Prof. Dr. M. Müller C. Aschermann, J. Hensel, J. Protze, P. Reble

Tutoraufgabe 1 (Typen):

Bestimmen Sie den Typ und das Ergebnis der folgenden Java-Ausdrücke und begründen Sie Ihre Antwort. Sollte der Ausdruck nicht typkorrekt sein, begründen Sie, worin der Fehler besteht.
Dabei seien die Variablen `x`, `y` und `z` wie folgt deklariert: `int x = 1; int y = 2; int z = 3;`

a) `false && true`

b) `10 / 3`

c) `10 / 3.`

d) `x == y ? x > y : y < z`

e) `(byte) (127 + 1)`

f) `'x' + y + z`

g) `x + y + "z"`

h) `1 || 0`

Lösung:

`int x = 1; int y = 2; int z = 3;`

a) `false && true`
Der Ausdruck liefert den Wert `false` vom Typ `boolean`, da die logische Und-Verknüpfung zweier `boolean` Werte hier ganz normal ausgeführt werden kann.

b) `10 / 3`
Der Ausdruck liefert den `int`-Wert `3`, da bei der Division zweier `int`-Werte in Java Ganzzahldivision ohne Rest verwendet wird.

c) `10 / 3.`
Der Ausdruck liefert den `double`-Wert `3.3333333333333335`, da der `int`-Wert `10` für die `double`-Division erst zu `double` konvertiert wird.

d) `x == y ? x > y : y < z`
Der Ausdruck liefert den `boolean`-Wert `true`, da zuerst der `boolean`-Vergleich `x == y` zu `false` und anschließend `y < z` zu `true` ausgewertet wird. Der Typ von `x > y` ist ebenfalls `boolean`, weshalb kein Fehler auftritt.

e) `(byte) (127 + 1)`
Der Ausdruck liefert das Ergebnis `-128`, da zuerst die `int`-Addition durchgeführt wird und das Ergebnis `+128` anschließend in den `byte`-Datentypen konvertiert wird. Dieser Datentyp kann diesen Wert allerdings nicht darstellen. Daher werden nur die letzten 8 Bit berücksichtigt (alle zusätzlichen Bits werden also „abgeschnitten“; dies wird auch „Overflow“ oder auf Deutsch Überlauf genannt).

f) `'x' + y + z`
Durch die Auswertung von links nach rechts wird zuerst `'x' + y` ausgewertet. Dafür wird das Zeichen `'x'` zuerst in die `int`-Zahl 120 konvertiert. Dies ergibt also `120 + 2 = 122`. Dieser Wert wird dann mit der `int`-Zahl `3` addiert, und somit wird der Gesamtausdruck zu `125` vom Typ `int` ausgewertet.

g) `x + y + "z"`
Durch die Auswertung von links nach rechts wird zuerst `x + y` zur `int`-Zahl `3` ausgewertet. Dieser Wert wird dann mit dem `String "z"` verkettet, und somit wird der Gesamtausdruck zu `"3z"` vom Typ `String` ausgewertet.

h) `1 || 0`
Der Ausdruck liefert einen Fehler, da `1` und `0` vom Typ `int` sind und damit der `boolean`-Vergleich nicht möglich ist.

Aufgabe 2 (Typen): (4 Punkte)

Bestimmen Sie den Typ und das Ergebnis der folgenden Java-Ausdrücke und begründen Sie Ihre Antwort. Sollte der Ausdruck nicht typkorrekt sein, begründen Sie, worin der Fehler besteht.
Dabei seien die Variablen `x`, `y` und `z` wie folgt deklariert: `int x = 1; int y = 2; int z = 122;`

a) `2147483600 + '2'`

b) `(double) (3 / 2)`

c) `"x" + y + z`

d) `x + y > z && true`

e) `9 / 4.5`

f) `"x" - z`

g) `17 + 0f`

h) `z != 'z'? 1f : 2`

Lösung:

`int x = 1; int y = 2; int z = 122;`

a) `2147483600 + '2'`
Der Ausdruck liefert `-2147483646` vom Typ `int`. Der `char`-Wert `'2'` wird zunächst zum `int`-Wert `50` konvertiert. Die anschließende Addition führt zu einem Überlauf mit dem genannten negativen Ergebnis.

b) `(double) (3 / 2)`
Der Ausdruck liefert `1.0` vom Typ `double`, da zuerst eine Ganzzahldivision durchgeführt wird und erst anschließend die Konversion erfolgt.

c) `"x" + y + z`
Der Ausdruck liefert `"x2122"` vom Typ `String`. Von links nach rechts wird zunächst `"x" + y` berechnet und `y` zu dem `String` `"2"` konvertiert. Das Zwischenergebnis `"x2"` wird anschließend analog mit dem konvertierten `String` `"122"` verkettet.

d) `x + y > z && true`
Der Ausdruck liefert den Wert `false` vom Typ `boolean`, da die logische Und-Verknüpfung zweier `boolean` Werte hier ganz normal ausgeführt werden kann und 3 nicht größer ist als 122.

e) `9 / 4.5`
Der Ausdruck liefert `2.0` vom Typ `double`, da der Dividend `9` erst in einen `double`-Wert konvertiert wird und anschließend eine `double`-Division durchgeführt wird.

f) `"x" - z`
Der Ausdruck liefert einen Fehler, da der Operator `-` nicht auf `String`s arbeitet und ein `String` auch nicht implizit zu `int` konvertiert werden kann.

g) `17 + 0f`
Der Ausdruck liefert `17.0f` vom Typ `float`, da der erste Operand `17` erst in einen `float`-Wert konvertiert und anschließend eine `float`-Addition durchgeführt wird.

h) `z != 'z'? 1f : 2`
Der Ausdruck liefert `2.0f` vom Typ `float`. Zur Auswertung des booleschen Ausdrucks wird zunächst der `char`-Wert `'z'` zum `int`-Wert `122` konvertiert. Der Vergleich schlägt also fehl, da der Wert von `z` gleich `122` ist. Demnach wird die zweite Alternative zurückgegeben. Da bei primitiven Datentypen aber die beiden Alternativen den gleichen Typ haben müssen, wird der Ausdruck `2` zunächst noch zum `float`-Wert `2.0f` konvertiert.

Tutoraufgabe 3 (Zweierkomplement):

a) Erklären Sie im Detail, wie die beiden Ausgaben des folgenden Programms berechnet werden.

```
public class Test {
    public static void main(String[] args) {
        int zahl = -2147483648;

        System.out.println(zahl + 1);
        System.out.println(zahl - 1);
    }
}
```

Hinweis: $-2^{31} = -2147483648$

b) Welche Zahlen repräsentieren die folgenden Bitfolgen im 5-Bit Zweierkomplement?

00010 10111 11011 01101 10000

c) Sei x eine ganze Zahl. Wie unterscheiden sich die Zweierkomplement-Darstellungen von x und $-x$?

Lösung:

a) Im Folgenden werden Binärzahlen mit einem Z markiert, wenn die Zahl im Zweierkomplement verstanden werden muss. Die Zahl $1111Z$ ist also als -1 zu verstehen, während 1111 für die Zahl 15 steht.

Der Datentyp `int` benutzt 32 Bit. Die Darstellung der Zahl −2147483648 im Zweierkomplement ist:

```
10000000000000000000000000000000 Z (31 Nullen)
```

Das Ergebnis der Addition `zahl + 1` berechnet sich wie folgt:

```
10000000000000000000000000000000 Z
00000000000000000000000000000001 Z
--------------------------------
10000000000000000000000000000001 Z
```

Auch hier gibt die führende 1 an, dass die dargestellte Zahl negativ ist. Den Dezimalwert der dargestellten Zahl erhält man durch Invertieren und Addieren von 1:

```
01111111111111111111111111111110
00000000000000000000000000000001
--------------------------------
01111111111111111111111111111111
```

Dies steht für 2147483647. Mit der Vorzeicheninformation von oben ergibt sich -2147483647.

Berechnet man `zahl - 1`, berechnet sich das Ergebnis durch die Addition mit -1. Die Zahl -1 ist im Zweierkomplement dargestellt durch:

```
11111111111111111111111111111111 Z
```

Die Addition -2147483648 + (-1) ergibt demzufolge:

```
10000000000000000000000000000000 Z
11111111111111111111111111111111 Z
--------------------------------
01111111111111111111111111111111 Z
```

Das Ergebnis ist also nicht negativ (erkennbar durch die führende 0) und entspricht der Dezimalzahl +2147483647. Dieses Ergebnis wird auch durch das Java-Programm ausgegeben.

b)

Bitfolge	5-Bit Zweierkomplement
00010	2
10111	-9
11011	-5
01101	13
10000	-16

c) Ausgehend von der Zweierkomplement-Darstellung von x erreicht man durch die folgenden beiden Schritte die Zweierkomplement-Darstellung von $-x$:

a) vertausche alle 0en und 1en

b) addiere 1

Mit diesen beiden Schritten ist auch die Rückrichtung ($-x$ zu x) möglich.

In der folgenden Tabelle finden Sie alle Binärzahlen mit drei Ziffern. Man erkennt das Muster, nach dem das genannte Verfahren funktioniert.

3	011
2	010
1	001
0	000
-1	111
-2	110
-3	101
-4	100

Aufgabe 4 (Zweierkomplement): (2+2 Punkte)

a) Welche Zahlen repräsentieren die folgenden Bitfolgen im 10-Bit Zweierkomplement?

0100000101 1111010110 0001011011 1000010000

b) Die zwei folgenden Java-Ausdrücke werten jeweils zu `true` aus, obwohl dies auf den ersten Blick nicht offensichtlich erscheint. Geben Sie dafür jeweils eine kurze informelle Begründung an.

i) $2\,000\,000\,000 + 1\,000\,000\,000 < 2\,000\,000\,000$

ii) $-(-2\,147\,483\,648) == -2\,147\,483\,648$

Lösung:

a)

Bitfolge	10-Bit Zweierkomplement
0100000101	261
1111010110	-42
0001011011	91
1000010000	-496

b) i) Bei der Addition von 2 000 000 000 und 1 000 000 000 werden intern die Zweierkomplement-Darstellungen mittels normaler Binärzahl-Rechnung addiert. Hierbei ist das Ergebnis zu groß, um mit 32 Stellen dargestellt zu werden (ein Überlauf passiert). Das zusätzliche Bit wird ignoriert und das restliche Ergebnis wird als Zahl im Zweierkomplement interpretiert. Bei dieser konkreten Berechnung ergibt sich eine Binärzahl, bei der das vorderste Bit eine 1 ist, weshalb das Ergebnis der Berechnung eine negative Zahl ist. Diese ist offensichtlich kleiner als 2 000 000 000.

ii) Der Wert in der Klammer ist -2147483648, also -2^{31}. Bei der Berechnung von $-(-2147483648)$ wird der entsprechende Bitvektor $10\ldots0$ zunächst invertiert (was $01\ldots1$ ergibt) und dann noch 1 auf diesen Bitvektor addiert (was den Bitvektor $10\ldots0$ ergibt). Der entstandene Bitvektor ist genau derselbe, von dem wir ursprünglich ausgegangen sind.

Tutoraufgabe 5 (Input):

Schreiben Sie ein einfaches Java-Programm, welches den Benutzer auffordert, eine positive Zahl ganze Zahl (d. h. größer als 0) einzugeben. Danach soll das Programm eine durch die Return/Enter-Taste beendete Zahl einlesen. Diese Eingabeaufforderung mit anschließendem Einlesen soll solange wiederholt werden, bis der Benutzer eine positive Zahl eingibt. Anschließend soll der Benutzer aufgefordert werden, ein Wort einzugeben. Dieses soll ebenfalls durch die Return/Enter-Taste beendet werden. Das Wort soll eingelesen und schließlich so oft hintereinander in einer Zeile ausgeben werden, wie durch die eingegebene positive Zahl festgelegt wurde.

Lösung:

```
/**
 * Programm zum Einlesen einer positiven Zahl und eines Worts, welches das Wort
 * anschliessend so oft ausgibt, wie durch die Zahl festgelegt wurde.
 */
public class Multiecho {

    public static void main(String[] args) {
        // Einlesen der Zahl mit Ueberpruefung, dass die Zahl positiv ist:
        int zahl = 0;
        while (zahl < 1) {
            System.out.println("Bitte geben Sie eine positive Zahl ein:");
            zahl = Integer.parseInt(System.console().readLine());
        }
        // Einlesen des Wortes:
        System.out.println("Bitte geben Sie ein Wort ein:");
        String wort = System.console().readLine();
        // Ausgabe des Wortes so oft wie durch die Zahl festgelegt wurde:
        int i = 0;
        while (i < zahl) {
            System.out.print(wort);
            i++;
        }
    }

}
```

Aufgabe 6 (Arithmetisches Mittel): (5 Punkte)

Schreiben Sie ein einfaches Java-Programm zur Berechnung des arithmetischen Mittels einer Menge ganzer Zahlen. Der Benutzer soll zuerst nach der Anzahl der Werte, von denen das Mittel berechnet werden soll, gefragt werden. Alle Eingaben des Benutzers sollen mit der Return/Enter-Taste beendet werden. Die Aufforderung zur Eingabe soll so lange wiederholt werden, bis der Benutzer eine positive Zahl ≥ 1 eingibt. Anschließend sollen entsprechend viele Integer-Werte eingelesen werden, deren arithmetisches Mittel am Ende ausgegeben wird.

Die Ausgabe eines beispielhaften Ablaufs des Programms könnte wie folgt aussehen:

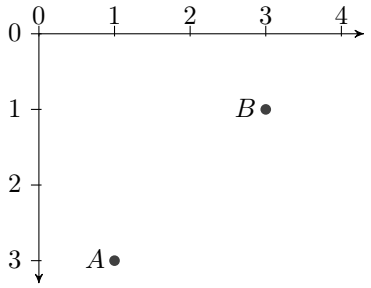
```
Bitte geben Sie die Anzahl der Werte an, fuer die Sie das
arithmetische Mittel berechnen wollen: 0
Die eingegebene Zahl ist nicht positiv!
Bitte geben Sie die Anzahl der Werte an, fuer die Sie das
arithmetische Mittel berechnen wollen: 3
Bitte geben Sie Wert 1 ein:
17
Bitte geben Sie Wert 2 ein:
4
Bitte geben Sie Wert 3 ein:
11
Das arithmetische Mittel betraegt 10.666666666666666
```

Lösung:

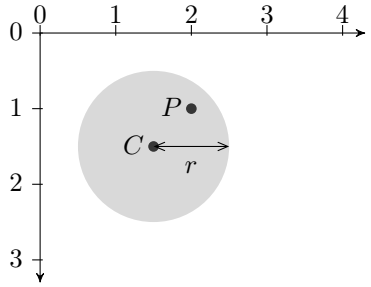
```
public class Arith {

    public static void main(String[] args) {
        // Einlesen der Anzahl:
        System.out.print(
            "Bitte geben Sie die Anzahl der Werte an, fuer die Sie das "
            + "arithmetische Mittel berechnen wollen: "
        );
        int anzahlInts = Integer.parseInt(System.console().readLine());
        while (anzahlInts < 1) {
            System.out.println(
                "Die eingegebene Zahl ist nicht positiv!"
            );
            System.out.print(
                "Bitte geben Sie die Anzahl der Werte an, fuer die Sie das "
                + "arithmetische Mittel berechnen wollen: "
            );
            anzahlInts = Integer.parseInt(System.console().readLine());
        }
        // Einlesen der einzelnen Werte:
        int i = 1;
        int summe = 0;
        while (i <= anzahlInts) {
            System.out.println("Bitte geben Sie Wert " + i + " ein: ");
            summe = summe + Integer.parseInt(System.console().readLine());
            i++;
        }
        double arithMittel = ((double) summe) / anzahlInts;
        System.out.println("Das arithmetische Mittel betraegt " + arithMittel);
    }

}
```

Tutoraufgabe 7 (Das Herz):

In dieser Aufgabe sollen Sie ein Programm schreiben, das auf der Konsole ein Herz ausgibt. Im Folgenden betrachten wir ein Koordinatensystem, in dem der Ursprung links oben ist. Betrachten Sie zur Illustration das folgende Bild, in dem der Punkt A an $(1, 3)$ und der Punkt B an $(3, 1)$ liegt:



a) Wir wollen nun untersuchen, ob ein Punkt P mit den Koordinaten (p_x, p_y) in einem Kreis liegt, den wir mit Hilfe eines Punktes C (an Position (c_x, c_y)) und eines Radius r angeben. Dies wird im folgenden Schaubild für $P = (2, 1)$, $C = (1.5, 1.5)$ und $r = 1$ illustriert:



Ein Punkt (p_x, p_y) liegt dann in einem Kreis, wenn sein Abstand vom Mittelpunkt (c_x, c_y) kleiner oder gleich dem Radius r ist (d. h. Punkte auf dem Rand des Kreises zählen als innerhalb des Kreises liegend).

Vervollständigen Sie nun die folgende Funktion so, dass `true` zurückgegeben wird, wenn (`px`, `py`) im vom Mittelpunkt (`cx`, `cy`) und Radius `r` definierten Kreis liegt und `false` in allen anderen Fällen:

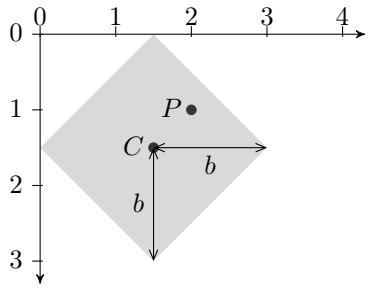
```
public static boolean inKreis(double px, double py, double cx,
                              double cy, double r) {
  return...
}
```

Sie dürfen dabei annehmen, dass der Radius `r` eine positive Zahl ist.

Hinweise:

- Die Anweisung "`return` *expr*" wertet den Ausdruck *expr* aus und setzt das Ergebnis an der Aufrufstelle der Funktion ein.
- Verwenden Sie die Funktion `Math.sqrt(double a)`, die $\sqrt[2]{\texttt{a}}$ als `double` zurückgibt.

b) Wir wollen nun untersuchen, ob ein Punkt P mit den Koordinaten (p_x, p_y) in einer quadratischen Raute liegt, die wir mit Hilfe eines Punktes C (an Position (c_x, c_y)) und einer Rautenbreite b definieren. Dies wird im folgenden Schaubild für $P = (2, 1)$, $C = (1.5, 1.5)$ und $b = 1.5$ illustriert:



Ein Punkt (p_x, p_y) liegt dann in einer Raute um (c_x, c_y), wenn die Summe der Differenzen $|p_x - c_x| + |p_y - c_y|$ kleiner oder gleich der Rautenbreite b ist (d. h. Punkte auf dem Rand der Raute zählen als innerhalb der Raute liegend).

Vervollständigen Sie nun die folgende Funktion so, dass `true` zurückgegeben wird, wenn (`px`, `py`) in der Raute um den Mittelpunkt (`cx`, `cy`) und Rautenbreite `b` liegt und `false` in allen anderen Fällen:

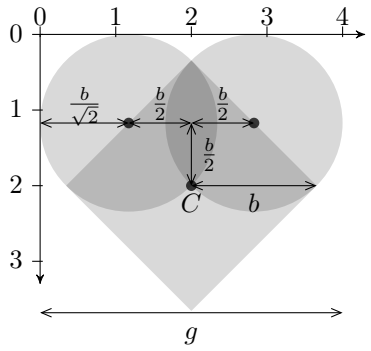
```
public static boolean inRaute(double px, double py, double cx,
                              double cy, double b) {
  return ...
}
```

Sie dürfen dabei annehmen, dass die Rautenbreite `b` eine positive Zahl ist.

Hinweise:

- Verwenden Sie die Funktion `Math.abs(double a)`, die |`a`| als `double` zurückgibt.

c) Wir wollen nun untersuchen, ob ein Punkt in einem Herz liegt. Betrachten Sie zuerst das folgende Schaubild:



Ein Punkt liegt dann in einem Herz mit Gesamtbreite g und Mittelpunkt $C = (c_x, c_y)$, wenn er in einer Raute mit Rautenbreite $b = \frac{g}{1+\frac{2}{\sqrt{2}}}$ um C, in einem Kreis mit Radius $\frac{b}{\sqrt{2}}$ um $(c_x - \frac{b}{2}, c_y - \frac{b}{2})$ oder in einem Kreis mit dem gleichen Radius um $(c_x + \frac{b}{2}, c_y - \frac{b}{2})$ liegt. Im Beispiel ist $g = 4$ gewählt worden.

Verwenden Sie nun die Funktionen `inKreis` aus Teilaufgabe a) und `inRaute` aus Teilaufgabe b), um die folgende Funktion zu implementieren, die genau dann `true` zurückgibt, wenn ein Punkt (`px`, `py`) in einem durch den Mittelpunkt (`cx`, `cy`) und die Gesamtbreite `g` beschriebenen Herzen liegt:

```
public static boolean inHerz(double px, double py, double cx,
                             double cy, double g) {
  return ...
}
```

Sie dürfen dabei annehmen, dass die Gesamtbreite `g` eine positive Zahl ist.

Hinweise:

- Verwenden Sie die boolesche Verknüpfung `||`, um die verschiedenen Bedingungen miteinander zu verbinden.

d) Sie sollen nun die Funktion `inHerz` aus Aufgabenteil c) verwenden, um ein Herz auf der Konsole auszugeben.

Schreiben Sie nun ein Programm, das

1. den Benutzer nach der Gesamtbreite g für das Herz fragt und eine Integer-Zahl einliest. Bestimmen Sie daraus $(\frac{g}{2}, \frac{g}{2})$ als Mittelpunkt für das Herz. Zur Vereinfachung dürfen Sie annehmen, dass der Benutzer hier eine positive ganze Zahl eingibt.
2. mit Hilfe zweier Schleifen alle Koordinaten $(0,0), \ldots, (0,g), (1,0), \ldots (1,g), \ldots (g,g)$ abläuft und ein "`#`" ausgibt, wenn sich die Koordinate im Herz befindet und sonst " " ausgibt.

Ein Aufruf des Programms soll dann etwa das folgende Resultat zeigen:

```
Geben Sie die gewuenschte Gesamtbreite ein: 30

      #########   #########
    ##########################
   ############################
  ##############################
  ##############################
   ############################
    ##########################
     ########################
       ####################
         ################
           ############
             #######
               ###
```

Hinweise:

- Wenn Sie mit dem Aussehen Ihres Herzens unzufrieden sind, weil es vertikal gestreckt ist, liegt dies vermutlich daran, dass in üblichen Schriftarten ein Zeichen ca. 2 mal so hoch wie breit ist. Wollen Sie ein gleichmäßigeres Ergebnis erzielen, so dürfen Sie die y-Werte (also die Höhe) aller Koordinaten vor der Eingabe in `inHerz` mit 2 multiplizieren. Dies wurde auch bei der obigen Beispielausgabe gemacht.
- Sie dürfen statt eines "#" für ein schöneres Ergebnis auch das Unicode-Zeichen "\u2665" (dargestellt als ♥) verwenden.

Lösung:

```
public class Herz {
  public static boolean inKreis(double px, double py, double cx,
                                double cy, double r) {
    double dx = px - cx;
    double dy = py - cy;
    return Math.sqrt(dx*dx + dy*dy) <= r;
  }

  public static boolean inRaute (double px, double py, double cx,
                                 double cy, double b) {
    double dx = px - cx;
    double dy = py - cy;
    return Math.abs(dx) + Math.abs(dy) <= b;
  }

  public static boolean inHerz(double px, double py, double cx,
                               double cy, double g) {
    double b = g/(1 + 2/Math.sqrt(2));
    double radius = b/Math.sqrt(2);
    return inRaute (px, py, cx, cy, b)
           || inKreis (px, py, cx + (b / 2), cy - (b / 2), radius)
           || inKreis (px, py, cx - (b / 2), cy - (b / 2), radius);
  }
```

```
  public static void main (String [] args) {
    System.out.print("Geben Sie die gewuenschte Gesamtbreite ein: ");
    int gesamtbreite = Integer.parseInt(System.console().readLine());
    double cx = gesamtbreite/2;
    double cy = gesamtbreite/2;

    //y*2 skaliert die Hoehe besser
    for (int y = 0; y*2 <= gesamtbreite; y++) {
      for (int x = 0; x <= gesamtbreite; x++) {
        if (inHerz (x, y*2, cx, cy, gesamtbreite)) {
          System.out.print("\u2665");
        } else {
          System.out.print(" ");
        }
      }
      System.out.println();
    }
  }
}
```

Aufgabe 8 (Das Haus): (1 + 3 + 5 + 2 = 11 Punkte)

In dieser Aufgabe sollen Sie ein Programm schreiben, das auf der Konsole ein Haus ausgibt. Die zur Illustration verwendeten Koordinatensysteme werden in der vorhergehenden Tutoraufgabe eingehend erklärt.

a) Wir wollen zuerst untersuchen, ob ein Punkt p mit den Koordinaten (p_x, p_y) in einem Rechteck liegt, das später den Hauptteil des Hauses darstellen soll.

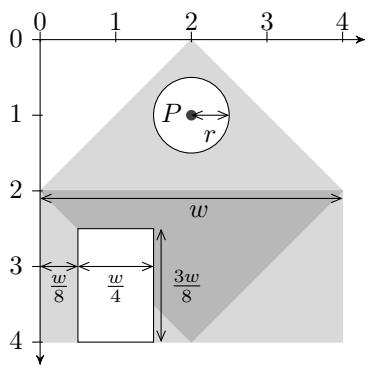
Vervollständigen Sie nun die folgende Funktion so, dass `true` zurückgegeben wird, wenn $(\texttt{px}, \texttt{py})$ in dem durch die obere linke Ecke $ol = (\texttt{olx}, \texttt{oly})$ und die untere rechte Ecke $ur = (\texttt{urx}, \texttt{ury})$ definierten Rechteck liegt und `false` in allen anderen Fällen:

```
public static boolean inRechteck(double px, double py, double olx,
                                  double oly, double urx, double ury)
{
  return...
}
```

Hinweise:

- Die Anweisung "`return` *expr*" wertet den Ausdruck *expr* aus und setzt das Ergebnis an der Aufrufstelle der Funktion ein.

b) Wir wollen nun untersuchen, ob ein Punkt in einem Haus der Breite w liegt. Das fertige Haus wird im folgenden Schaubild für $w = 4$, $r = w/8$, und $P = (w/2, w/4)$ illustriert:



Ein Punkt liegt genau dann in dem Haus mit Gesamtbreite w, wenn er gleichzeitig die folgenden Eigenschaften erfüllt:

- Er liegt in einem Rechteck mit den Parametern $ol = (0, w/2)$, $ur = (w, w)$ oder einer quadratischen Raute mit Zentrum $C = (w/2, w/2)$ und Rautenbreite $b = w/2$.
- Er liegt **nicht** im Fenster des Hauses, das durch einen Kreis mit Mittelpunkt $(w/2, w/4)$ und Radius $r = w/8$ dargestellt wird.
- Er liegt **nicht** in der Tür des Hauses. Diese wird durch ein Rechteck mit den Parametern $ol = (w/8, 5w/8)$ und $ur = (3w/8, w)$ beschrieben.

Verwenden Sie nun die Funktionen `inRaute` und `inKreis` aus der vorherigen Tutoraufgabe und `inRechteck` aus Teilaufgabe a), um die folgende Funktion zu implementieren, die genau dann `true` zurückgibt, wenn ein Punkt (`px`, `py`) in einem durch die Gesamtbreite `w` beschriebenen Haus liegt:

```
public static boolean inHaus(
        double px,
        double py,
        double w)
{
  return ...
}
```

Sie dürfen dabei annehmen, dass die Gesamtbreite `w` eine positive Zahl ist.

Hinweise:

- Verwenden Sie boolesche Verknüpfungen (`||` und/oder `&&`) sowie die Negation (`!`), um die verschiedenen Bedingungen miteinander zu verbinden.

c) Sie sollen nun die Funktion `inHaus` aus Aufgabenteil b) verwenden, um ein Haus auf der Konsole auszugeben.

Schreiben Sie ein Programm, das

1. den Benutzer nach der Gesamtbreite w für das Haus fragt und eine Integer-Zahl einliest. Zur Vereinfachung dürfen Sie annehmen, dass der Benutzer hier eine positive ganze Zahl eingibt.
2. mit Hilfe zweier Schleifen alle Koordinaten $(0,0), \ldots, (0,w), (1,0), \ldots (1,w), \ldots (w,w)$ abläuft und ein "`#`" ausgibt, wenn sich die Koordinate im Haus befindet, und sonst " " ausgibt.

Ein Aufruf des Programms soll dann etwa das folgende Resultat zeigen:

```
Geben Sie die gewuenschte Gesamtbreite ein: 46
                        *
                       *****
                      *********
                    *****   *****
                  ****         ****
                ****             ****
              *******            *******
            *********            *********
          **************     **************
        ******************************************
      ********************************************
    ************************************************
  ****************************************************
**********************************************************
**********************************************************
**********************************************************
******              ******************************
******              ******************************
******              ******************************
******              ******************************
******              ******************************
```

```
******              ****************************
******              ****************************
******              ****************************
******              ****************************
```

Hinweise:

- Wenn Sie mit dem Aussehen Ihres Hauses unzufrieden sind, weil es vertikal gestreckt ist, liegt dies vermutlich daran, dass in üblichen Schriftarten ein Zeichen ca. 2 mal so hoch wie breit ist. Wollen Sie ein gleichmäßigeres Ergebnis erzielen, so dürfen Sie die y-Werte (also die Höhe) aller Koordinaten vor der Eingabe in `inHaus` mit 2 multiplizieren. Dies wurde auch bei der obigen Beispielausgabe gemacht.

d) Passen Sie das Programm so an, dass die Aufforderung zur Eingabe einer Gesamtbreite so lange wiederholt wird, bis die vom Benutzer eingegebene Zahl mindestens 20 beträgt. Bieten Sie ihm außerdem die Möglichkeit, nach Eingabe der Breite eines von drei möglichen Zeichen zu wählen, aus dem das Haus bestehen soll. Eine mögliche Ausgabe bis zu diesem Punkt könnte aussehen wie folgt:

```
Geben Sie die gewuenschte Gesamtbreite ein (mind. 20): 10
Geben Sie die gewuenschte Gesamtbreite ein (mind. 20): 45
Aus welchem Zeichen soll das Haus bestehen?
1: Plus (+)
2: Stern (*)
3: Herz (♥)
```

Hier kann der Benutzer mit "1", "2" oder "3" sein bevorzugtes Zeichen wählen. Nutzen Sie eine `switch`-Anweisung zur Ausgabe der korrekten Zeichen. Falls der Benutzer eine ungültige Zahl eingibt, soll hier "#" gewählt werden.

Hinweise:

- Um ein Herz darzustellen, können Sie das Unicode-Zeichen "\u2665" verwenden.

Lösung:

```
public class Haus {

    public static boolean inKreis(
        double px,
        double py,
        double cx,
        double cy,
        double radius)
    {
        double dx = px - cx;
        double dy = py - cy;
        return Math.sqrt(dx * dx + dy * dy) <= radius;
    }

    public static boolean inRechteck(
        double px,
        double py,
        double olx,
        double oly,
        double urx,
        double ury)
    {
```

```
    return px >= olx && px <= urx && py >= oly && py <= ury;
}
public static boolean inRaute ( double px , double py , double cx ,
  double cy , double b ) {
  double dx = px - cx ;
  double dy = py - cy ;
  return Math.abs( dx ) + Math.abs( dy ) <= b ;
}

public static boolean inHaus(
    double px,
    double py,
    double w)
{
  return (inRechteck(px,py, 0,w/2, w,w) || inRaute(px,py, w/2,w/2,w/2)) &&
    !inKreis(px,py, w/2, w/4, w/8) &&  //Fenster
    !inRechteck(px,py, w/8, 5*w/8, 3*w/8, w); //Tuer
}

public static void main(String[] args) {
    int gesamtbreite;
    int zeichenwahl;
    do {
        System.out.print("Geben Sie die gewuenschte Gesamtbreite ein ");
        System.out.print("(mind. 20): ");
        gesamtbreite = Integer.parseInt(System.console().readLine());
    } while (gesamtbreite < 20);

    System.out.println("Aus welchem Zeichen soll das Haus bestehen?");
    System.out.println("1: Plus (+)");
    System.out.println("2: Stern (*)");
    System.out.println("3: Herz (\u2665)");
    zeichenwahl = Integer.parseInt(System.console().readLine());

    // waehle das richtige Zeichen
    char zeichen;
    switch (zeichenwahl) {
        case 1:
            zeichen = '+';
            break;
        case 2:
            zeichen = '*';
            break;
        case 3:
            zeichen = '\u2665';
            break;
        default:
            zeichen = '#';
    }

    // y * 2 skaliert die Hoehe besser
    for (int y = 0; y * 2 <= gesamtbreite; y++) {
        for (int x = 0; x <= gesamtbreite; x++) {
            if (inHaus(x, y * 2, gesamtbreite)) {
                System.out.print(zeichen);
            } else {
```

```
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```