Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 2 SoSe 2023
Fakultät für Angewandte Informatik
Institut für Informatik
Dr. M. Frieb, Dr. D. Müller, V. Le Claire, M. Steinberger 23.06.2023

Übungsblatt 9

Abgabe: 03.07.2023, 09:00 Uhr (via Digicampus an den Tutor: .java-Dateien für Code, .uxf für UML, .pdf für alles andere)

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt!).
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.

* leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe
Allgemeine Hinweise zur Erstellung und Abgabe von UML-Diagrammen:

- UML-Diagramme sind mit dem Editor **UMLet**[1] zu erstellen (es darf auch die webbasierte Variante **UMLetino**[2] verwendet werden). Damit die Leserichtung von Assoziationen beim PDF-Export aus UMLet erhalten bleibt, müssen Sie eine Schriftart installieren, welche die verwendeten Zeichen beinhaltet, wie z.B. `https://fonts2u.com/download/free-sans.family`. Exportieren Sie die enthaltene .zip-Datei und tragen Sie den absoluten Pfad zu den Dateien in UMLet unter *Optional font to embedd in PDF* ein, verwenden Sie die `Oblique`-Dateien für *italic* und *bold+italic* Text.
- Die (graphischen) Syntaxvorgaben aus der Vorlesung sind einzuhalten.
- Modellieren Sie Attribute, Konstruktoren und Operationen im Stil der Vorlesung (z.B. im Hinblick auf die Benennung).
- Erfinden Sie keine Daten oder Operationen hinzu, die nicht der jeweiligen Beschreibung entnommen werden können.
- Wenn Sie eine API-Klasse bzw. -Schnittstelle als Attribut oder Eingabeparameter verwenden, reicht die Angabe als `java.module.package.Klasse` ohne Modellierung als eigene Klasse bzw. Schnittstelle.
- Wenn Sie Beziehungen zu API-Klassen bzw. Schnittstellen wie Vererbung oder Implementierung benötigen, modellieren Sie die Klassen bzw. Schnittstellen mit derselben Namensgebung `java.module.package.Klasse`.
- Operationen und Attribute von API-Klassen und Assoziationen zwischen API-Klassen werden nur dann modelliert, wenn sie für die Funktionalität der modellierten Anwendung notwendig sind oder wenn die Aufgabenstellung es explizit verlangt.
- UML-Diagramme sind **sowohl** als UMLet-Datei (`*.uxf`) **als auch** als PDF abzugeben.

[1] `https://www.umlet.com`
[2] `http://www.umletino.com`

Aufgabe 41 + 42 ** *(Modellierung eines vollständigen Systems, 50 Minuten)*

In dieser Aufgabe sollen Sie alle in der folgenden Systembeschreibung enthaltenen Informationen zu einem **Hotelbuchungssystem** in einem in sich konsistenten UML-Klassendiagramm darstellen.

In einem Hotel arbeiten Angestellte. Ein Hotel besteht aus einer beliebigen Anzahl von Räumen. Ein Raum beinhaltet eine ganzzahlige Anzahl von Betten und hat einen festen reellwertigen Preis pro Tag. Beide Werte sind positiv. Vorerst sollen zwei Arten von Räumen existieren: Familienzimmer haben mindestens drei Betten. Eine Luxussuite hat genau einen Angestellten als Privatkellner. Das Hotel soll leicht um weitere Arten von Räumen erweitert werden können. Ein Hotel wird von Gästen besucht. Ein Gast hat eine Zeichenkette als Name. Ein Gast kann Stammgast sein. Ein Gast kann einen oder mehrere Räume buchen. Für jede Buchung sind Beginn- und Enddatum zu speichern, wobei das Enddatum immer nach dem Beginndatum liegen muss. Die Buchungen eines Raums dürfen sich zeitlich nicht überlappen. Eine Buchung ist eine Art von Rechnung: Ein Gast bezahlt Rechnungen. Von einer Rechnung kann der Preis bestimmt werden. Wenn der Gast, der die Rechnung bezahlt, ein Stammgast ist, so bekommt der Gast einen Rabatt von 20% auf alle Rechnungen. Damit das an das Hotel angeschlossene Restaurant in Zukunft auch an das beschriebene System angeschlossen werden kann, sollen weitere Arten von Rechnungen leicht hinzugefügt werden können.

Modellieren Sie geeignete Klassen mit passenden Attributen und zugehörigen Verwaltungsoperationen. Modellieren Sie zusätzlich geeignete Beziehungen zwischen diesen Klassen.

Aufgabe 43 + 44 *(Sequenzdiagramm modellieren und implementieren, 43 Minuten)*

In den folgenden Aufgaben sollen Sie so detailliert wie möglich ein Sequenzdiagramm modellieren oder ein Sequenzdiagramm implementieren. Modellieren Sie dabei jeweils auch alle ineinander- und hintereinandergeschachtelten Operationsaufrufe. Sollten Anweisungen nicht in einem Sequenzdiagramm darstellbar sein, so ignorieren Sie diese.

a) (*, 6 Minuten)
Betrachten Sie folgenden Programmcode:

```
package blatt09;

public class A43a_main {
    public static void main(String[] args) {
        String combined = String.join(" ", args);
        System.out.println(combined);
    }
}
```

Stellen Sie Aufruf und Ausführung der Methode `main` so detailliert wie möglich als Sequenzdiagramm dar. Insbesondere ist jeder Methodenaufruf als separate Botschaft zu modellieren.

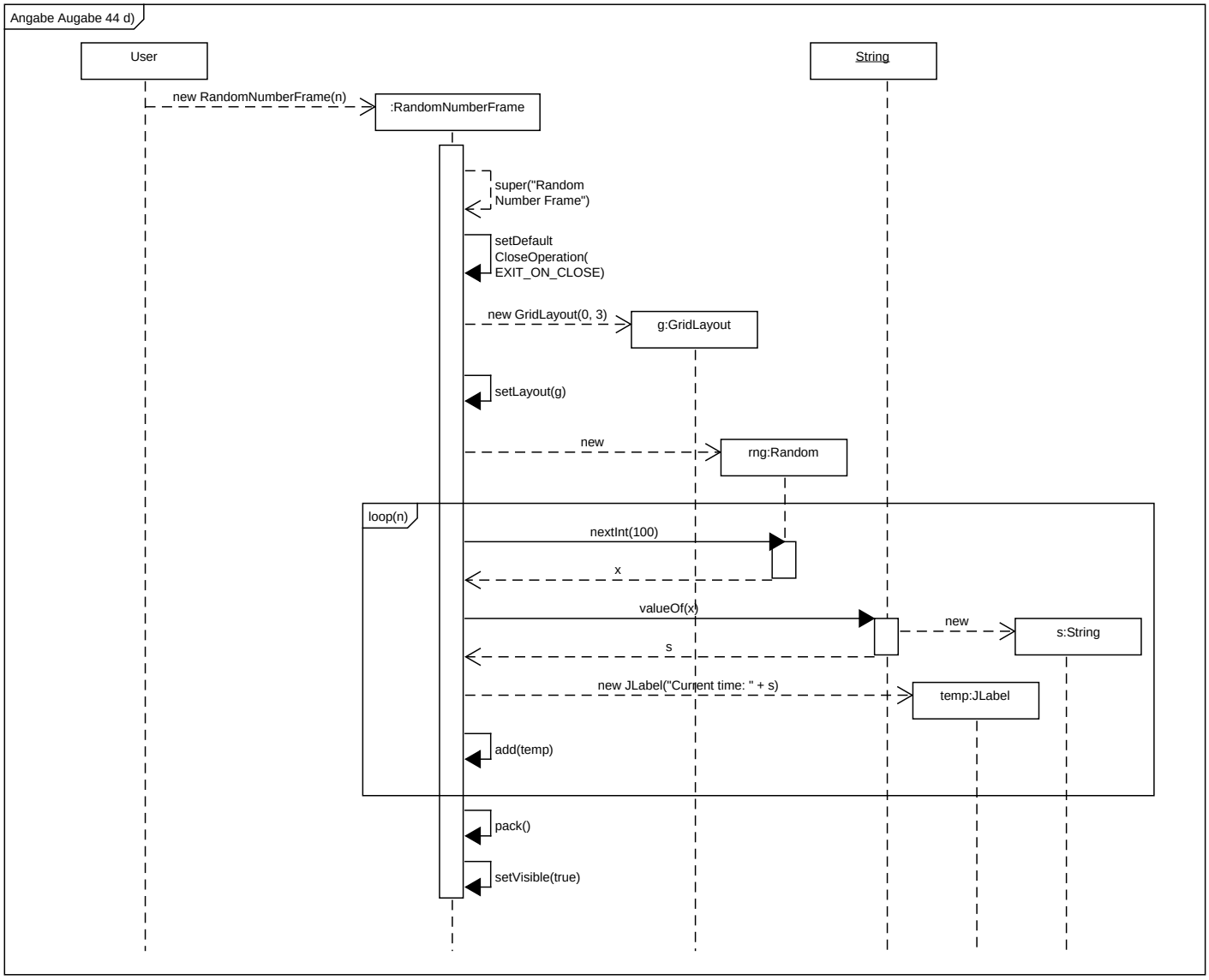
b) (**, 11 Minuten)
Betrachten Sie folgenden Programmcode:

```
    public BankAccount(String accountHolderName) {
        setOpeningDate();
        setAccountHolderName(accountHolderName);
    }

    private static boolean checkAccountHolderName(String accountHolderName) {
        return accountHolderName != null && accountHolderName.matches("[A-Z][a-z]*(
        ↪  [A-Z][a-z]*)*");
    }

    public void setAccountHolderName(String accountHolderName) {
        if (checkAccountHolderName(accountHolderName)) {
            this.accountHolderName = accountHolderName;
        } else {
            throw new IllegalArgumentException("Wrong name format");
        }
    }

    private void setOpeningDate() {
        this.openingDate = LocalDate.now();
    }
}
```

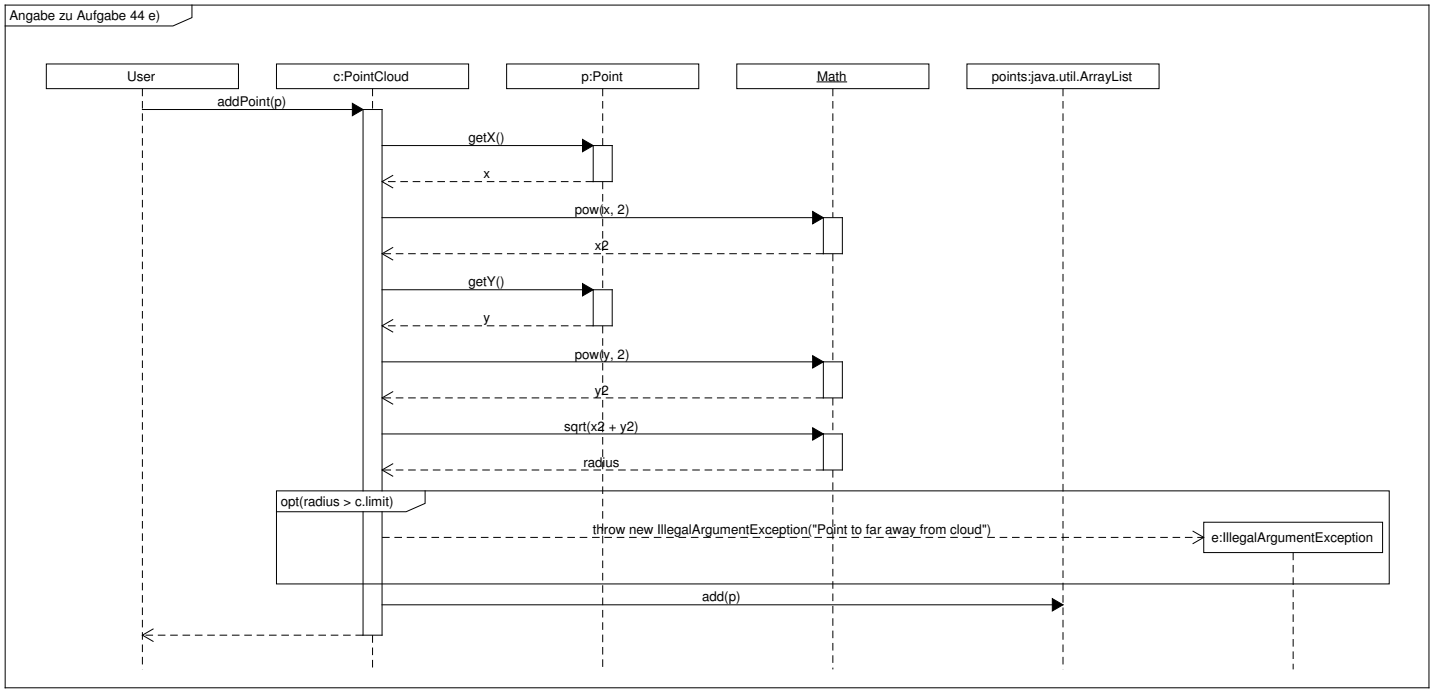
Stellen Sie einen Aufruf des Konstruktors so detailliert wie möglich als Sequenzdiagramm dar. Modellieren Sie dabei jeweils auch alle ineinander- und hintereinandergeschachtelten Operationsaufrufe. Insbesondere ist jeder Methodenaufruf als separate Botschaft zu modellieren.

c) (**, 10 Minuten)
Modellieren Sie durch ein Sequenzdiagramm den Aufruf und die Implementierung des Konstruktors eines sichtbaren Fensters. Das Fenster hat die Größe `250 x 100` und den Titel `Aufgabe 43 c)`. In der Mitte des Fensters ist ein Button mit dem Text `Current time` und der aktuellen Zeit im Format `HH:MM:SS`. Das Fenster soll durch das Fensterkreuz geschlossen werden können und keine weiteren Komponenten oder Funktionalitäten enthalten.

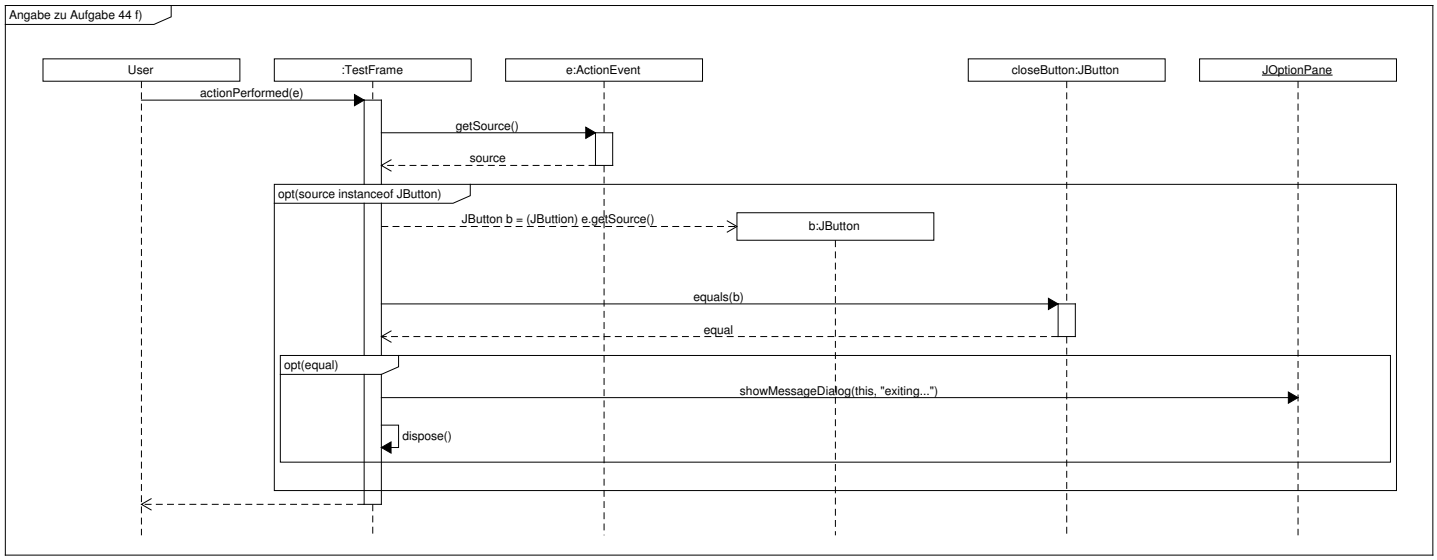
d) (*, 8 Minuten)
Implementieren Sie eine Klasse `RandomNumberFrame` gemäß dem folgenden Sequenzdiagramm.



e) (*, 4 Minuten)
Implementieren Sie die Methode `addPoint` der Klasse `PointCloud` gemäß dem folgenden Sequenzdiagram.



f) (*, 4 Minuten)
Implementieren Sie die Methode `actionPerformed` der Klasse `TestFrame` gemäß dem folgenden Sequenzdiagram.



Aufgabe 45 * *(Aufgabe zum Uni.Profi-Training)*

Das **achte UniProfi-Kapitel „Strategisch. Dynamisch. Individuell.“** ist in Digicampus **ab Montag, dem 26.06.2023** freigeschaltet. Klicken Sie sich rein und verbessern Sie Ihre Studierkompetenzen!
Für die Bearbeitung der Aufgaben in Kapitel 8 können Sie **0,5 Bonuspunkte** erhalten. Denken Sie außerdem daran, dass Sie bei mindestens 7 erfolgreich bearbeiteten UniProfi-Kapiteln (also 3,5 gesammelten Bonuspunkten) zusätzlich 0,5 Bonuspunkte erhalten. Die Einreichung der Aufgaben ist **bis Montag, 03.07. (9 Uhr)** möglich. Wenn Sie dem Kurs noch nicht beigetreten sind, können Sie dies nun nachholen: https://digicampus.uni-augsburg.de/dispatch.php/course/details?sem_id=bffff7d141569f9b1a3baf773286aa42&again=yes
(Passwort: BJf7S73uNX)

Viel Erfolg und viel Spaß!