Java04 − Lotto (2/3) − Gewinnzahlen korrekt ziehen

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 0 − Problem von „`ziehenGewinnzahlen1()`“

Der Test von „`ziehenGewinnzahlen1()`“ und „`ausgebenGewinnzahlen()`“ mehrmals (jeweils direkt hintereinander) zeigt:

Bei dieser Methode kann es passieren, dass dieselbe Zahl mehrmals als Gewinnzahl gezogen wird.

Das muss natürlich verhindert werden!

Schritt 1 − Hilfsmethode „`istInGezogen(int z)`“

Wir erstellen eine Hilfsmethode, mit der ermittelt werden kann, ob eine neue gezogene Zahl bereits in der Menge der gezogenen Gewinnzahlen enthalten ist.

Diese Hilfsmethode „`istInGezogen(int z)`“ soll eine Wahrheitswert als Rückgabewert haben, nämlich den Wahrheitswert „true“ als Rückgabewert ausgeben, wenn die neue Zahl bereits im Feld „gezogen[]“ enthalten ist, ansonsten den Wahrheitswert „false“.

Beim Aufruf dieser Methode wird eine ganze Zahl „`int z`“ als Parameter übergeben. Das wird bei der Methode „`ziehenGewinnzahlen()`“ dann für das korrekte Ziehen der Gewinnzahlen (ohne Doppelungen) die jeweils neue gezogene Zahl sein.

Der Methoden-Kopf ist deshalb:

```
public boolean istInGezogen(int z) {  // Methoden-Kopf
    …
}
```

Im Methoden-Rumpf muss der Wahrheitswert als lokale Variable deklariert (und benannt; hier „enthalten“) werden und zunächst mit „false“ initialisiert werden:

```
public boolean istInGezogen(int z) {  // Methoden-Kopf
    boolean enthalten = false;  // lokale boolean-Variable „enthalten“
    …
}
```

Nun muss die Prüfung erfolgen, ob die ganze Zahl **`int z`** im Feld „gezogen[]“ enthalten ist:
Es muss also für alle Feld-Elemente geprüft werden, ob `gezogen[i] == z` gilt.
Wenn dies für ein Feld-Element der Fall ist, soll der Wahrheitswert „enthalten“ auf „true“ gesetzt werden.

```
public boolean istInGezogen(int z) {  // Methoden-Kopf
    boolean enthalten = false;  // lokale boolean-Variable „enthalten“
    for(… ERGÄNZEN! …) {
        if(… ERGÄNZEN!){
            enthalten = true;
        }
    }
    …
}
```

Schließlich muss noch ein „Return-Statement“ erfolgen, denn die Methode soll ja als Rückgabe den Wert des Wahrheitswertes „enthalten“ liefern (und nicht „nichts“ wie bei `public void …`):

```
public boolean istInGezogen(int z) {  // Methoden-Kopf
    boolean enthalten = false;  // lokale boolean-Variable „enthalten“
    for(… ERGÄNZEN! …) {
        if(… ERGÄNZEN!){
            enthalten = true;
        }
    }
   return enthalten;
}
```

Schritt 2 – Test von „`istInGezogen(int z)`“

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

Führe „`ziehenGewinnzahlen1()`“ und „`ausgebenGewinnzahlen()`“ dann einmal aus.

Rufe nun auf dem Objekt „lotto1“ die neue Prüf-Methode „`istInGezogen(int z)`“ auf.
Gib als Zahl einmal (1) eine der gezogenen Zahlen ein und einmal (2) eine nicht gezogene Zahl.

Erfolgt die Ausgabe des jeweils gewünschten Rückgabewertes?
(„true“ bei (1) und „false“ bei (2))

Schritt 3 – Methode „`ziehenGewinnzahlen()`“ **zum korrekten Ziehen (ohne Doppelungen)**

Mit Hilfe von „`istInGezogen(int z)`“ kann nun beim Ziehen der Gewinnzahlen geprüft werden, ob die jeweils neu gezogene Zahl „neueZahl“ bereits Inhalt eines Feld-Elements ist.

Solange dies der Fall ist („`while(…)`“), soll eine „neue“ „neueZahl“ als Zufallszahl erzeugt werden.

Erst wenn dies nicht (mehr) der Fall ist, wird dem i-ten Feld-Element „gezogen[i]“ die neue Zahl als Wert zugeordnet.

```
public void ziehenGewinnzahlen() {
    int neueZahl;  // Deklaration der lokalen int-Variable „neueZahl“
    for(int i = 0; i < gezogen.length; i++) {
        neueZahl = zufallszahl.nextInt(16)+1;
        while(istInGezogen(neueZahl) == true) {  // Hier wird geprüft
            neueZahl = zufallszahl.nextInt(16)+1;
        }
        gezogen[i] = neueZahl;  // Zuordnung zum i-ten Feld-Element
    }
}
```

Schritt 4 – Test von „`ziehenGewinnzahlen()`“

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

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

Gibt es Doppelungen? Vielleicht ja „zufällig“ gerade nicht.

Füge zum zehnmaligen Testen auf Doppelungen die folgende Methode ein (ganz unten):

```
public void zehnMalZiehenUndAusgeben() {
    for(int i = 0; i < 10; i++) {
        ziehenGewinnzahlen();
        ausgebenGewinnzahlen();
    }
}
```

Du kannst auch „`ziehenGewinnzahlen()`“ testweise ersetzen durch „`ziehenGewinnzahlen1()`“.
Gibt es Doppelungen?

ZUSATZ-AUFGABE

Auch beim Tippen dürfen Zahlen nicht doppelt getippt werden. Wie lässt sich das verhindern?