Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M.Sc. Stefan Karch
M.Sc. Mariia Sukhova

07.11.2023

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 4

Bearbeitungszeitraum: 20.11.2023 – 01.12.2023

Aufgabe 1 (Pflichtaufgabe) *Zahlen raten*

Erstellen Sie ein Java-Programm, in welchem eine vom Computer erzeugte Zufallszahl im Bereich $\{1, ..., 1000\}$ erraten werden soll. Als Hilfestellung wird nach jedem Versuch angegeben, ob der Rateversuch des Benutzers zu niedrig oder zu hoch ist. Gehen Sie folgendermaßen vor:

- Erstellen Sie eine Klasse `Raten` mit einer `main`-Methode. Zunächst soll der Benutzer danach gefragt werden, wieviele Versuche er zur Verfügung haben soll. Speichern Sie die Eingabe in einer ganzzahligen Variablen `versuche`.
- Erzeugen Sie die geheime Zufallszahl.
  - Mit der Methode `Math.random()` lässt sich eine zufällige **`double`**-Zahl im Intervall $(0, 1)$ erzeugen. Durch eine anschließende Multiplikation mit 1000 erhält man also eine Gleitkommazahl im offenen Intervall $(0, 1000)$.
  - Die so erhaltene Zahl soll durch eine Typumwandlung von **`double`** zu **`int`** abgerundet werden. Ist `d` eine **`double`**-Variable, so wird sie mit `(int)d` als **`int`**-Variable interpretiert. Angewendet auf die bisher erzeugte Zahl erhält man somit eine zufällige ganze Zahl im Bereich $\{0, ..., 999\}$.
  - Addieren Sie 1 um eine zufällige ganze Zahl im Bereich $\{1, ..., 1000\}$ zu erhalten. Speichern Sie diese in einer Variablen `geheimzahl` ab.

    Hinweis: Die obigen Schritte lassen sich - durch passende Klammerung - auch in einem einzelnen Befehl durchführen.

  Erstellen Sie außerdem eine Variable `gewonnen` vom Typ **`boolean`** und weisen Sie ihr den Wert **`false`** zu.
- Lesen Sie in einer **`for`**-Schleife für $i = 1, \ldots,$`versuche` jeweils einen Rateversuch in Form einer ganzen Zahl ein und geben Sie aus, ob diese Vermutung größer oder kleiner als die gesuchte Geheimzahl ist. Falls `geheimzahl` korrekt erraten wurde soll der Benutzer darüber informiert und `gewonnen` auf **`true`** gesetzt werden. Passen Sie die Schleife an, sodass sie abbricht, wenn das Spiel gewonnen wurde.

Hinweis: Sie können die Bedingung, dass noch nicht gewonnen wurde, direkt in die Laufzeitbedingung der `for`-Schleife mit einbauen:

```
for( ... ; ... && !gewonnen ; ...)
```

- Falls der Benutzer die Geheimzahl mit der gegebenen Anzahl an Versuchen nicht erraten konnte, so soll ihm Mut für ein erneutes Spiel zugesprochen werden.

Fragen 1 *Zahlen raten*

- Welche Strategie ist wohl die beste, um dieses Spiel zu gewinnen?
- Wie viele Versuche werden maximal benötigt, um das Spiel auf jeden Fall zu gewinnen?

Aufgabe 2 *Pythagoräische Tripel*

Ein Tripel (a, b, c), welches aus drei natürlichen Zahlen a, b und c besteht, wird ein *pythagoräisches Tripel* genannt, falls

$$a^2 + b^2 = c^2 \tag{1}$$

gilt. Schreiben Sie mit Java ein Programm mit dem Namen `PythagTripel`, welches durch reines Ausprobieren alle pythagoräischen Tripel bis zu einer oberen Grenze n findet. Gesucht sind also alle Tripel (a, b, c), mit $1 \leq a \leq b \leq c \leq n$, so dass die Gleichung (1) gilt. Gehen Sie beim Lösen dieser Aufgabe wie folgt vor:

- Erstellen Sie in der Programmklasse zunächst eine `main`–Methode. Lesen Sie in dieser Methode eine Zahl n vom Typ **`int`** von der Konsole ein und speichern Sie diese ab.
- Verwenden Sie drei ineinander geschachtelte **`for`**–Schleifen, um alle möglichen Werte für a, b und c zu durchlaufen. Prüfen Sie in der innersten Schleife mit einer **`if`**-Anweisung jeweils nach, ob die Gleichung (1) erfüllt ist. Wenn ja, so geben Sie a, b und c auf der Konsole aus.
- Testen Sie Ihr Programm für verschiedene Werte von n.

Musterlösung: Die ersten drei pythagoräischen Tripel lauten $(3, 4, 5)$, $(5, 12, 13)$, $(6, 8, 10)$

Aufgabe 3 *Rundungsfehler*

In dieser Aufgabe soll das Ergebnis der Formel

$$9x^4 - y^4 + 2y^2 \tag{I}$$

für $x = 10864$ und $y = 18817$ berechnet werden. Schreiben Sie ein Java-Programm namens `Rundungsfehler` welches das Ergebnis auf verschiedene Weisen berechnet.

- Das exakte Ergebnis kann mit 64-Bit-Ganzzahlarithmetik berechnet werden (siehe Anmerkung). Definieren Sie daher für x und y zunächst Variablen vom Typ `long`. Werten Sie anschließend die Formel (I) aus, und geben Sie das Ergebnis auf der Konsole aus.
- Das Ergebnis soll nun in 64-Bit-Gleitkommaarithmetik berechnet werden. Definieren Sie daher für x und y neue Variablen vom Typ `double`. Werten Sie anschließend die Formel (I) aus, und geben Sie das Ergebnis auf der Konsole aus.
- Nach dem Assoziativgesetz kann man die Formel (I) äquivalent zu

  $$9x^4 + (-y^4 + 2y^2) \tag{II}$$

  umformen. Werten Sie die Formel (II) in 64-Bit-Gleitkommaarithmetik aus, und geben Sie das Ergebnis auf der Konsole aus.
- Nach dem Kommutativgesetz kann man die Formel (I) äquivalent zu

  $$2y^2 + 9x^4 - y^4 \tag{III}$$

  umformen. Werten Sie die Formel (III) in 64-Bit-Gleitkommaarithmetik aus, und geben Sie das Ergebnis auf der Konsole aus.
- Nach der 3. binomischen Formel ist die Formel (I) äquivalent zu

  $$(3x^2 - y^2)(3x^2 + y^2) + 2y^2 \tag{IV}$$

  Werten Sie die Formel (IV) in 64-Bit-Gleitkommaarithmetik aus, und geben Sie das Ergebnis auf der Konsole aus.

Wie erklären Sie sich die Ergebnisse in den Aufgabenteilen (b) bis (e)? Gelten Assoziativ-, Kommutativ- und Distributivgesetz für die Rechnerarithmetik?

Anmerkung: Mit dem Datentyp `long` *können ganze Zahlen mit Beträgen bis zu* $9 \cdot 10^{18}$ *exakt dargestellt werden. Die Werte von* x *und* y *können durch* $2 \cdot 10^4$ *nach oben abgeschätzt werden. Das größte Zwischenergebnis in Formel* (I) *erhält man daher für den Term* $9x^4$*. Aus der oberen Abschätzung für* x *folgt, dass dieser Term durch* $9 \cdot (2 \cdot 10^4)^4 = 1{,}44 \cdot 10^{18}$ *nach oben abgeschätzt werden kann. Das Zwischenergebnis kann also mit dem Datentyp* `long` *noch exakt dargestellt werden.*