Aufgabensammlung zu Aufgabenblatt 2

Kompetenzstufe 1

Allgemeine Informationen zu den Aufgaben:

- Die Sammlung beinhaltet Aufgaben, die in den Übungen bzw. auch zu Hause gelöst werden können.
- Fragen dazu können in den Übungen bzw. den Programmiertutorien beantwortet werden.

Calculation

Erstellen Sie eine Klasse Calculation und erweitern Sie die Methode `main`:

Implementieren Sie ein Programm, das alle Kombinationen der Werte a, b und c (jeweils startend ab 1 und kleiner gleich 30) für die Formel

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

berechnet und jede Kombination auf einer eigenen Zeile ausgibt. Die Ausgabe sollte für die oben beschriebenen Bereiche folgendermaßen aussehen:

```
All combinations in the range [1, 30]:
a = 3 b = 4 c = 5
a = 4 b = 3 c = 5
a = 5 b = 12 c = 13
a = 6 b = 8 c = 10
a = 7 b = 24 c = 25
a = 8 b = 6 c = 10
a = 8 b = 15 c = 17
a = 9 b = 12 c = 15
a = 10 b = 24 c = 26
a = 12 b = 5 c = 13
a = 12 b = 9 c = 15
a = 12 b = 16 c = 20
a = 15 b = 8 c = 17
a = 15 b = 20 c = 25
a = 16 b = 12 c = 20
a = 18 b = 24 c = 30
a = 20 b = 15 c = 25
a = 20 b = 21 c = 29
a = 21 b = 20 c = 29
a = 24 b = 7 c = 25
a = 24 b = 10 c = 26
a = 24 b = 18 c = 30
```

Hinweis: Falls Sie drei Schleifen für das Programm benötigt haben, dann überlegen Sie sich, wie Sie die obige Aufgabe auch nur mit zwei Schleifen lösen können!

Maximum

Erstellen Sie eine Klasse Maximum:

- Implementieren Sie eine Methode `maximumOfThree`:

  ```
  int maximumOfThree(int a, int b, int c)
  ```

  Diese Methode liefert das Maximum der drei Zahlen `a`, `b` und `c` zurück.

- Implementieren Sie eine Methode `maximumOfFour`:

  ```
  int maximumOfFour(int a, int b, int c, int d)
  ```

  Diese Methode liefert das Maximum der vier Zahlen `a`, `b`, `c` und `d` zurück.

Implementieren Sie beide Methoden selbständig aus, d.h. benutzen Sie keine Methoden aus der Java-API. Sie dürfen aber eigene Implementierungen wiederverwenden. Testen Sie die Methoden durch mehrere Aufrufe in der Methode `main` aus!

Strings

Erstellen Sie eine Klasse Strings:

- Implementieren Sie eine Methode `generateString`:

  ```
  String generateString(String first, String second)
  ```

  Diese Methode gibt einen String zurück, der folgendermaßen gebildet wird: Es wird der längere der beiden Strings `first` und `second` mit dem kürzeren konkateniert. Sind beide gleich lang, dann wird der String `"Teststring"` zurückgegeben.

  Beispiel(e) für Tests:
  `generateString("Hello", "EP1")` liefert `HelloEP1` zurück
  `generateString("Hello", "World!")` liefert `World!Hello` zurück
  `generateString("Hello", "World")` liefert `Teststring` zurück

  Vorbedingung: `first != null` und `second != null`.

- Implementieren Sie eine Methode `mixStrings`:

  ```
  String mixStrings(String first, String second)
  ```

  Diese Methode erzeugt einen neuen String und gibt diesen zurück. Der Rückgabestring wird folgendermaßen erzeugt: Zuerst wird das erste Zeichen von `first` und danach das erste Zeichen von `second` übernommen. Danach kommt das zweite Zeichen von `first` und danach das zweite Zeichen von `second` usw.

  Beispiel(e) für Tests:
  `mixStrings("ABCDE", "123")` liefert `A1B2C3DE` zurück
  `mixStrings("ABC", "1234567")` liefert `A1B2C34567` zurück
  `mixStrings("AB", "12")` liefert `A1B2` zurück
  `mixStrings("", "")` liefert einen leeren String zurück

  Vorbedingung: `first != null` und `second != null`.

ChessPattern

Erstellen Sie eine Klasse ChessPattern und erweitern Sie die Methode `main`:

- Erstellen Sie ein quadratisches Fenster mit einer Seitengröße von 400 Pixel.
- Zeichnen Sie ein Schachbrett mit Feldern der Größe 50.
- Das Feld links oben ist weiß eingefärbt.
- Platzieren Sie auf jedem fünften Feld (beginnend von links oben nach rechts unten) eine Spielfigur.
- Symbolisieren Sie diese Figur mit einem Kreis (mit Palette.GOLD eingefärbt), der sich in der Mitte des jeweiligen Quadrats befindet. Der Radius des Kreises entspricht einem Viertel der Seitenlänge eines weißen bzw. schwarzen Quadrates.

Ausgabe: