Aufgabenblatt 2

Aufgaben zu RMI

Verteilte Systeme
DHBW Karlsruhe
Stand Sommersemester 2018

Prof. Dr. Dirk Eisenbiegler

Aufgabe 2.1 Time-Service mit RMI

Programmieren Sie einen Time-Service mit RMI.

Analog zu den Aufgaben im vorangegangenen Aufgabenblatt, soll der Service die aktuelle Uhrzeit und das aktuelle Datum an die Clients ausliefern.

Anders als in der bisher betrachteten Lösung, sollen Datum und Uhrzeit nicht getrennt abfragbar sein, sondern werden gemeinsam ausgeliefert. Java enthält in der Standard-Library die Klasse *Date*. Jedes *Date*-Objekt repräsentierte einen Zeitpunkt bestehend aus Datum und Uhrzeit. Mit dem Aufruf des Defaultkonstruktors von *Date* erhält man den aktuellen Zeitpunkt und das aktuelle Datum.

- ✗ Konstruieren Sie eine RMI-Klasse mit dem Namen *TimeService*.
- ✗ Diese Klasse soll eine einzige Methode haben: *getDateAndTime*. Die Methode *getDateAndTime* hat keine Parameter und einen Rückgabewert vom Typ *Date*. Die Methode soll den aktuellen Serverzeitpunkt bestimmen und an den Client ausliefern.
- ✗ Starten Sie eine Registry und legen Sie mit *bind* eine Instanz von *TimeService* in der Klasse ab.
- ✗ Schreiben Sie ein Client-Pogramm, das die Methode *getDateAndTime* über einen entfernten Methodenaufruf aufruft und den Rückgabewert auf der Konsole ausgibt.

Aufgabe 2.2 Ereignisse

Die *TimeService*-Klasse aus der vorangegangenen Aufgabenstellung soll um Ereignisse erweitert werden. Ein Ereignis ist eine Datenstruktur, die sich aus einem Zeitpunkt in Form eines *Date*-Objekts und einer Beschreibung des Ereignisses in Form eines Strings zusammensetzt.

Der Time-Service soll intern eine Liste mit Ereignissen verwalten. Deklarieren Sie zunächst eine Klasse *Event*, die ein einzelnes Ereignis repräsentiert und verwenden Sie dann eine Instanz von *Vector<Event>*, um darin eine Menge von Ereignissen zu speichern.

Ergänzen Sie die Klasse *TimeService* um die folgenden Methoden:

- ✗ *addEvent*
  - ⇨ fügt einen Event zur Liste hinzu
  - ⇨ Parameter: Event
  - ⇨ kein Rückgabewert
- ✗ *getAllEvents*
  - ⇨ alle Events, die im Time-Service gespeichert sind
  - ⇨ kein Parameter
  - ⇨ Rückgabewert: *Vector<Event>*
- ✗ *getNextEvent*
  - ⇨ bestimmt den zeitlich nächsten Event, der im Time-Service gespeichert ist – Rückgabewert *null*, falls es keinen solchen gibt
  - ⇨ kein Parameter
  - ⇨ Rückgabewert: *Event*
- ✗ *getFutureEvents*
  - ⇨ alle zukünftigen Events, die im Time-Service gespeichert sind
  - ⇨ kein Parameter
  - ⇨ *Rückgabewert: Vector<Event>*

Aufgabe 2.3 RMI-Callbacks

Der Time-Service soll um eine Funktion erweitert werden, mit der der Client via Callbacks über eintretende Ereignisse informiert wird.

Schreiben Sie ein Interface mit dem Namen *EventListener*. Das Interface *EventListener* soll eine einzige Methode mit dem Namen *handleEvent* enthalten. Diese Methode hat einen Parameter vom Typ *Event* und keinen Rückgabewert.

Der Time-Service soll eine Liste von EventListener-Objekten verwalten. Verwenden Sie dazu ein zusätzliches Attribut vom Typ *Vector<EventListener>*.

Ergänzen Sie die Klasse TimeService um die folgenden Methoden:

- ✗ *addEventListener*
  - ⇨ fügt ein EventListener-Objekt zur Liste hinzu
  - ⇨ Parameter: EventListener
  - ⇨ kein Rückgabewert
- ✗ *removeEventListener*
  - ⇨ entfernt ein EventListener-Objekt aus der Liste
  - ⇨ Parameter: EventListener
  - ⇨ kein Rückgabewert

Sorgen Sie dafür, dass die *handleEvent*-Methode von allen *EventListener* aus der Liste aufgerufen wird, sobald der jeweilige Zeitpunkt erreicht ist. Das jeweilige Event-Objekt soll der *handleEvent*-Methode übergeben werden.

Tipps:

- ✗ Starten Sie im Konstruktor von *TimeService* einen Thread.
- ✗ der Thread führt eine Endlosschleife aus
- ✗ in jedem Schleifendurchlauf Blockierung über *Thread.sleep* bis zum nächsten Ereignis
- ✗ nach Thread.sleep werden die handleEvent-Methoden aufgerufen
- ✗ *interrupt* für den Fall, dass ein neuer Event hinzugefügt wird
- ✗ dann: Neuberechnung der *sleep*-Dauer

Aufgabe 2.4 Einfacher Client für RMI-Callbacks

Schreiben Sie ein einfaches Client-Programm, das mit der Methode *addEventListener* einen Event-Handler beim Time-Service anmeldet. Der Event-Handler soll die eingehenden Ereignisse mit *System.out.println* auf der Konsole ausgeben.

Aufgabe 2.5 Client-Programm mit GUI

Schreiben ein Client-Programm für den Time-Service mit einer graphischen Benutzeroberfläche.

Verwenden Sie zur GUI-Programmierung irgend eine GUI-API wie AWT, Swing, SWT oder JavaFX.

Das Client-Programm soll folgende Funktionen enthalten:

- ✗ Anzeige der Server-Uhrzeit mit einer automatischen Aktualisierung jede Minute.
- ✗ Anzeige der serverseitig gespeicherten Ereignisse, die noch ausstehen, mit Zeitpunkt und Beschreibung
- ✗ Funktion, um neue Ereignisse hinzuzufügen. Formular mit zwei Feldern für Zeitpunkt und Beschreibung und einem Abschicken-Button.
- ✗ Anzeige von eintretenden Ereignissen durch Meldung in Dialogbox. (Callbacks)