Java05 − Lotto (3/3) − Auswertung und Sortieren

Projekt-Beschreibung: Lotto „6 aus 16“

Wir wollen als BlueJ-Projekt ein Lotto-Spiel „6 aus 16“ programmieren.

Man kann bei diesem Lotto-Spiel 6 verschiedene Zahlen von 1 bis 16 als Tipp abgeben.

Anschließend werden die 6 Gewinnzahlen als Zufallszahlen von 1 bis 16 „gezogen“.

Abschließend sollen die 6 gezogenen Zahlen und die 6 getippten Zahlen ausgegeben werden und eine Auswertung erfolgen, wie viele „Richtige“ unter den getippten Zahlen sind.

Überblick:

Lotto (1/3): Gewinnzahlen „naiv“ ziehen, Tipp abgeben, Gewinnzahlen und Tipp-Zahlen ausgeben

Lotto (2/3): Gewinnzahlen „korrekt“ ziehen, ggf. auch Kontrolle, ob der Tipp „korrekt“ ist

Lotto (3/3): Auswertung mit Ausgabe der Anzahl der „Richtigen“, ggf. Sortierung der beiden Felder

Schritt 1 − Ermittlung der Anzahl der „Richtigen“ mit Hilfe von `„auswerten()“`

Mit der Methode `„auswerten()“` soll ermittelt werden, wie viele „Richtige“ im Tipp (Methode `„tippen()“`) der getippten 6 Zahlen (von 1 bis 16) enthalten sind.

Füge den folgenden Code ganz unten in Deinen Quellcode (aber noch oberhalb der schließenden „Klassen-Klammer“) ein und ergänze ihn passend:

```
public void auswerten() {
    int anzahlRichtige = 0;
    for (int i = 0; i < 6; i++) {
        if(… ERGÄNZEN! …) {
            … ERGÄNZEN! …;
        }
    }
    System.out.println("Du hast " + anzahlRichtige + " Richtige!");
}
```

Tipp 1: In der Bedingung von `„if“` muss geprüft werden, ob der Inhalt des i-ten Feld-Elements von „getippt“ in den Gewinnzahlen enthalten ist (→ Hilfsmethode `„istInGezogen(…)“` von „Java_04_Lotto_2“).

Tipp 2: Wenn dies der Fall ist, muss die Varibale für die Anzahl der Richtigen um 1 erhöht („inkrementiert“) werden.

Schritt 2 − Vorübung zum Sortieren: Tauschen der Inhalte zweier Feld-Elemente

Wir betrachten eine (zunächst nicht ganz korrekte) Hilfsmethode zum Tauschen der Inhalte zweier Feld-Elemente.

Füge den folgenden Code ganz unten in Deinen Quellcode (aber noch oberhalb der schließenden „Klassen-Klammer“) ein:

```
public void tauschenGewinnzahlen(int pos1, int pos2) {
    gezogen[pos1] = gezogen[pos2];
    gezogen[pos2] = gezogen[pos1];
}
```

Teste diese Methode:

Übersetze die Klasse, erzeuge ein neues Objekt „lotto1“ der Klasse „Lotto“.

Führe `„ziehenGewinnzahlen()“` und `„ausgebenGewinnzahlen()“` dann jeweils einmal aus.

Es sollen nun die Inhalte der ersten beiden Feld-Elemente (Vorsicht: Index „0“ und Index „1“) vertauscht werden. Führe also `„tauschenGewinnzahlen(int pos1, int pos2)“` aus (mit 0 und 1 als Eingabe-Parameter) und danach noch einmal `„ausgebenGewinnzahlen()“`.

Was stellst Du fest?

Warum kommt es zu diesem Fehler?

Korrektes Tauschen:

Bei der oben aufgeführten Methode „`tauschenGewinnzahlen(int pos1, intpos2)`“ geht der Inhalt des Feld-Elements „gezogen[pos1]“ offenbar „verloren“.

So kann man ihn „retten“:

```
public void tauschenGewinnzahlen(int pos1, int pos2) {
    int hilf = gezogen[pos1];
    gezogen[pos1] = gezogen[pos2];
    gezogen[pos2] = hilf;
}
```

Teste diese (nun korrekte) Methode mit dem oben beschriebenen Vorgehen. Macht sie, was sie soll?

Schritt 3 – Methode „`sortierenGewinnzahlen()`“ **zum Sortieren der Gewinnzahlen**

Die Gewinnzahlen des Feldes „gezogen“ sollen der Größe nach (aufsteigend) sortiert werden.

Dazu werden zwei „verschachtelte“ for-Schleifen benötigt.

Zunächst nimmt man an, dass das erste Feld-Element (Vorsicht: Index „0“) den kleinsten Inhalt hat:

`int minPos = i;` (am Anfang gilt ja „i = 0“)

Nun wird mit der inneren for-Schleife (Zählvariable j mit Start bei j = i) geprüft, ob ein „späteres“ Feld-Element einen kleineren Wert hat. Wenn dies der Fall ist, wird der Hilfsvariable „minPos“ der Index des Feld-Elements mit dem kleineren Wert zugewiesen. Da die innere for-Schleife bis zum Ende des Feldes durchlaufen wird, wird auf diese Weise der Index des Feldes mit dem kleinsten Wert gefunden.

Die innere for-Schleife ist nun beendet. Es erfolgt nun noch ein Tausch der Inhalte der Feld-Elemente mit den Indizes „i“ (beim ersten Durchlauf: „i = 0“) und „minPos“.

Am Ende des ersten Durchlaufs der äußeren for-Schleife wird der Wert ihrer Zählvariable um 1 erhöht durch „i++“. Nun hat die Zählvariable i den 0 + 1 = 1. Es wird nun wieder angenommen, dass das Feld-Element mit dem Index i (nun: 1) den kleinsten Inhalt hat, und die innere for-Schleife „tut ihre Arbeit“, allerdings beginnend bei j = 1 (denn i hat ja nun den Wert 1).
Anschließend erfolgt wieder der Tausch der Inhalte der Feld-Elemente.

```
public void sortierenGewinnzahlen() {
    for (int i = 0; i < gezogen.length; i++) {
        int minPos = i;
        for (int j = i; j < gezogen.length; j++) {
            if(gezogen[j] < gezogen[minPos]) {
                minPos = j;
            }
        }
        int hilf = gezogen[i];
        gezogen[i] = gezogen[minPos];
        gezogen[minPos] = hilf;
    }
}
```

Teste die Methode „`sortierenGewinnzahlen()`“ zum Sortieren der Gewinnzahlen:

Übersetze die Klasse, erzeuge ein neues Objekt „lotto1“ der Klasse „Lotto“.

Führe „`ziehenGewinnzahlen()`“ und „`ausgebenGewinnzahlen()`“ dann jeweils einmal aus.

Führe nun „`sortierenGewinnzahlen()`“ und „`ausgebenGewinnzahlen()`“ jeweils einmal aus.

Schritt 4 – Methode „`sortierenTippzahlen()`“ **zum Sortieren der getippten Zahlen**

Auch die getippten Zahlen des Feldes „getippt“ sollen der Größe nach (aufsteigend) sortiert werden.

Füge eine Methode „`sortierenTippzahlen()`“ zum Sortieren der getippten Zahlen ein.

Teste diese Methode „`sortierenTippzahlen()`“ auf geeignete Weise!