ETH zürich

Einführung in die Programmierung

Prüfung HS16

Aufgabe 1 Theorie

*Nachfolgend sind die Operatoren in **absteigender Reihenfolge** ihrer Präzedenz aufgelistet:*

Kategorie	Operator
Unäre Operatoren	++ -- + - ! ~ (typ)
Arithmetische Operatoren	* / % + -
Schiebe-Operatoren	<< >> >>>
Vergleichs-Operatoren	< <= > >= instanceof
	== !=
Logische Operatoren	& ^ |
	&& ||
	? :
Zuweisungs-Operatoren	=, "op="

Treten mehrere Operatoren in einem Ausdruck auf, werden diese in einer definierten Reihenfolge ausgeführt. Diese Definition heißt Präzedenz. Dabei werden Operatoren mit höherer Präzedenz vor Operatoren mit niedrigerer Präzedenz ausgewertet. Durch Klammerung kann diese Präzedenz abgeändert werden. Oft werden die Ausdrücke dadurch auch übersichtlicher.

Aufgabe 1 (6 Punkte)

Gegeben sei eine Methode `main` in einer Java Klasse.

```
public static void main(String[] args) {
    /* body */
}
```

Kategorie	Operator
Unäre Operatoren	++ -- + - ! ~ (typ)
Arithmetische Operatoren	* / % + -
Schiebe-Operatoren	<< >> >>>
Vergleichs-Operatoren	< <= > >= instanceof
	== !=
Logische Operatoren	& ^ |
	&& ||
	? :
Zuweisungs-Operatoren	=, "op="

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefuegt werden. Geben Sie fuer jede Anweisung an, was fuer eine Ausgabe erzeugt wird. Achten Sie auf die korrekte Formatierung der verschiedenen Typen, also z.B. 7.0 statt 7 fuer eine reelle Zahl (`double`).

```
1. System.out.println(8 + 5 * 3 / 2                          );
2. System.out.println(1.5 * 4 * 7 / 8 + 3.4                  );
3. System.out.println(73 % 10 - 6 % 10 + 28 % 3              );
4. System.out.println(4 + 1 + 9 + "." + (-3 + 10) + 11 / 3   );
       // oder auch ``14.73''
5. System.out.println(3 / 14 / 7 / (1.0 * 2) + 10 / 6        );
6. System.out.println(10 > 11 == 4 / 3 > 1                   );
```

Aufgabe 1 (6 Punkte)

Gegeben sei eine Methode `main` in einer Java Klasse.

```
public static void main(String[] args) {
    /* body */
}
```

Kategorie	Operator
Unäre Operatoren	++ -- + - ! ~ (typ)
Arithmetische Operatoren	* / % + -
Schiebe-Operatoren	<< >> >>>
Vergleichs-Operatoren	< <= > >= instanceof
	== !=
Logische Operatoren	& ^ |
	&& ||
	? :
Zuweisungs-Operatoren	=, "op="

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefuegt werden. Geben Sie fuer jede Anweisung an, was fuer eine Ausgabe erzeugt wird. Achten Sie auf die korrekte Formatierung der verschiedenen Typen, also z.B. 7.0 statt 7 fuer eine reelle Zahl (`double`).

```
1. System.out.println(8 + 5 * 3 / 2                          ); // 15
2. System.out.println(1.5 * 4 * 7 / 8 + 3.4                  ); // 8.65
3. System.out.println(73 % 10 - 6 % 10 + 28 % 3              ); // -2
4. System.out.println(4 + 1 + 9 + "." + (-3 + 10) + 11 / 3   ); //  14.73
       // oder auch ``14.73''
5. System.out.println(3 / 14 / 7 / (1.0 * 2) + 10 / 6        ); // 1.0
6. System.out.println(10 > 11 == 4 / 3 > 1                   ); // true
```

Aufgabe 2 Theorie

*Java ist **Pass-By-Value***

-A copy of the passed-in variable is copied into the argument of the method. Any changes to the argument do not affect the original one. (gilt sowohl für primitive types als auch für reference types)

```
public class Swap {
    public static void swap(int x, int y) {
        int temp = x;
        x = y;
        y = temp;
        System.out.println("x(1) = " + x);
        System.out.println("y(1) = " + y);
    }
    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        swap(x, y);
        System.out.println("x(2) = " + x);
        System.out.println("y(2) = " + y);
    }
}
```

In swap(x,y) wird nicht die Referenz von x und y mitgegeben sondern eine Kopie ihrer Werte.

Deshalb wird in der Methode swap(x,y) weder x noch y (in der mainMethode) verändern.

(Bemerkung: int ist ein primitive type)

Aufgabe 2 Theorie

*Java ist immer **Pass-By-Value***

-A copy of the passed-in variable is copied into the argument of the method. Any changes to the argument do not affect the original one. (gilt sowohl für primitive types als auch für reference types)

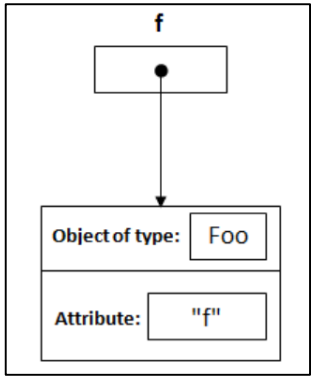
```
public class Main{
    public static void main(String[] args){
        Foo f = new Foo("f");
        changeReference(f); // It won't
change the reference!
        modifyReference(f); // It will
modify the object that the reference variable
"f" refers to!
    }
   public static void changeReference(Foo a){
        Foo b = new Foo("b");
        a = b;
    }
   public static void modifyReference(Foo c){
        c.setAttribute("c");
    }
}
```

wir übergeben eine **Kopie** der Referenz von f in die Methode changeReference(f).

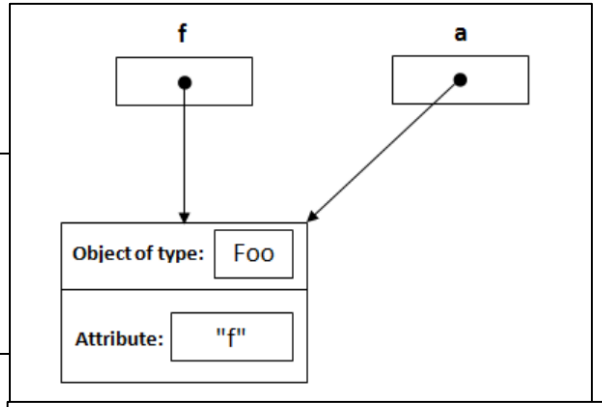
In der Methode changeReference selber, ist das Argument a eine Kopie der Referenz von f. Heisst, a = b; führt dazu das a auf b referenziert. Dies ändert aber nich die Referenz von f, da a lediglich eine **Kopie** von f ist!

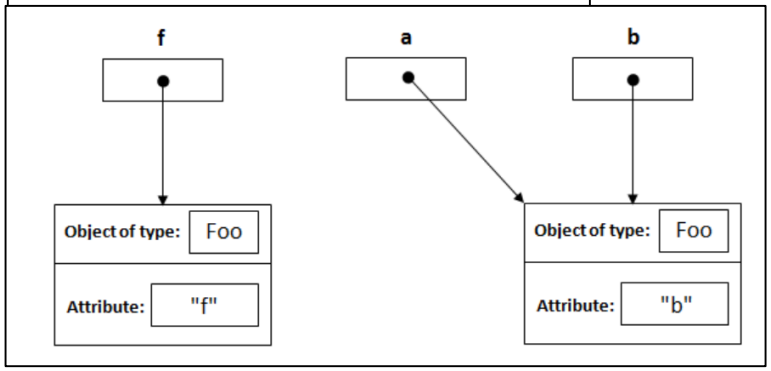
In der Methode modifyReference selber, ist das Argument c eine Kopie der Referenz von f. Mit c.setAttribute rufen wir die Methode via f auf(da c eine Kopie der Referenz von f ist). Deshalb verändert c.setAttribute auch f.

Aufgabe 2 Theorie

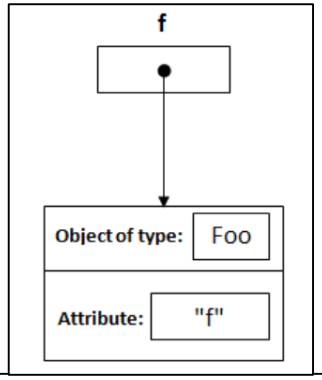



```
public static void changeReference(Foo a){
        Foo b = new Foo("b");
        a = b;
    }
```

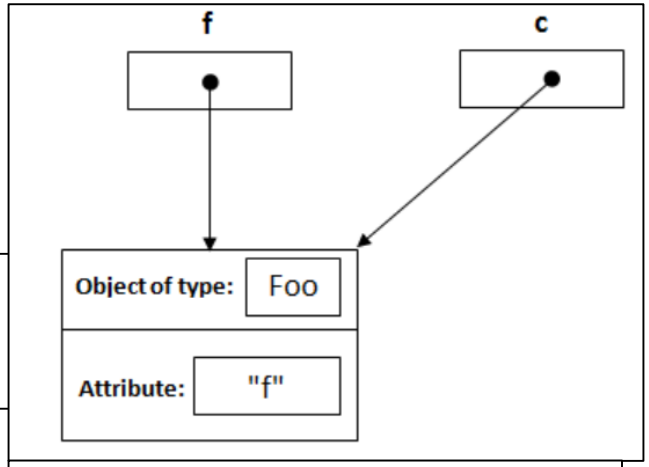


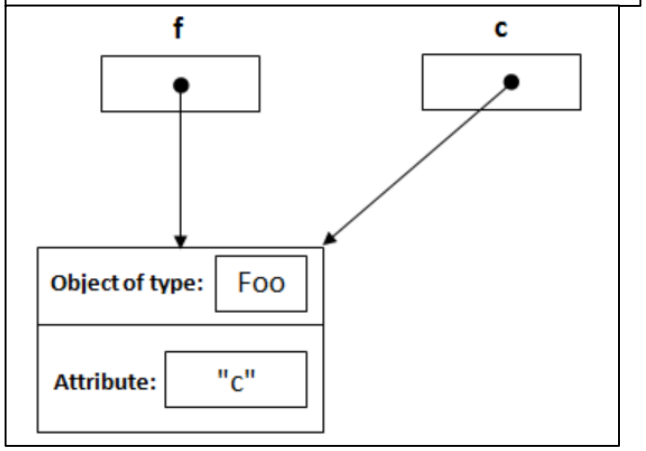


Aufgabe 2 Theorie




```
public static void modifyReference(Foo c){
        c.setAttribute("c");
    }
```





Aufgabe 2 (8 Punkte)

```
public class Problem {

public static void main(String[] args) {

            int a = 0, b = 1, c = 2, d;
            int [] x = {10, 20, 30};
            int [] y;
            int [] z = {100, 200, 300};

            d = a;
            method1(b, a);
            y = x;
            method2(y, 0);
            method3(b);

            System.out.println(a + " " + b + " " + c +
            " " + d);
            System.out.println(Arrays.toString(x));
            System.out.println(Arrays.toString(y));
            System.out.println(Arrays.toString(z));
}
```

```
static void method1(int p, int q) {
            q = p;
            q++;
}

static void method2(int []r, int i) {
            r[i] = r[i+1];
            r[i+1] = 0;
}

static void method3(int v) {
            int []z = new int[3];
            z[0] = v;
            z[1] = v + 1;
            z[2] = v + 2;
            method2(z, 0);
            }
}
```

Was gibt die main Methode aus?

Aufgabe 2

```
public class Problem {

public static void main(String[] args) {

            int a = 0, b = 1, c = 2, d;
            int [] x = {10, 20, 30};
            int [] y;
            int [] z = {100, 200, 300};

            d = a;
            method1(b, a);
            y = x;
            method2(y, 0);
            method3(b);

            System.out.println(a + " " + b + " " + c +
            " " + d);
            System.out.println(Arrays.toString(x));
            System.out.println(Arrays.toString(y));
            System.out.println(Arrays.toString(z));
}
```

Dieses z hat nichts mit dem z in der main Methode zu tun.

```
static void method1(int p, int q) {
            q = p;
            q++;
}

static void method2(int []r, int i) {
            r[i] = r[i+1];
            r[i+1] = 0;
}

static void method3(int v) {
            int []z = new int[3];
            z[0] = v;
            z[1] = v + 1;
            z[2] = v + 2;
            method2(z, 0);
            }
}
```

Sowohl p als auch q sind nur kopien von b und a. Deshalb verändert sich der wert von a und b nicht.

r[i] = r[i+1] ändert den Wert von y[i] (beim aufruf method2(y, 0).

Was gibt die main Methode aus?

```
0 1 2 0
[20, 0, 30]
[20, 0, 30]
[100, 200, 300]
```

Aufgabe 3 (5 Punkte)

```
public static int method(int n) {
if (n < 0) {
        n = n * 3;
        return n;
}else {
        n = n + 3;
        }
if(n % 2 == 1) {
        n = n + n%10;
        }
return n;
}
```

Aufrufe:

method(-5):
method(0):
method(7):
method(18):
method(49):

Aufgabe 3

```
public static int method(int n) {
if (n < 0) {
        n = n * 3;
        return n;
}else {
        n = n + 3;
        }
if(n % 2 == 1) {
        n = n + n%10;
        }
return n;
}
```

Aufrufe:

method(-5): -15
method(0): 6
method(7): 10
method(18): 22
method(49): 52

Aufgabe 4 (10 Punkte)

```
public static int mystery(int i, int j) {
int k = 0;
while (i > j) {
        i = i - j;
        k = k + (i-1);
        }
return k;
}
```

Aufrufe:

mystery(2, 9):
mystery(5, 1):
mystery(17, 5):
mystery(5, 5):
mystery(40, 10):

Aufgabe 4

```
public static int mystery(int i, int j) {
int k = 0;
while (i > j) {
        i = i - j;
        k = k + (i-1);
        }
return k;
}
```

Aufrufe:

mystery(2, 9): 0
mystery(5, 1): 6
mystery(17, 5): 18
mystery(5, 5): 0
mystery(40, 10): 57

Aufgabe 5 (10 punkte)

Geben Sie (fuer die angegebenen Stellen im Programm) an ob die drei Aussagen next < 0 und y > z und y == 0 IMMER, MANCHMAL, oder NIE wahr sind. Sie koennen diese drei Moeglichkeiten mit I/M/N abkuerzen.

```
public static int eineMethode
(Scanner console){
int y = 0;
int z = 1;
int next = console.nextInt();

//Point A
while(next >= 0) {
//Point B
          if(y > z) {
          //Point C
          z = y;
          }
          y++;
          next = console.nextInt();
          //Point D
          }
// Point E
return z;
}
```

	$next < 0$	$y > z$	$y == 0$
Point A			
Point B			
Point C			
Point D			
Point E			

Aufgabe 5 (10 punkte)

Geben Sie (fuer die angegebenen Stellen im Programm) an ob die drei Aussagen next < 0 und y > z und y == 0 IMMER, MANCHMAL, oder NIE wahr sind. Sie koennen diese drei Moeglichkeiten mit I/M/N abkuerzen.

```
public static int eineMethode
(Scanner console){
int y = 0;
int z = 1;
int next = console.nextInt();

//Point A
while(next >= 0) {
//Point B
          if(y > z) {
          //Point C
          z = y;
          }
          y++;
          next = console.nextInt();
          //Point D
          }
// Point E
return z;
}
```

	$next < 0$	$y > z$	$y == 0$
Point A	M	N	I
Point B	N	M	M
Point C	N	I	N
Point D	M	M	N
Point E	I	M	M

Aufgabe 6 (6 punkte)

1. Geben Sie fuer die EBNF Beschreibung octal ein Beispiel an, das legal ist , sowie ein Beispiel, das nicht legal ist. Nicht-legale Beispiele muesse die gleichen Symbole verwenden, die auch in der EBNF Beschreibung auftreten. (<= bedeutet «ist definiert als»)

    <sign> <= | + | -
    <digit> <= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
    <digits> <= | <digit> <digits>
    <octal> <= <sign> <digit> <digits>

2. Sind die folgenden EBNF Beschreibungen aequivalent? Begruenden Sie ihre Antwort.

a) [A [B]] und [A] [B]

b) { A { B } } und { A | B}

Aufgabe 6 (6 punkte)

1. Geben Sie fuer die EBNF Beschreibung octal ein Beispiel an, das legal ist , sowie ein Beispiel, das nicht legal ist. Nicht-legale Beispiele muesse die gleichen Symbole verwenden, die auch in der EBNF Beschreibung auftreten. (<= bedeutet «ist definiert als»)

<sign> <= | + | -
<digit> <= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
<digits> <= | <digit> <digits>
<octal> <= <sign> <digit> <digits>

Legal: +1, 2, -12 nicht legal: ++1, +-1, 1+1

2. Sind die folgenden EBNF Beschreibungen aequivalent? Begruenden Sie ihre Antwort.

a) [A [B]] und [A] [B]
Nein, B ist legal fuer den 2. Ausdruck aber nicht fuer den Ersten.

b) { A { B } } und { A | B}
Nein, BB ist legal fuer den 2. Ausdruck aber nicht fuer den Ersten.

Aufgabe 7 (5 punkte)

Schreiben Sie eine EBNF Beschreibung fuer aufzaehlung. Eine aufzaehlung besteht aus einem oder mehreren *X*, die nach den alternativen Regeln der Kommatrennung aneinander gereiht sind. Genauer: eine *<aufzaehlung>* besteht entwer

-aus einem *X*
-aus n >= 2 *X*, von denen die ersten n-1 *X* durch ein Komma(,) getrennt sind und die letzten beiden *X* sowohl durch ein Komma als auch das Wort *und* getrennt wird.

Legale Beispiele fuer <aufzaehlung> sind
- X
- X, und X
- X, X, X, X, und X

Illegale Beispiel fuer <aufzaehlung> sind
- X,X
- Und X
- , X , X, und X
- X und X, X ,X ,X
- X und X und X und X

Aufgabe 7 (5 punkte)

Schreiben Sie eine EBNF Beschreibung fuer aufzaehlung. Eine aufzaehlung besteht aus einem oder mehreren *X*, die nach den alternativen Regeln der Kommatrennung aneinander gereiht sind. Genauer: eine *<aufzaehlung>* besteht entwer

-aus einem *X*
-aus n >= 2 *X*, von denen die ersten n-1 *X* durch ein Komma(,) getrennt sind und die letzten beiden *X* sowohl durch ein Komma als auch das Wort *und* getrennt wird.

Legale Beispiele fuer <aufzaehlung> sind
- X
- X, und X
- X, X, X, X, und X

Illegale Beispiel fuer <aufzaehlung> sind
- X,X
- Und X
- , X , X, und X
- X und X, X ,X ,X
- X und X und X und X

Mögliche Lösung: *<group>* <= X { , X}
<aufzaehlung> <= X [, und X] | *<group>*, und X

Aufgabe 8 (10 punkte)

Beschreiben Sie in 2-3 Saetzen, was diese Methode macht:

```
public static int unbekannt(int []a1, int []a2) {
        int funny = -1;
        for(int i = 0; i <= a1.length - a2.length; i++) {
                int count = 0;
                for(int j = 0; j < a2.length; j++) {
                        if(a1[ i + j] == a2[j]) {
                        count++;
                        }
                }
                if (count == a2.length) {
                        funny = i;
                        }
        }
        return funny;
}
```

Beispiel
int [] x = {10, 20, 30, 60, 70, 20, 30}; // Länge 7
int [] z = {20, 30}; // Länge 2

```
(sonbekannt(x, z)
  funny = -1
  for (int i = 0; i <= 7-2; i++) {
        int count = 0;
        for (int j = 0; j < 2; j++) {
              if (x[i+j] == z[j])
                    count++;
        }
        if (count == 2){
              funny = i;
        }
  }
  return funny
}
```

i = 0, ..., 5

Erster Durchgang i = 0

x = {10, 20, 30, 60, 70, 20, 30}
z = {20, 30}

i = 1

x = {10, 20, 30, 60, 70, 20, 30}
z = {20, 30}

Bis i = 5

x = {10, 20, 30, 60, 70, 20, 30}
z = {20, 30}

Aufgabe 8 (10 punkte)

Beschreiben Sie in 2-3 Saetzen, was diese Methode macht:

```
public static int unbekannt(int []a1, int []a2) {
        int funny = -1;
        for(int i = 0; i <= a1.length - a2.length; i++) {
                int count = 0;
                for(int j = 0; j < a2.length; j++) {
                        if(a1[ i + j] == a2[j]) {
                        count++;
                        }
                }
                if (count == a2.length) {
                        funny = i;
                        }
        }
        return funny;
}
```

Gegeben zwei int Arrays a1 und a2, unbekannt (a1,a2) findet die letzte Position in a1 an der die Folge in a2 auftritt. Wenn a2 nicht in a1 auftritt wird -1 zurueckgegeben

Aufgabe 9 (10 punkte)

Schreiben Sie eine Methode mitte die als Parameter eine positive ganze Zahl x mit einer **ungeraden Anzahl an Ziffern** akzeptiert (Sie koennen davon ausgehen, dass **nur** solche Zahlen uebergeben werden). Mitte soll den Wert der Ziffer **in der Mitte von x** zurueckgeben.

Beispiele: mitte(145) ergibt 4

Aufgabe 9 (10 punkte)

Schreiben Sie eine Methode mitte die als Parameter eine positive ganze Zahl x mit einer **ungeraden Anzahl an Ziffern** akzeptiert (Sie koennen davon ausgehen, dass **nur** solche Zahlen uebergeben werden). Mitte soll den Wert der Ziffer **in der Mitte von x** zurueckgeben.

Beispiele: mitte(145) ergibt 4
Mögliche Lösung:

```
public static int mitte(int x) {
String s = Integer.toString(x);

String t =
s.substring(s.length()/2,
s.length()/2 + 1);
int i = Integer.parseInt(t);
return i;
}
```

Aufgabe 10 (10 punkte)

Schreiben Sie eine Methode spannweite, die als Parameter einen Array von int Werten akzeptiert und die Spannweite der Werte in dem Array als Ergebnis zurueckgibt. Die Spannweite ist definiert als die Differenz zwischen dem groessten und kleinsten Element plus 1. Z.B wenn das groesste Element 12 und das kleinste Element 4 ist, dann ist die Spannweite 9. Wenn das groesste und das kleinste Element gleich sind, dann ist die Spannweite 1. Falls das Array leer ist, ist die Spannweite 0.

Aufgabe 10 (10 punkte)

Schreiben Sie eine Methode spannweite, die als Parameter einen Array von int Werten akzeptiert und die Spannweite der Werte in dem Array als Ergebnis zurueckgibt. Die Spannweite ist definiert als die Differenz zwischen dem groessten und kleinsten Element plus 1. Z.B wenn das groesste Element 12 und das kleinste Element 4 ist, dann ist die Spannweite 9. Wenn das groesste und das kleinste Element gleich sind, dann ist die Spannweite 1. Falls das Array leer ist, ist die Spannweite 0.

```
public static int spannweite(int [] a) {
if(a.length == 0) {
          return 0;
          }
int min = a[0];
int max = a[0];
for(int i = 1; i <a.length; i++) {
          min = Math.min(min,  a[i]);
          max = Math.max(max, a[i]);
          }
return max - min + 1;
}
```

Aufgabe 11 (8 punkte)

Das folgende Programm gibt 4 Zeilen aus. Geben Sie den Output an wie er auf der Konsole gedruckt erscheinen wuerde. (Tipp: Siehe Theorie zu Aufgabe 2)

```
import java.util.Arrays;

public class ReferenceMystery {
        public static void main(String [] args) {
                int x = 0;
                int [] a = new int[4];

                x++;
                mystery(x,a);
                System.out.println(x + " " + Arrays.toString(a));

                x++;
                mystery(x,a);
                System.out.println(x + " " + Arrays.toString(a));
        }

        public static void mystery(int x, int [] a) {
                x++;
                a[x]++;
                System.out.println(x + " " + Arrays.toString(a));
        }
}
```

Aufgabe 11 (8 punkte)

Das folgende Programm gibt 4 Zeilen aus. Geben Sie den Output an wie er auf der Konsole gedruckt erscheinen wuerde. (Tipp: Siehe Theorie zu Aufgabe 2)

```
import java.util.Arrays;

public class ReferenceMystery {
        public static void main(String [] args) {
                int x = 0;
                int [] a = new int[4];

                x++;
                mystery(x,a);
                System.out.println(x + " " + Arrays.toString(a));

                x++;
                mystery(x,a);
                System.out.println(x + " " + Arrays.toString(a));
        }

        public static void mystery(int x, int [] a) {
                x++;
                a[x]++;
                System.out.println(x + " " + Arrays.toString(a));
        }
}
```

2 [0, 0, 1, 0]
1 [0, 0, 1, 0]
3 [0, 0, 1, 1]
2 [0, 0, 1, 1]

Aufgabe 12 (12 punkte)

Gegeben seien diese Klassen:

```
class Eins{
public String toString() {
            return "Eins";
            }
public void method1() {
            System.out.print("Eins 1   ");
            }

public void method2() {
            System.out.print("Eins 2   ");
            }
}
```

```
class Zwei extends Eins{
public void method2() {
            System.out.print("Zwei 2");
            method1();
            }
}
```

```
class Drei extends Zwei{
public void method1() {
            System.out.print("Drei 1     ");
            }
public String toString() {
            return "Drei";
            }
}
```

```
class Vier extends Drei{
public void method2() {
            super.method2();
            System.out.print("Vier 2");
            }

public String toString() {
            return "Vier " + super.toString();
            }
}
```

Das folgende Programmsegment ist ein Klient dieser Klassen. Welchen Output produziert dieses Programmsegment?

```
Eins [] elements = { new Zwei() ,new Eins(), new Vier(),
new Drei()};
for(int i = 0; i < elements.length; i++) {
            System.out.println(elements[i]);
            elements[i].method1();
            System.out.println();
            elements[i].method2();
            System.out.println();
            System.out.println();
}
```

Aufgabe 12 (12 punkte)

Gegeben seien diese Klassen:

```
class Eins{
public String toString() {
            return "Eins";
            }
public void method1() {
            System.out.print("Eins 1   ");
            }

public void method2() {
            System.out.print("Eins 2   ");
            }
}
```

```
class Zwei extends Eins{
public void method2() {
            System.out.print("Zwei 2");
            method1();
            }
}
```

```
class Drei extends Zwei{
public void method1() {
            System.out.print("Drei 1     ");
            }
public String toString() {
            return "Drei";
            }
}
```

```
class Vier extends Drei{
public void method2() {
            super.method2();
            System.out.print("Vier 2");
            }

public String toString() {
            return "Vier " + super.toString();
            }
}
```

Das folgende Programmsegment ist ein Klient dieser Klassen. Welchen Output produziert dieses Programmsegment?

```
Eins [] elements = { new Zwei() ,new Eins(), new Vier(),
new Drei()};
for(int i = 0; i < elements.length; i++) {
            System.out.println(elements[i]);
            elements[i].method1();
            System.out.println();
            elements[i].method2();
            System.out.println();
            System.out.println();
}
```

Eins
Eins 1
Zwei 2 Eins 1

Eins
Eins 1
Eins 2

Vier Drei
Drei 1
Zwei 2 Drei 1 Vier 2

Drei
Drei 1
Zwei 2 Drei 1

Aufgabe 13 (10 punkte)

Ergaenzen Sie die folgende Methode admin, welche eine Tabelle von Angestellten und ihren Gehaeltern entgegen nimmt, sodass die Methode die Person mit dem hoechsten Gehalt ermittelt und diese Person und ihr Gehalt ausgibt. Sie koennen davon ausgehen, dass es nur eine Person mit dem hoechsten gehalt gibt und dass alle Gehaelter ganze Zahlen und groesser als 0 sind. Was gibt das Prgoramm aus? (Sie brauchen nicht unbedingt alle Zeilen in admin zu fuellen

```
import java.util.*;

public class Problem13 {

    public static void main(String[] args) {

        _____ = new TreeMap<_____>();


        salaries.put("Peter", 40000);
        sal
        salaries.put("Jane", 56000);
        salaries.put("Robert", 80000);
        salaries.put("Clara", 50000);

        admin(salaries);
    }
```

```
public static void admin(_____ table) {

    _____

    _____

    _____
```

Aufgabe 13 (10 punkte)

Ergaenzen Sie die folgende Methode admin, welche eine Tabelle von Angestellten und ihren Gehaeltern entgegen nimmt, sodass die Methode die Person mit dem hoechsten Gehalt ermittelt und diese Person und ihr Gehalt ausgibt. Sie koennen davon ausgehen, dass es nur eine Person mit dem hoechsten gehalt gibt und dass alle Gehaelter ganze Zahlen und groesser als 0 sind. Was gibt das Prgoramm aus? (Sie brauchen nicht unbedingt alle Zeilen in admin zu fuellen

```
import java.util.*;

public class Problem13 {

    public static void main(String[] args) {

        Map<String, Integer> salaries = new TreeMap<String, Integer>();

        salaries.put("Peter", 40000);
        sal
        salaries.put("Jane", 56000);
        salaries.put("Robert", 80000);
        salaries.put("Clara", 50000);

        admin(salaries);
    }
```

```
        public static void admin(Map<String,Integer> table) {
            int max = 0;
            String maxName;

            for (String name: table.keySet()) {
                int s = table.get(name);
                if (s > max) {
                    max = s;
                    maxName = name;
                }
            }
            System.out.print("Person: " + maxName);
            System.out.println(" Gehalt: " + max  );
        }

    }
```

[redacted]

Aufgabe 14 (10 punkte)

Geben Sie fuer die folgenden Hoare Tripel an, ob sie gueltig oder ungueltig sind. Alle Variablen sind vom Typ `int` und es gibt keinen Overflow/Underflow.

1. `{ y != 0 }  x = 2 * y * y - 1;  { x > 0 }`

2. `{ y != 0 ∧ z != 0 }  x = y * y;  { x > 0 ∧ z > 0 }`

3. `{ y > 0 }  x = 2 * y;  { x > y }`

4.
```
{ true }
if (x > 1)
    y = x * x;
else
    y = 2;
{ y > 1 }
```

Aufgabe 14 (10 punkte)

Geben Sie fuer die folgenden Hoare Tripel an, ob sie gueltig oder ungueltig sind. Alle Variablen sind vom Typ `int` und es gibt keinen Overflow/Underflow.

1. `{ y != 0 }  x = 2 * y * y - 1;  { x > 0 }`
   Gueltig.
2. `{ y != 0 ∧ z != 0 }  x = y * y;  { x > 0 ∧ z > 0 }`
   Ungueltig.
3. `{ y > 0 }  x = 2 * y;  { x > y }`
   Gueltig.
4.
```
{ true }
if (x > 1)
    y = x * x;
else
    y = 2;
{ y > 1 }
```
   Gueltig.