HSLU Hochschule Luzern

Objektorientierte Programmierung (OOP)

Übung SW02: Klassen und Datentypen

Themen: Klassen (O02) und Datentypen (O03)
Zeitbedarf: ca. 120min.

Roland Gisler, Version 1.5.1 (HS24)

1 Temperatur: Eine erste, einfache Klasse

1.1 Lernziele

- Schrittweise Erstellung einer ersten einfachen Klasse.
- Verständnis von Datentypen, Attributen, Methoden und Konstruktoren.
- Implementation von einfachen Berechnungen.

1.2 Grundlagen

In dieser Aufgabe erstellen Sie eine erste, ganz einfache Klasse in BlueJ. Grundlage sind die Kenntnisse aus den Inputs **O02_IP_Klassen** und **O03_IP_Datentypen-Operatoren**.

Diese Übung bearbeiten Sie am einfachsten mit BlueJ. Erstellen Sie dazu ein neues Projekt für die Aufgaben der zweiten Semesterwoche.

Hinweis: Diese Klassen haben noch nicht den Anspruch perfekt zu sein, sondern es geht um erste Versuche! Wir werden sie in den folgenden Wochen sukzessive weiter entwickeln und verbessern!

1.3 Aufgaben

a.) Erstellen Sie eine neue Klasse mit dem Namen `Temperatur`. Überlegen Sie sich welchen Datentyp Sie verwenden, um die auf der Erde üblichen (Umgebungs-)Temperaturen auf mindestens zwei Nachkommastellen genau abbilden zu können. Legen Sie dafür ein entsprechendes Attribut in der Klasse an.

b.) Haben Sie sich überlegt in welcher Einheit (Grad Celsius, Kelvin, Grad Fahrenheit?) Sie die Temperatur ablegen? Ist das aus der Namensgebung des Attributes ersichtlich? Verbessern Sie ggf. Ihre Implementation!

c.) Implementieren Sie eine einfache Methode, mit welcher Sie die im Attribut enthaltene Temperatur abfragen können, in dem Sie einfach den Wert zurückliefern. Methoden dieser Art bezeichnet man als Getter-Methoden. Sie erhalten den Namen des Attributes (mit grossem Anfangsbuchstaben), dem man ein `get` voranstellt: Für ein Attribut `int demo` würde die Getter-Methode somit `int getDemo()` heissen. Probieren (testen) Sie Ihre Methode interaktiv aus.

d.) Analog zu den Getter-Methoden gibt es auch Setter-Methoden, mit welchen man sofern gewünscht die Werte von Attributen verändern kann. Erstellen Sie eine Setter-Methode für die Temperatur. Für die Namensgebung gilt eine analoge Regel zu den Gettern: Für ein beispielhaftes Attribut `int demo` heisst die Setter-Methode entsprechend `void setDemo(int)`. Testen Sie auch diese Methode!

e.) Vermutlich haben Sie sich entschlossen die Temperatur in Grad Celsius zu speichern? Nun möchten wir die Temperatur aber auch in Kelvin wissen. Schreiben Sie eine Methode, welche die im Attribut gespeicherte Temperatur von Celsius in Kelvin umrechnet (dabei aber nicht verändert!) und zurückgibt. Formel für Umrechnung: $T_K = T_C + 273.15$

f.) Sie möchten Ihre Software auch in den USA einsetzen. Darum brauchen wir den Temperatur nun auch noch in Fahrenheit. Schreiben Sie auch dafür eine Methode. Umrechnung: $T_F = T_C \cdot 1{,}8 + 32$ Unsere Software soll Werte im Bereich von 0°K bis 500°K verarbeiten können. Was sind die entsprechenden Bereiche in Celsius und Fahrenheit? Klappt das noch mit dem von Ihnen verwendeten Datentyp?

g.) Unsere Klasse lässt somit zu, die Temperatur beliebig oft auf einen absoluten Wert zu setzen. Neu soll es zusätzlich möglich sein, die Temperatur auch um einen **relativen** Wert (in Kelvin oder Celsius, positiv oder negativ) zu verändern[1]. Implementieren Sie die entsprechenden Methoden! Testen Sie, ob danach die Temperatur (in allen Einheiten!) korrekt ist.

h.) Welcher Temperaturwert ist eigentlich vorhanden, wenn Sie ein neues **`Temperatur`**-Objekt erzeugen? Sorgen Sie dafür, dass die initiale (Default-)Temperatur 20°C ist! Hinweis: Es gibt verschiedene Möglichkeiten das zu erreichen. Was ist der einfachste Weg? Implementieren Sie!

i.) Neu möchten wir direkt bei der Erstellung eines Objektes die Möglichkeit haben, eine initiale Temperatur angeben zu können. Sie erinnern sich: Es gibt sogenannte Konstruktoren. Implementieren Sie!

k.) Können wir parallel dazu nun noch **`Temperatur`**-Objekte erzeugen, welche die Default-Temperatur 20°C haben, wenn wir keine initiale Temperatur angeben? Was müssen wir machen, damit das (ggf. wieder) funktioniert?

l.) Sie haben nun schon eine ganz hübsche Klasse entwickelt. Wir werden diese Klasse im Verlaufe des Semesters immer wieder kopieren und dann weiter verbessern und verändern, passen Sie also auf, dass Sie Ihre aktuellen Lösungen nicht verlieren!

[1] Denken Sie dabei an ein Thermostat: Egal was für eine Temperatur grad eingestellt ist, Sie hätten es einfach gerne z.B. 2°C kühler oder 5°C wärmer (also +2 oder -5).

2 Elementare Datentypen in Java

2.1 Lernziele

- Verwendung von elementaren Datentypen.
- Zwischen Wertebereich und Genauigkeit differenzieren können.
- Unterschiedliches Verhalten von Ganzzahl- und Fliesskommatypen erkennen.

2.2 Grundlagen

In dieser Aufgabe wollen wir uns durch ein paar Experimente mit den elementaren Datentypen von Java vertraut machen. Grundlage ist der Input **O03_IP_Datentypen-Operatoren**.

In dieser Übung arbeiten Sie mit BlueJ. Wir verwenden dafür das «Code Pad» (deutsch: «Direkteingabe»-Bereich), welchen Sie über das Menu unter **`View → Show Code Pad`** (deutsch: **`Ansicht → Direkteingabe anzeigen`**) oder **`Ctrl-E`** sichtbar machen können.

2.3 Aufgaben

a.) Nehmen Sie das Lehrbuch «Objects First with Java» zur Hand und lesen Sie den Anhang **B**. Besonders die darin enthaltende Tabelle mit den Wertebereichen ist für die folgenden Aufgaben sehr nützlich. Weitere Informationen finden Sie auch in der Java-Dokumentation unter https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

b.) Experimente zum Wertebereich vom Datentyp **`int`**:
- Geben Sie den maximalen positiven Wert für **`int`** in das Code Pad / den Direkteingabebereich ein und schliessen Sie mit Enter ab. Was wir angezeigt?
- Geben Sie Sie nun einen um **`+1`** erhöhten Werte ein, was passiert dann?
- Statt den zu grossen Werte absolut einzugeben, formulieren Sie ihn als Berechnung: «**`2147483647 + 1`**». Was passiert jetzt? Hätten Sie das erwartet?
- Können Sie sich erklären, warum das passiert?
- Wie lautet das Resultat, wenn Sie den minimalen Bereich überschreiten?
- Studieren Sie den folgenden Wikipedia-Eintrag zum Zweierkomplement: https://de.wikipedia.org/wiki/Zweierkomplement

c.) Experimente zum Wertebereich von **`float`** (Achten Sie darauf, dass Sie alle Werte mit angehängtem '**`f`**' eingeben, da sie sonst als **`double`** interpretiert werden!):
- Geben Sie den maximalen positiven Wert für float (**`3.4028235e38f`**) ein. Was wird angezeigt?
- Provozieren Sie auch hier mit einer Addition einen Bereichsüberlauf. Versuchen Sie es mal mit **`3.4028235e38f + 1.0f`**! Erklären Sie das Verhalten!
- Welchen Wert müssen Sie minimal addieren, dass ein Bereichsüberlauf passiert?
- Für Interessierte: https://de.wikipedia.org/wiki/IEEE_754

d.) Berechnen Sie im Eingabebereich die Division **`2 + 5 / 2`** (sollte **`4.5`** ergeben).
- Experimentieren Sie mindestens mit den Datentypen **`int`**, **`float`** und **`double`**. Denken Sie daran, dass Sie mit **`(`***`<datentyp>`***`)`** die Datentypen umwandeln (casten) können.
- In Java gilt auch «Punkt vor Strich», d.h. Multiplikation und Division haben Vorrang.
- Was müssen Sie machen (es gibt mehrere Varianten!), dass Sie das richtige Resultat **`4.5`** erhalten?

e.) Vielleicht hat Sie die eine oder andere Erfahrung/Beobachtung in dieser Aufgabe überrascht oder verunsichert. Tatsächlich ist ein unbedachter Umgang mit Datentypen einer der **häufigsten** Fehler und hat sogar schon Raketen zerstört: https://de.wikipedia.org/wiki/Ariane_V88
Beachten Sie auch, dass die in dieser Aufgabe beobachteten Effekte nicht nur bei Java, sondern in vielen Sprachen auftreten, siehe: https://0.30000000000000004.com/