DHBW Karlsruhe, Vorlesung Programmieren I+II, Klausur

09.10.2014

Bearbeitungszeit: 120 Minuten

Aufgabe

Schreiben Sie eine Java-Anwendung `QUp`, die ein Warteschlangensystem (z.B. für Behörden) implementiert!

Grundidee ist es, dass Kunden beim Betreten des Wartebereichs eine Nummer ziehen. Sachbearbeiter können von ihrem Arbeitsplatz aus mittels einer Anwendung („DeskApp") weitere Kunden aufrufen („Call").
Anzeigetafeln im Wartebereich signalisieren den Kunden (anhand ihrer Nummer), dass sie an der Reihe sind und wohin sie sich wenden sollen.
Wird ein Kunde vom Sachbearbeiter aufgerufen, so wird er (bzw. seine Nummer) aus dem Warteschlangensystem gelöscht.

Teilaufgabe a) [10%]

Erstellen Sie zunächst ein Interface `QueueObserver` und die Klasse `WaitingQueue`:

Das Interface **`QueueObserver`** definiert eine Methode

```
public void updateView()
```

Die Klasse **`WaitingQueue`** soll eine Methode besitzen, die eine neue, ganzzahlige und einmalige Nummer erzeugt und zurückgibt. Die vergebenen Nummern sollen intern so verwaltet werden, dass der Kunde mit der „ältesten" Nummer zuerst zum Aufruf kommt.
[*Hinweis: Kunden werden nur anhand ihrer Nummer identifiziert.*]

Über die Methode

```
public void addQueueObserver(QueueObserver ob)
```

sollen sich beliebig viele Objekte, die das Interface `QueueObserver` implementieren, registrieren können. Beim Aufruf der Methode

```
public void updateObservers()
```

wird für alle so registrierten Objekte jeweils deren `updateView()`-Methode aufgerufen, z.B. wenn sich etwas an den Daten der `WaitingQueue` geändert hat.

Teilaufgabe b) [10%]

Ein **`QueueDisplay`** soll eine Anzeige im Wartebereich implementieren.
Erstellen Sie eine Klasse `QueueDisplay`, die das Interface `QueueObserver` implementiert!
Ein `QueueDisplay` soll per Konstruktor eine Warteschlange (`WaitingQueue`) übergeben bekommen, und sich selbst per Aufruf von `addQueueObserver(...)` dort registrieren. [*s. Code in Teilaufgabe f)*]

Eine **`DeskApp`** soll von den Sachbearbeitern an ihrem Arbeitsplatz zum Aufruf des jeweils nächsten Kunden genutzt werden können.
Erstellen Sie eine Klasse `DeskApp`, die ebenfalls das Interface `QueueObserver` implementiert und sich bei der im Konstruktor übergebenen Warteschlange (`WaitingQueue`) registriert.

[*Hinweise: `QueueDisplay` und `DeskApp` werden in den Teilaufgaben d) bzw. e) erweitert.*
Das Drücken der Schließen-Buttons in `QueueDisplay` bzw. `DeskApp` soll die gesamte Anwendung beenden.]

Wenn ein Sachbearbeiter mit seiner `DeskApp` den nächsten Kunden aufruft, soll (in der `WaitingQueue`) ein neues Objekt vom Typ **`Call`** erstellt werden. Erstellen Sie eine Klasse `Call` mit Attributen zum Speichern der Nummer des Kunden sowie der aufrufenden `DeskApp` sowie einem passenden Konstruktor!

[*Hinweis: S. Methodenaufruf `getNextCall(DeskApp desk)` von `WaitingQueue` in c)*]

Teilaufgabe c) **[10%]**

Erweitern Sie die Klasse **`WaitingQueue`** um weitere Methoden:

```
public Call getNextCall(DeskApp desk)
```

soll, sofern die Liste der wartenden Nummern (Kunden) nicht leer ist, ein neues `Call`-Objekt erzeugen, das die am längsten wartende Nummer sowie die übergebene `DeskApp` enthält. Ist die Kundenliste leer, wird `null` zurückgegeben.
Falls ein `Call`-Objekt erzeugt wurde, wird die Nummer aus der Liste der wartenden Kunden gelöscht. Die `WaitingQueue` merkt sich den neuen `Call` in einer geeigneten Variable `calls`. Außerdem merkt sie sich den `Call` als zuletzt erfolgt, aktualisiert alle registrierten `QueueObserver` und gibt den `Call` zurück.

Die Methode

```
public void confirmCall(Call call)
```

soll später aufgerufen werden, wenn ein Kunde beim Sachbearbeiter eingetroffen ist und die Bearbeitung seines Falles beginnt. Dann soll dieser `Call` aus `calls` gestrichen werden und auch nicht mehr als zuletzt erfolgter `Call` bekannt sein.

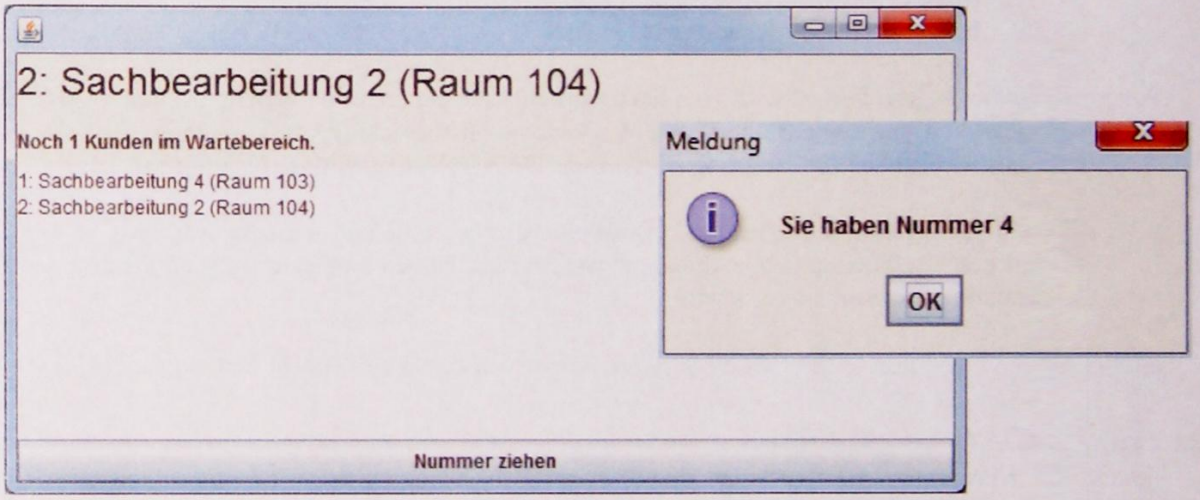
Die Methoden

```
public Call getLastCall()
public [GEEIGNETER_DATENTYP] getCurrentCalls()
public int getWaitingClients()
```

liefern eine Referenz auf den zuletzt aufgerufenen `Call` (ggf. `null`), alle aktuellen `Calls`, bzw. die Anzahl der wartenden (bisher nicht aufgerufenen) Kunden im Wartebereich.

Teilaufgabe d) **[15%]**

Erweitern Sie die Klasse **`QueueDisplay`** so, dass sie eine grafische Benutzeroberfläche erzeugt (s. Abbildung).



Das Fenster soll dabei Folgendes anzeigen:

- Details zum letzten Aufruf (`Call`), d.h. Nummer, Arbeitsplatz, Raum; bzw. „`---`“ wenn kein Aufruf vorliegt
- Anzahl der im Wartebereich verbliebenen (noch nicht aufgerufenen) Kunden
- Liste aller aktuellen Aufrufe (falls nötig mit einem Scrollbalken versehen)
- Knopf zum Ziehen einer Nummer für neu hinzukommende Kunden. Das Drücken dieses Knopfes soll selbverständlich eine neue Nummer in der Warteschlange erzeugen und sie dem Nutzer in einer Dialogbox mitteilen (s. Abbildung). Die Änderung soll allen `QueueObservern` bekannt gemacht werden.

[*Hinweis: Beachten Sie, dass Inhalte beim Aufruf der Methode `updateView()` aktualisiert werden müssen!*]

[*Hinweis: Änderung der Schriftgröße, z.B. bei einem `JLabel` namens `label`:*
`label.setFont(new Font(label.getFont().getName(), Font.PLAIN, 24));`

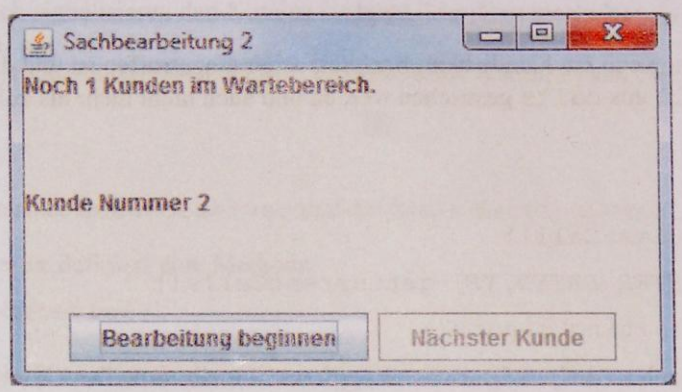
Teilaufgabe e) **[20%]**

Erweitern Sie die Klasse **`DeskApp`** so, dass sie eine grafische Benutzeroberfläche in einem eigenen Fenster erzeugt (s. Abbildung). Dafür werden weitere Attribute und ein entsprechend erweiterter Konstruktor benötigt [*s. Teil f)*]:

- Beschreibung des Arbeitsplatzes, z.B. „Sachbearbeitung 1"
- Raum, z.B. „Raum 102"
- Name des Mitarbeiters, z.B. „Herr Maier"

In der Titelzeile des Fensters soll die Beschreibung des Arbeitsplatzes stehen.
Ansonsten soll die Oberfläche folgende Elemente enthalten:

- Anzeige der verbliebenen Kunden im Wartebereich
- Anzeige der Nummer des aktuellen (mit dieser `DeskApp` aufgerufenen) Kunden, bzw. „---"
- Buttons „Bearbeitung beginnen" und „Nächster Kunde"



Zunächst soll nur der Button „Nächster Kunde" klickbar (aktiv) sein. Wird er gedrückt geschieht Folgendes:

- Wenn kein Kunde mehr wartet: Anzeige einer Dialogbox „Keine wartenden Kunden!".
- Wenn noch Kunden warten:
  1. Aufruf von `getNextCall(...)` in der `WaitingQueue`
  2. Button „Bearbeitung beginnen" wird aktiv
  3. Button „Nächster Kunde" wird inaktiv

Wenn der Kunde eingetroffen ist wird vom Sachbearbeiter der Button „Bearbeitung beginnen" gedrückt.

1. Der Button ändert seine Beschriftung in „Bearbeitung abgeschlossen" und bleibt aktiv
2. Der entsprechende `Call`-Eintrag wird aus der Warteschlange entfernt [*`confirmCall`, s. Teilaufgabe c)*]

Wird vom Sachbearbeiter anschließend auf „Bearbeitung abgeschlossen" gedrückt, werden

1. der Button „Bearbeitung abgeschlossen" wieder zu „Bearbeitung beginnen" und inaktiv
2. der Button „Nächster Kunde" aktiv

[Hinweis: Beachten Sie auch hier die passenden Aufrufe von `updateView()` bzw. `updateObservers()`!]

Teilaufgabe f) **[5%]**

Testen Sie Ihr Programm mit dem Folgenden bzw. syntaktisch angepasstem Code. Es sollten drei `DeskApp`-Fenster und zwei `QueueDisplay`-Fenster starten.

```
WaitingQueue qu = new WaitingQueue();
for (int i=0; i<5; i++)  // Erzeugen von 5 Einträgen in der Warteschlange
    System.out.println("Nummer "+qu.getNextQueueNumber()+" gezogen");
new DeskApp("Sachbearbeitung 1", "Raum 102", "Herr Maier", qu);
new DeskApp("Sachbearbeitung 2", "Raum 104", "Herr Schmid", qu);
new DeskApp("Sachbearbeitung 4", "Raum 103", "Herr Müller", qu);
new QueueDisplay(qu);
new QueueDisplay(qu);
```

Teilaufgabe g) **[5%]**

Erweitern Sie **`QueueDisplay`** um einen Thread, der (wenn gleichzeitig mehrere `Call`s vorliegen) im oberen Bereich z.B. alle 5 Sekunden einen anderen `Call` anzeigt!