Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 2 SoSe 2024
Fakultät für Angewandte Informatik
Institut für Informatik
Prof R. Lorenz, L. Petrak, J. Linne, V. Le Claire, P. Schalk 26.04.2024

Lösungsvorschlag zu Übungsblatt 2

Abgabe: 06.05.2024, 10:00 Uhr (im Digicampus via VIPS: .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

Aufgabe 5 + 6 *(Eigene Datenklassen modellieren und implementieren, 40 Minuten)*

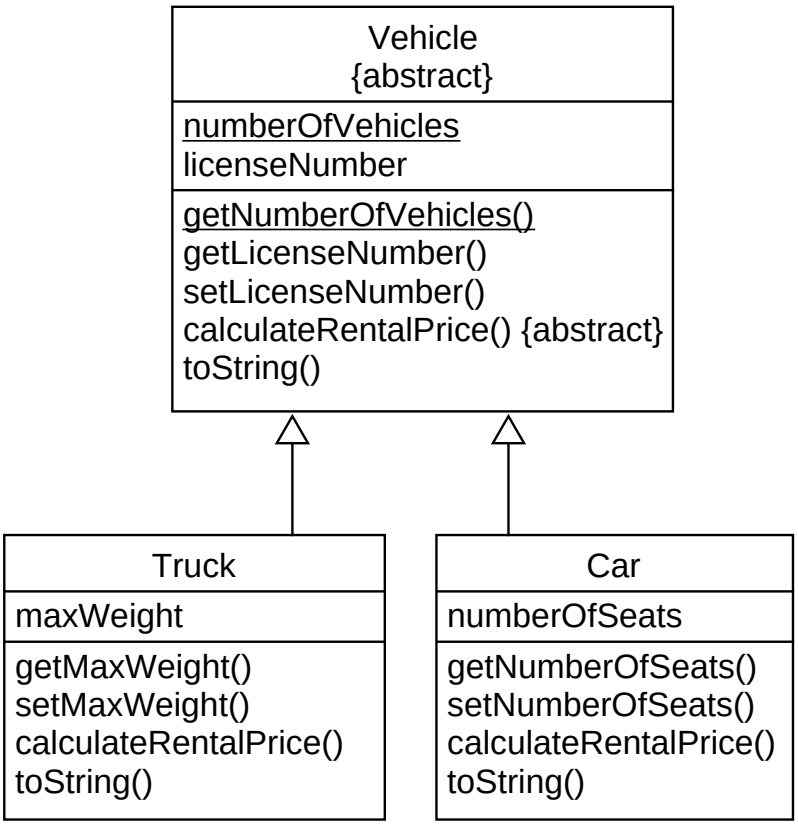
In dieser Aufgabe sollen Sie drei Klassen zum Verwalten von Fahrzeugen für eine Fahrzeugvermietung modellieren und implementieren. Es gibt keine allgemeinen Fahrzeuge, sondern nur Autos und LKWs. Trotzdem soll die Möglichkeit offen gehalten werden, das Programm später um zusätzliche Fahrzeugtypen zu erweitern. Implementieren Sie deshalb eine **abstrakte** Klasse `Vehicle` und zwei Unterklassen `Car` und `Truck` dieser abstrakten Klasse. Jedes Fahrzeug hat ein Kennzeichen und stellt Möglichkeiten zur Verfügung, dieses zu lesen und zu ändern. Außerdem erfordert die Klasse `Vehicle`, dass jede Unterklasse eine Möglichkeit zur Verfügung stellt, den reellwertigen Mietpreis des Fahrzeuges zu bestimmen. Weil das für allgemeine Fahrzeuge nicht sinnvoll ist, soll diese Methode abstrakt sein. Zusätzlich wird gespeichert, wie viele Fahrzeuge insgesamt aktuell verwaltet werden. Es soll auch eine Methode angeboten werden, um die aktuelle Anzahl an verwalteten Fahrzeugen zu erhalten. Die Darstellung eines Fahrzeugs als Zeichenkette sieht wie folgt aus:

```
"License plate: A-UA-1234"
```

Ein Auto hat eine ganzzahlige Anzahl an Sitzplätzen und stellt Möglichkeiten zur Verfügung, diese anzupassen und abzufragen. Der Mietpreis eines Autos ist das 2.5-fache seiner Anzahl an Sitzplätzen. Ein LKW hat ein reellwertiges Maximalgewicht und stellt Möglichkeiten zur Verfügung, dieses anzupassen und abzufragen. Der Mietpreis eines LKWs ist das 10-fache seines Maximalgewichts. Die Zeichenkettendarstellung der beiden Unterklassen sieht wie folgt aus:

```
"License plate: A-UA-1234 Number of seats: 4 Rental price: 10.0"
"License plate: A-UA-1235 Maximum weight: 40.0 Rental price: 400.0"
```

a) *(*, Klassenkarte entwerfen, 12 Minuten)*
Entwerfen Sie passende Klassenkarten für Fahrzeuge, Autos und LKWs. Achten Sie darauf, das Verhältnis zwischen Fahrzeugen und Autos bzw. LKWs korrekt darzustellen.
Hinweis: Das Überschreiben von Methoden in Unterklassen modellieren wir, indem wir die zu überschreibende Methode auch in der Klassenkarte der Unterklasse aufführen.
Lösung:



b) *(**, Klassen implementieren, 23 Minuten)*
Implementieren Sie die Klassen `Vehicle`, `Car` und `Truck`. Versehen Sie auch die Klasse `Vehicle` mit einem Konstruktor, den Sie in den Konstruktoren von `Car` und `Truck` verwenden. Überschreiben Sie die `toString()`-Methode der Klasse `Vehicle` und verwenden Sie diese in der Implementierung der `toString()`-Methoden von `Car` und `Truck`.

Lösung:

```
public abstract class Vehicle {
    private String licenseNumber;
    private static int vehicleCount;

    public Vehicle(String licenseNumber) {
        setLicenseNumber(licenseNumber);
        vehicleCount++;
    }

    public static int getVehicleCount() {
        return vehicleCount;
    }

    public String getLicensePlate() {
        return licensePlate;
    }

    private void setLicenseNumber(String licenseNumber) {
        this.licenseNumber = licenseNumber;
    }

    public abstract double calculateRentalPrice();

    @Override
    public String toString() {
        return "License Plate: " + licenseNumber;
    }
}
```

Car

```
public class Car extends Vehicle {

    private int numberOfSeats;
    public Car(String licensePlate, int numberOfSeats) {
        super(licensePlate);
        setNumberOfSeats(numberOfSeats);
    }

    public int getNumberOfSeats() {
        return numberOfSeats;
    }

    public void setNumberOfSeats(int numberOfSeats) {
        this.numberOfSeats = numberOfSeats;
    }

    public double calculateRentalPrice() {
        return 2.5 * getNumberOfSeats();
    }

    @Override
    public String toString() {
        return super.toString() + " Number of seats: " + numberOfSeats + " Rental price: " +
        ↪  calculateRentalPrice();
}
```

Truck

```
public class Truck extends Vehicle {
    private double maxWeight;

    public Truck(String licensePlate, double maxWeight) {
        super(licensePlate);
        setMaxWeight(maxWeight);
    }

    public double getMaxWeight() {
        return maxWeight;
    }

    public void setMaxWeight(double maxWeight) {
        this.maxWeight = maxWeight;
    }

    public double calculateRentalPrice() {
        return 10 * getMaxWeight();
    }

    @Override
    public String toString() {
        return super.toString() + " Maximum weight: " + maxWeight + " Rental price: " +
        ↪  calculateRentalPrice();
    }
}
```

c) *(**, Programmklasse implementieren, 5 Minuten)*
Implementieren Sie eine Programmklasse, die

- ein `Vehicle`-Array `vehicles` der Größe 2 anlegt,
- ein Auto mit dem Kennzeichen `"A-UA-1234"` und 4 Sitzen anlegt und an der ersten Stelle des Arrays speichert,
- einen LKW mit dem Kennzeichen `"A-UA-1235"` und Maximalgewicht 40.0 anlegt und an der zweiten Stelle des Arrays speichert und
- die Zeichenkettendarstellung aller Fahrzeuge im Array mit einer Schleife ausgibt, ohne `vehicles.length` zu verwenden.

Lösung:

```
public class VehicleMain {
    public static void main(String[] args) {
        Vehicle[] vehicles = new Vehicle[2];
        vehicles[0] = new Car("A-UA-1234", 4);
        vehicles[1] = new Truck("A-UA-1235", 40.0);
        for (int i = 0; i < Vehicle.getVehicleCount(); i++) {
            System.out.println(vehicles[i].toString());
        }
    }
}
```

Aufgabe 7 *(Klassen für Zeichenketten benutzen, 20 Minuten)*

In den folgenden Teilaufgaben sollen Sie jeweils eine Programmklasse implementieren. Benennen Sie die Programmklasse jeweils nach der Nummer der Aufgabe.
In allen Aufgaben ist mit Ausgabe die Ausgabe auf der Kommandozeile (= Standardausgabe) gemeint.

a) *(**, Klasse `StringBuilder`, 9 Minuten)*
Schreiben Sie eine Methode, die zwei Zeichenketten `text` und `key` sowie eine ganze Zahl `delete` übergeben bekommt und eine Zeichenkette zurückgibt:
Mithilfe eines `StringBuilder`-Objekts mit dem Inhalt `text` soll jedes Vorkommen des Substrings `key` sowie die darauf folgenden `delete` Zeichen aus `text` gelöscht werden. Am Ende soll die resultierende Zeichenkette ausgegeben werden. Den Kopf der Methode legen Sie selbst fest.

Beispiel: Gilt `text = "Eine bunte Kuh"`, `key = " "` und `delete = 4`, so soll die Methode `"Einee"` zurück geben, weil `" bunt"` sowie `" Kuh"` gelöscht wurden.

Lesen Sie in der `main`-Methode vom Benutzer eine Zeile von der Kommandozeile für `text`, eine weitere Zeile für `key` und eine Zahl für `delete` ein und testen ihre Methode damit. Geben Sie das Ergebnis Ihrer Methode auf der Kommandozeile aus. **Lösung:**

```
import java.util.Scanner;

public class StringCreation {
    public static void main(String[] args) {
        Scanner myScanner = new Scanner(System.in);
        System.out.println("Enter the text:");
        String text = myScanner.nextLine();
        System.out.println("Enter the key:");
        String key = myScanner.nextLine();
        System.out.println("How many characters do you want to delete:");
        int delete = myScanner.nextInt();
        System.out.println(deleteFromString(text, key, delete));
        myScanner.close();
    }

    public static String deleteFromString(String text, String key, int delete) {
        StringBuilder s = new StringBuilder(text);
        while (s.indexOf(key) != -1) {
            int i = s.indexOf(key);
            s.delete(i, i + key.length() + delete);
        }
        return s.toString();
    }
}
```

b) *(**, Gleichheit von `String`- bzw. `StringBuilder`-Objekten, 11 Minuten)*
Schreiben Sie ein übersetzbares Java-Programm, das

- ein `String`-Objekt `str1` mit dem Wert `"abc"` erstellt und ein `String`-Objekt `str2` mit dem Wert `"a" + "bc"` enthält,
- ein leeres `StringBuilder`-Objekt `sb1` sowie ein `StringBuilder`-Objekt `sb2` mit dem Konstruktur-Parameter `"abc"` erzeugt, mithilfe der `append`-Methode an `sb1` zuerst `"ab"` und dann `"c"` anhängt,
- alle vier Objekte in jeweils einer eigenen Zeile zusammen mit ihren Namen ausgibt,
- ein `String`-Objekt `str3` anlegt und ihr den Wert von `str1.concat("test")` zuweist,
- alle 5 Objekte in jeweils einer eigenen Zeile zusammen mit ihren Namen ausgibt,
- mit der `equals`-Methode zuerst für `str1` und `str2` testet und ausgibt, ob diese Objekte gleich sind und
- dann für `sb1` und `sb2` mit der `equals`-Methode ausgibt, ob diese Objekte gleich sind.

Lösung:

```
package blatt02;

public class StringEquality {
    public static void main(String[] args) {
        String str1 = "abc";
        String str2 = "a" + "bc";
        StringBuilder sb1 = new StringBuilder();
        StringBuilder sb2 = new StringBuilder("abc");
        sb1.append("ab");
        sb1.append("c");
        System.out.println("str1: " + str1);
        System.out.println("str2: " + str2);
        System.out.println("sb1: " + sb1);
        System.out.println("sb2: " + sb2);
        String str3 = str1.concat("test");
        System.out.println("str1: " + str1);
        System.out.println("str2: " + str2);
        System.out.println("sb1: " + sb1);
        System.out.println("sb2: " + sb2);
        System.out.println("str3: " + str3);
        System.out.println("str1 equals str2: " + str1.equals(str2));
        System.out.println("sb1 equals sb2: " + sb1.equals(sb2));
    }
}
```

Aufgabe 8 *(Zeitdarstellung in Java, 20 Minuten)*

In dieser Aufgabe arbeiten Sie mit Klassen zur Verwaltung von Datums- und Zeitangaben aus dem Paket `java.time`. Erstellen Sie für jede der Teilaufgaben eine eigene Programmklasse.

a) *(Einfache Zeitdarstellungen, *, 6 Minuten)*
Erstellen Sie ein Programm, das

- ein `LocalDate`-Objekt vom aktuellen Tag erzeugt und auf der Kommandozeile ausgibt,
- daraus ein neues `LocalDate`-Objekt `date1` erzeugt, das zwei Monate nach dem aktuellen Datum liegt und auf der Kommandozeile ausgibt,
- daraus wieder ein neues `LocalDate`-Objekt `date2` erzeugt, das 10 Tage vor `date1` liegt und auf der Kommandozeile ausgibt und
- berechnet, wie viele Tage zwischen dem 01.01.2024 und `date2` liegt und das Ergebnis auf der Kommandozeile ausgibt. Verwenden Sie dazu die `between`-Methode des Objekts `ChronoUnit.DAYS`.

Lösung:

```java
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;

public class TimeRepresentation {
    public static void main(String[] args) {
        LocalDate today = LocalDate.now();
        System.out.println("Today is " + today);
        LocalDate date1 = today.plusMonths(2);
        System.out.println("Two month later is " + date1);
        LocalDate date2 = date1.minusDays(10);
        System.out.println("Ten days earlier is " + date2);
        LocalDate date3 = LocalDate.of(2024, 1, 1);
        System.out.println("Days between " + date3 + " and " + date1 + ": " +
        ↪  ChronoUnit.DAYS.between(date3, date1));
    }
}
```

b) *(Zeitraum der STUDIS-Anmeldung, *, 7 Minuten)*
Erstellen Sie ein Programm, das

- ein `LocalDate`-Objekt für das Datum des Beginns der STUDIS-Anmeldung anlegt,
- ein `LocalTime`-Objekt für die Uhrzeit des Beginns der STUDIS-Anmeldung anlegt,
- diese beiden Objekte zu einem `LocalDateTime`-Objekt kombiniert,
- direkt ein `LocalDateTime`-Objekt für das Ende der STUDIS-Anmeldung anlegt,
- ein `Duration`-Objekt erstellt, das die Zeitspanne zwischen den beiden Terminen enthält und
- die Länge des Anmeldungszeitraums in Stunden auf der Kommandozeile ausgibt.

Den Zeitraum der STUDIS-Anmeldephase entnehmen Sie dem Foliensatz „Informatik2-Organisation“ im Digicampus.

Lösung:

```java
import java.time.LocalTime;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.Duration;

public class STUDISRegistration {
    public static void main(String[] args) {
        LocalDate date1 = LocalDate.of(2024, 6, 6);
        LocalTime time1 = LocalTime.of(12, 0);
        LocalDateTime dateTime1 = date1.atTime(time1);
        LocalDateTime dateTime2 = LocalDateTime.of(2024, 6, 17, 12, 0);
        Duration registrationDuration = Duration.between(dateTime1, dateTime2);
```

```
        System.out.println("Duration of STUDIS registration period: " +
        ↪  registrationDuration.toHours() + " hours.");
    }
}
```

c) *(Weitere `LocalDate`-Methoden und Enumeration `DayOfWeek`, **, 7 Minuten)*
Erstellen Sie ein Programm, das bestimmt, welcher der häufigste Wochentag zu Beginn eines Monats im Jahr 2024 ist:

- Legen Sie Array der Länge 7 an.
- Iterieren Sie über alle Monate und erzeugen Sie jeweils ein `LocalDate`-Objekt für den 1. des jeweiligen Monats.
- Erhöhen Sie jeweils den $(i-1)$-ten Eintrag des Arrays für den i-ten Wochentag um 1: mit der Klassenmethode `getValue()` erhalten Sie eine `int`-Repräsentation des `DayOfWeek`-Objekts: 1 für Montag, 2 für Dienstag usw.
- Bestimmen Sie in einer neuen Schleife für das Array den Index des Eintrags mit dem größten Wert: z.B. ist in dem Array `{3, 9, 10, 2}` der Index des Eintrags mit dem größten Wert 2, weil 10 größer ist als alle anderen Einträge.
- Geben Sie auf der Kommandozeile aus, mit welchem Wochentag die meisten Monate im Jahr 2024 anfangen.

Lösung:

```
import java.time.DayOfWeek;
import java.time.LocalDate;

public class Weekdays {
    public static void main(String[] args) {
        int[] count = new int[7];
        for (int i = 1; i <= 12; i++) {
            count[LocalDate.of(2024, i, 1).getDayOfWeek().getValue() - 1]++;
        }
        int index = 0;
        for (int i = 1; i < 7; i++) {
            if (count[i] > count[index]) {
                index = i;
            }
        }
        System.out.println("This year, most months start with " + DayOfWeek.of(index + 1));
    }
}
```