.name` wird auf die Instanzvariable verwiesen, `name` verweist auf den Parameter.

```
public MailClient(MailServer server, String benutzer)
    {
        this.server = server;
        this.benutzer = benutzer;
    }
```

Übung 3 - 5: Debugger



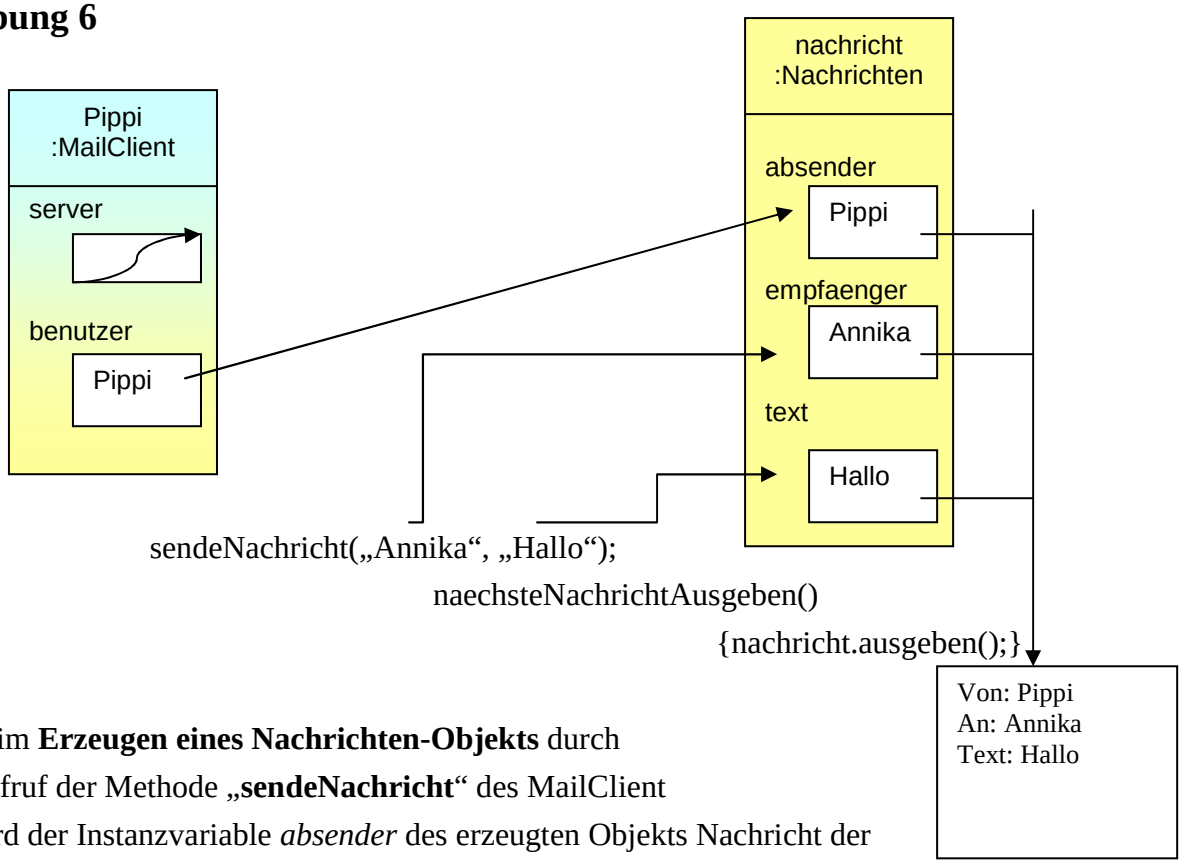
3) Wird jetzt auf „Schritt über“ geclickt, wird in der Nachrichten-Liste des Server-Objekts nach Nachrichten-Referenzen gesucht, die als Empfänger „Harry“ eingetragen haben. Diese Referenz wird an die lokal erzeugte Nachrichten-Variable übergeben (hier wird aber kein neues Objekt erzeugt), die nun unter „Lokale Variablen“ im Debugger angezeigt wird. Beim nächsten „Schritt über“ wird das Programm zu „nachricht.ausgeben()“ springen, da eine Nachricht für Harry vorliegt.

4) Klickt man jetzt auf „Schritt hinein“, so öffnet sich der Quelltext der Klasse Nachricht, und als nächstes auszuführende Zeile (hier: Text mit Absenderangabe auf die Konsole ausgeben).
Im Debugger werden nun die Instanzvariablen mit ihren Zuständen des konkreten Nachrichten-Objekts angezeigt. Ist die Nachricht vollständig ausgegeben, springt das Programm zurück in den Quelltext der Klasse „MailClient“, auch im Debugger werden wieder die Werte der Instanzvariablen des MailClient-Objekts dargestellt.

5) Bei Aufruf des Konstruktors von Nachricht sind die Instanzvariablen noch auf „null“, nur die lokalen Variablen (die Parameter) sind mit den erzeugenden Werten belegt. Durch die drei Zeilen this.name = name; wird nun der jeweiligen Instanzvariable der Wert des entsprechenden Parameters zugewiesen.

Hinweis: Wenn die Banderole in BlueJ rot-grau ist, läuft das untersuchte Programm, wenn die Banderole hellgrau-dunkelgrau ist, ist dieses Programm nicht aktiv.
Schließt man den Debugger, so entspricht das für die Objekte einer neuen Übersetzung – sie werden also aus dem Arbeitsspeicher gelöscht.

Übung 6



Beim **Erzeugen eines Nachrichten-Objekts** durch Aufruf der Methode „**sendeNachricht**“ des MailClient wird der Instanzvariable *absender* des erzeugten Objekts Nachricht der Wert der Instanzvariablen *benutzer* des MailClient-Objektes zugewiesen. Der zu übergebende Inhalt der Nachricht-Instanzvariablen *empfaenger* und *text* werden bei Aufruf der Methode „sendeNachricht“ als Parameter abefragt. Diese Inhalte sind im MailClient nur solange verfügbar, wie die Methode ausgeführt wird, da sie nicht zusätzlich an Instanzvariablen des MailClient übergeben werden.

Die Methode „**naechsteNachrichtAusgeben**“ fragt zunächst, wie die Methode „gibNaechsteNachricht“ beim server nach Objektreferenzen. Die Methode „gibNaechsteNachricht“ ist an dieser Stelle beendet. Die Methode „naechsteNachrichtAusgeben“ prüft jetzt jedoch, ob eine Referenz zurückgegeben wurde, wenn ja, wird die Methode **ausgeben** der Klasse Nachricht aufgerufen, um die Inahlte der Instanzvariablen des referenzierten Nachricht-Objektes auf die Konsole auszugeben. Ist diese Methode beendet, besteht kein Zugriff mehr auf das Nachrichten-Objekt, da die Referenz vom server an eine lokale Nachrichten-Variable in dieser Methode übergeben wurde (und auf dem server gelöscht wurde).

Aufgabe 8

Der MailClient ruft die Server-Methoden *gibNaechsteNachrichtFür(benutzer)* und *leiteWeiter(nachricht)* auf, vom Nachricht-Objekt wird die Methode ausgeben() und der Konstruktor aufgerufen. Diese externen Aufrufe folgen immer der Syntax ***instanzname.methodenname()***. Bei internen Aufrufen wird nur der Methodenname verwendet.