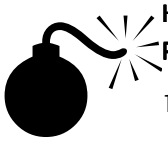

Halten Sie beim Programmieren rigoros ALLE aufgestellten Programmierregeln ein. Ihre Programme werden in Hinsicht auf die Einhaltung dieser Regeln kontrolliert.

1. Die Methode playSound wurde überladen und gibt die ihr übergebene Audio-Datei einmal oder auch mehrmals wieder. Funktionieren folgende Methodenaufrufe:

```
playSound("file:///c:\\windows\\media\\ringin.wav", 4);
playSound(4, "file:///c:\\windows\\media\\ringin.wav");
playSound(4);
playSound("file:///c:\\windows\\media\\ringin.wav", 4.0);
```

```
public static void playSound(String datei) {
  try {
     java.net.URL url = new java.net.URL(datei);
     java.applet.AudioClip clip = java.applet.Applet.newAudioClip(url);
     clip.play();
     try {
        Thread.sleep(1500);
     } catch (InterruptedException e) {
     }
  } catch (java.net.MalformedURLException e) {
     System.out.println(e.toString());
  }
}

public static void playSound(String datei, int repeat) {
  for (int i = 0; i < repeat; i = i + 1)
     playSound(datei);
}

public static void playSound(int repeat) {
  playSound("file:///c:\\windows\\media\\Windows XP-Start.wav", repeat);
}
```

2. Welche *Ausgabe* liefert folgendes Programm:

```
public class TesteMich
{
   public static void main(String[] args) {
      int y = 1;
      System.out.println(y);
      y = undJetzt(y);
      System.out.println(y);
   }

   public static int undJetzt(int x) {
      int ret = x + 2;
      return ret;
   }
}
```

3. Welche *Ausgabe* liefert folgendes Programm? Zeichnen Sie sich während der Programmausführung den jeweiligen Zustand des *Programmstapels* auf:

```
public class HaarigeSache
{
   public static void main(String[] args) {
      int a = 3; int b = 6;
      einHaar(a, b); zweiHaar(b, a);
   }

   public static void einHaar(int x, int y) {
      int a = x + y;
      System.out.println(a + " " + (int)(a - x) + " "  + (int)(a - y));
   }

   public static void zweiHaar(int x, int y) {
      unterHaar(x); unterHaar(y);
      int a = x + y; int b = 3 + x;
      einHaar(a, b);
   }

   public static void unterHaar(int a) {
      int x = 6;
      a = a + x;
      System.out.println(a + " " + x);
   }
}
```

4. Im folgenden Programm wird eine Methode in sich selbst aufgerufen (*rekursiver Aufruf*). Welche *Ausgabe* liefert das Programm? Zeichnen Sie sich während der Programmausführung den jeweiligen Zustand des *Programmstapels* auf:

```
public class Zaehlen
{
   public static void main(String[] args) {
      schrittFuerSchritt(5);
   }

   public static void schrittFuerSchritt(int x) {
      System.out.println(x);
      if (x > 0)
         schrittFuerSchritt(x - 1);
   }
}
```

5. Im obigen Programm wird die *Reihenfolge* der Befehle geringfügig geändert. Ändert sich die *Ausgabe*? Begründen Sie Ihre Antwort…

```
public class Zaehlen
{
   public static void main(String[] args) {
      schrittFuerSchritt(5);
   }

   public static void schrittFuerSchritt(int x) {
      if (x > 0)
         schrittFuerSchritt(x - 1);
      System.out.println(x);
   }
}
```

6. Schreiben Sie eine rekursive Methode mit dem Namen potenz, welche Ihnen die Potenz einer Zahl x berechnet. Gehen Sie davon aus, dass nur Potenzen von positiven, ganzen Zahlen $y \geq 0$ berechnet werden sollen. Die Potenz ist rekursiv folgendermaßen definiert:

$$x^y = \begin{cases} 1 \quad \text{für} \quad y = 0 \\ x \cdot x^{y-1} \text{ für } y \geq 1 \end{cases}$$

Legen Sie zuerst den *Methodenkopf* mit der *Parameterliste* und den *Rückgabewert* fest.

7. Die bereitgestellte Klasse `MeinStringAnalysierer` enthält einige *Methodendeklarationen* und *-beschreibungen*. Ergänzen Sie die Klasse indem Sie leere Methodenrümpfe mit notwendigen Anweisungen erstellen, so dass sich diese *compilieren* lässt. Dabei sollen die Methoden noch nicht vollständig ausprogrammiert werden. Halten Sie dabei die aufgestellten *Programmierregeln* ein.

Programmieren Sie dann der Reihe nach Methode für Methode und lassen Sie sich dabei von den Kommentaren helfen. Die Methoden sollen exakt laut Anleitung funktionieren. Wenn Sie die erste Methode fertig programmiert haben, sollen Sie diese anhand des bereitgestellten Testprogramm mit dem Namen `TestStringAnalysierer` testen.

WICHTIG: Nachdem Sie eine Methode programmiert haben, sollen Sie sofort an das Testen der Methode gehen. Erst nachdem die Methode verlässlich funktioniert, sollen Sie die nächste Methode programmieren und testen.

8. Die bereitgestellte Klasse `MeinZahlensystemwandler` enthält einige *Methodendeklarationen* und *-beschreibungen* welche zur Umrechnung von Zahlen aus Zahlensystemen dienen. Ergänzen Sie die Klasse indem Sie leere Methodenrümpfe mit notwendigen Anweisungen erstellen, so dass sich diese *compilieren* lässt. Dabei sollen die Methoden noch nicht vollständig ausprogrammiert werden. Halten Sie dabei die aufgestellten *Programmierregeln* ein.

Programmieren Sie dann der Reihe nach Methode für Methode und lassen Sie sich dabei von den Kommentaren helfen. Die Methoden sollen exakt laut Anleitung funktionieren. Wenn Sie die erste Methode fertig programmiert haben, sollen Sie diese anhand eines selbst geschriebenen Testprogramms mit dem Namen `TestZahlensystemwandler` testen.

WICHTIG: Wiederum sollen Sie jede einzelne Methode nach ihrer Programmierung sofort eingehend testen.

9. Erweitern Sie dann die Klasse `MeinZahlensystemwandler` um folgende *Methoden*, definieren Sie *Parameterlisten* und *Rückgabewerte*, und *kommentieren* Sie die Methoden bevor Sie diese *ausprogrammieren* und *testen*:

`hexToDec`	Umwandlung von Hexadezimal nach Dezimal
`decToHex`	Umwandlung von Dezimal nach Hexadezimal
`dualToDec`	Umwandlung von Dual nach Dezimal
`decToDual`	Umwandlung von Dezimal nach Dual
`numToNum`	Umwandlung von einem beliebigen Zahlensystem in ein anderes beliebiges Zahlensystem

   WICHTIG: In den Methoden dürfen *keine* Bildschirmein- oder -ausgaben erfolgen.

10. Sie sollen das Spiel *Mastermind* programmieren wobei *keine doppelten Farben* im Spiel vorkommen dürfen. Der Computer soll sich *zufällig* einen Code generieren, den der Benutzer des Programmes erraten soll.

    Sie sollen das Programm erstellen, indem Sie zuerst folgende *Methoden* programmieren. Die Anforderungen an die Methoden werden nachfolgend beschrieben. Ihre Aufgabe besteht zuerst darin, die *Methodenköpfe* mit *Parameterliste* und *Rückgabewert* exakt zu definieren, die Methodenköpfe zu *dokumentieren*, die einzelnen Methoden *auszuprogrammieren* und diese sofort zu *testen*. Erst nachdem Sie sicher sind, dass alle Methoden wie gewünscht funktionieren, sollen Sie das Programm selbst erstellen. Das Programm soll den Namen `Mastermind` haben. Methoden und Programm sollen in derselben Klasse ausprogrammiert werden.

    Methode `erzeugeCode`
    Dieser Methode wird die *Anzahl der Stellen* des Codes und die *Anzahl der Farben* übergeben. Sie liefert den *zufällig ermittelten Code* zurück, wobei anstelle von Farben Großbuchstaben von `A` beginnend zurückgeliefert werden. Die Methode sorgt dafür, dass im Code *keine doppelten Farben* bzw. *Buchstaben* vorhanden sind. Die Methode liefert nur dann ein Ergebnis *ungleich* **`null`** zurück, falls die Farbenanzahl größer oder gleich der Stellenanzahl ist. Die Methode darf *keine Bildschirmausgabe* machen. Wird der Methode als *Stellenanzahl* 4 und als *Farbenanzahl* 6 übergeben, so kann sie beispielsweise den Text "ACFD" zurück liefern.

    Methode `enthaeltDoppelte`
    Dieser Methode wird ein *String* übergeben, und sie kontrolliert, ob im String *Buchstaben doppelt vorkommen*. Ist das der Fall, so wird **`true`** zurück geliefert. Die Methode liefert immer dann **`false`** zurück, falls der übergebene String **`null`** oder die Länge des Strings *gleich 0* ist. Bei der Kontrolle auf doppelte Buchstaben wird die *Groß-/Kleinschreibung nicht beachtet*. In der Methode selbst dürfen *keine Bildschirmausgaben* erfolgen. Wird der Methode beispielsweise der String "ACFD" übergeben so wird **`false`** zurück geliefert, wird "ACAD" übergeben, so wird **`true`** zurück geliefert.

    Methode `eingabeTipp`
    Durch diese Methode wird dem Benutzer die Möglichkeit gegeben, seinen Tipp einzugeben. Wie *lang der einzugebende Tipp* sein darf, wird der Methode übergeben. Der vom Benutzer eingegebene Tipp wird von der Methode zurück geliefert. Innerhalb der Methode wird der Text "`Ihr Tipp:`" ausgegeben. Die Eingabe muss in der Methode *automatisch wiederholt* werden, falls der *Benutzer keinen Tipp eingibt*, der Tipp nicht die *geforderte Länge* hat und falls *doppelte Buchstaben* im Tipp vorhanden sind. Gibt der Benutzer den Text "ende" ein, so muss die *Eingabe abgebrochen* und der eingegebene Text zurück geliefert werden. Der eingegebene Text muss in *Großbuchstaben konvertiert* und zurückgegeben werden.

Methode `ermittleSchwarz`

Anhand dieser Methode soll ermittelt werden wie viele Buchstaben sich am *richtigen Platz befinden*. Dazu werden der Methode der zu *erratende Code* und der *Tipp* übergeben. Die Methode liefert die *Anzahl der Richtigen* zurück. Der übergebene Code und der Tipp müssen gefüllt sein und dieselbe Länge haben, ansonsten wird -1 zurück geliefert. Wird der Methode beispielsweise "ABCD" und "BACF" übergeben, so liefert sie 1 zurück.

Methode `ermittleWeiss`

Diese Methode soll die Anzahl der richtigen Buchstaben, die sich noch am *falschen Platz* befinden, ermitteln. Dabei werden wiederum der zu *erratende Code* und der *Tipp* übergeben. Die Methode liefert wiederum die *Anzahl der Richtigen* Buchstaben zurück. Der übergebene Code und der Tipp müssen gefüllt sein und dieselbe Länge haben, ansonsten wird -1 zurück geliefert. Wird der Methode beispielsweise "ABCD" und "BACF" übergeben, so liefert sie 2 zurück.

Nachdem Sie die obigen Methoden *kommentiert*, *ausprogrammiert* und *eingehend getestet* haben, sollen Sie sich an die Programmierung des Programmes machen. Dieses soll die folgende Benutzerschnittstelle haben.

```
Mastermind
==========
Anzahl Farben: 6
======================> Ihr Tipp: abcd
1): ABCD = (w: 1, s: 2): Ihr Tipp: acef
2): ACEF = (w: 1, s: 2): Ihr Tipp: acfd
3): ACFD = (w: 3, s: 1): Ihr Tipp: acdf
4): ACDF = (w: 2, s: 2): Ihr Tipp: adcf
5): ADCF = (w: 0, s: 4): Code gefunden
======================> Ihr Tipp: bcde
1): BCDE = (w: 2, s: 0): Ihr Tipp: ende
2): ENDE = (w: 1, s: 0): Ende
```