

# Technische Universität Braunschweig

Institut für Anwendungssicherheit

## Programmieren I

Prof. Dr. Martin Johns

4. März 2019

Nachname:

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Vorname:

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Matrikelnummer:

--	--	--	--	--	--	--	--

Klausurnummer: 001

**Studiengang:**  Bachelor  Master  Diplom  Frühstudium  Erasmus

**Fachrichtung:**  Informatik  Wirtschaftsinformatik  Mathematik  Physik

Mobilität und Verkehr  Psychologie  Maschinenbau ( Mechatronik)

Elektrotechnik  Wirtschaftsingenieurwesen ( Bauing  E-Technik  Maschbau)

Finanz- und Wirtschaftsmathematik  IST  Sonstige: \_\_\_\_\_

**Versuch der Notenverbesserung:**

Die Bearbeitungszeit beträgt 120 Minuten. Die Klausur besteht aus 6 Aufgaben.

Aufgabe	1	2	3	4	5	6	$\Sigma$
max. Punkte	5	5	13	10	8	19	60
Punkte							

### Note:

Hinweise: Die Aufgaben befinden sich in der Regel auf den **Rückseiten** der Blätter! So haben Sie auf dem gegenüberliegenden Blatt Platz für Ihre Lösung ohne Umblättern zu müssen. Bitte stellen Sie sicher, dass Sie keine der Aufgaben übersehen. Die Noten werden direkt über das QIS bekanntgegeben, Sie brauchen sich die Klausurnummer nicht merken.

**Aufgabe 1 (5 Punkte)** Bitte kreuzen Sie die zutreffenden Aussagen an. Für jede vollständige richtige Antwort erhalten Sie einen Punkt. Alle Fragen dieser Aufgabe beziehen sich auf Java 10.

*Datentypen:*

- a und b sind vom Typ `int` und  $\geq 0$ . a+b ist immer  $\geq a$  und  $\geq b$ .
- Referenztypen werden standardmäßig mit `null` initialisiert.
- Der Inhalt von `Strings` kann mit `==` verglichen werden.

*Deklaration:*

- Diese Deklaration ist zulässig: `long a = 2.2F;`
- Diese Deklaration ist zulässig: `double b = 2e2f;`
- Diese Deklaration ist zulässig: `float c = .2;`

*Objektorientierung:*

- Eine Klasse kann direkt von mehreren Klassen erben.
- Mehrere Klassen können von derselben Basisklasse erben.
- Von als `final` deklarierten Klassen kann geerbt werden.

*Interface:*

- Interfaces können direkt instanziert werden:
- Ein Interface kann Konstanten enthalten:
- Eine Klasse kann beliebig viele Interfaces implementieren:

*Ausnahmebehandlung:*

- Eine try-catch-Anweisung kann mehr als eine catch Klausel haben:
- Eine try-catch-Anweisung muss über mindestens eine catch Klausel verfügen:
- Die `catch-or-throw`-Regel gilt auch für `RuntimeExceptions`:

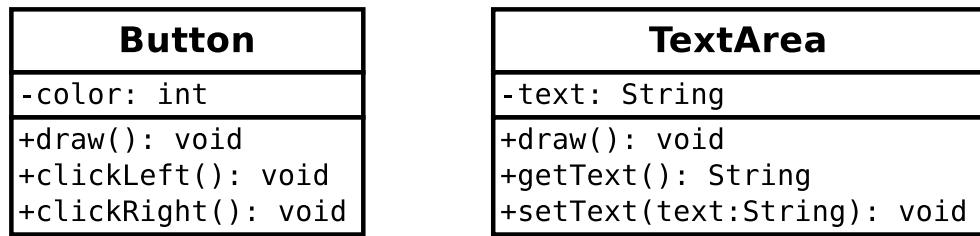


Abbildung 1: Klassendiagramm

**Aufgabe 2 (5 Punkte)** In Abbildung 1 ist ein UML Klassendiagramm gegeben.

- a) Geben Sie den passenden Java Code für die Klasse `TextArea` an.  
Die Methodenrümpfe sollen Sie hierbei **leer** lassen. (*3 Punkte*)
- b) Überlegen Sie sich ein *Interface*, welches das gemeinsame Verhalten von `TextArea` und `Button` modelliert. Geben Sie diesem einen sprechenden Namen und schreiben Sie den zugehörigen Java Code auf. (*2 Punkte*)

**Lösung Aufgabe 2:**

**Aufgabe 3 (13 Punkte)** Welche der folgenden Schleifen terminieren? Geben Sie im Falle der Terminierung an, welche Werte die Variablen x und y einnehmen, nachdem die jeweiligen Anweisungen ausgeführt wurden. (4 Punkte)

a) `int x = 8;  
int y = 2;  
while(y < x) {  
 x = x + 4;  
 if (y % 2 == 0) {  
 y = y * 2;  
 } else {  
 y = y - 1;  
 }  
}`

Die Schleife terminiert: ja  nein       x = \_\_\_\_\_      y = \_\_\_\_\_

b) `int x = 5;  
int y = 20;`

```
for(x = 1; x < y; ) {  
    switch(x % 4) {  
        case 1:  
        case 3:  
            x = x + 3;  
            break;  
        case 0:  
        case 2:  
            x = x - 1;  
            y = y - 5;  
            break;  
        default:  
            x = -1;  
            y = -1;  
            break;  
    }  
}
```

Die Schleife terminiert: ja  nein       x = \_\_\_\_\_      y = \_\_\_\_\_

Vor jedem der folgenden Ausdrücke ist die Deklaration `int x = 15;` vorrausgesetzt. Geben Sie für jeden der Ausdrücke den Typ und den Wert des Ausdrucks an. (3 Punkte)

- c)  $(x/2 == x/2.)$       Typ: \_\_\_\_\_      Wert: \_\_\_\_\_
- d)  $(x++\%2)+x$       Typ: \_\_\_\_\_      Wert: \_\_\_\_\_
- e)  $(x\%\!11) != (--x/3)$       Typ: \_\_\_\_\_      Wert: \_\_\_\_\_

Gegeben ist `a`, deklariert durch `int[] a = new int[6]`. Geben Sie jeweils den Inhalt des Felds nach Ausführung der beiden Schleifen an: (6 Punkte)

```
f) for(int i = 11; i >= 0; i = i - 1) {  
    if (i % 2 != 0 || i == 8 || i == 3) {  
        a[i/2] = i;  
    } else {  
        a[i/2] = i * -i;  
    }  
}  
for(int i = 0; i < a.length; i++) {  
    a[i] = a[a.length - 1 - i];  
}
```

Nach der ersten Schleife:

`a[0] = _____`      `a[1] = _____`      `a[2] = _____`  
`a[3] = _____`      `a[4] = _____`      `a[5] = _____`

Nach der zweiten Schleife:

`a[0] = _____`      `a[1] = _____`      `a[2] = _____`  
`a[3] = _____`      `a[4] = _____`      `a[5] = _____`

**Aufgabe 4 (10 Punkte) (Objektorientierung)** Gegeben sei das folgende Java-Programm:

```
_____ Formattable {  
    String format();  
}  
abstract class FormattableDefault _____ Formattable {  
    public String format() {  
        return this.toString();  
    }  
}  
class Person implements Formattable {  
    private String name;  
    public Person(String name) { this.name = name; }  
    public String format() {  
        return "Person: " + name;  
    }  
}  
class Document _____ FormattableDefault {  
    private _____ int number;  
    public Document(int number) { _____ .number = number; }  
    public String toString() {  
        return "Document: Nr. " + number;  
    }  
}  
class Main {  
    public _____ void main(String[] args) {  
        _____ items = {  
            new Person("Max Mustermann"),  
            new Document(1),  
            new _____(12)  
        };  
        for(_____ i : items) {  
            System.out.println(i._____());  
        }  
    }  
}
```

Füllen Sie die Lücken im obigen Programm, so dass es die nachfolgende Ausgabe erzeugt:

Person: Max Mustermann

Document: Nr. 1

Document: Nr. 12

Sie dürfen dabei den vorhandenen Code nicht ändern und auch nicht streichen. Zusätzliche Ausgaben einzubauen ist ebenfalls nicht zulässig. Nicht jede Lücke muss notwendigerweise auch gefüllt werden!

**Aufgabe 5 (8 Punkte)** Gegeben sei die folgende rekursive Methode:

```
static int f(int n) {  
    if (n < 2) {  
        return -n;  
    } else {  
        return n * (f(n-2) - f(n-1));  
    }  
}
```

- a) Welchen Wert liefert der Aufruf  $f(4)$ ? In welcher Reihenfolge und mit welchen Parametern wird  $f$  dabei aufgerufen? Geben Sie die Reihenfolge der Aufrufe explizit an. Wie groß ist die maximale Rekursionstiefe, das heißt die maximale Anzahl gleichzeitig aktiver Aufrufe? (*6 Punkte*)
- b) Liefert der Aufruf  $f(x)$  mit  $x \geq 0$ , unter der Annahme dass der Aufruf erfolgreich terminiert, immer einen positiven `int`-Wert? Begründen Sie Ihre Antwort. (*2 Punkte*)

**Lösung Aufgabe 5:**

**Aufgabe 6 (19 Punkte)** Erstellen Sie ein Programm, welches folgende Aufgabe löst:  
Gegeben ist einen mit natürlichen Zahlen, d.h. mit Zahlen  $\geq 0$ , gefülltes Array. Darin soll die größte, doppelt vorkommende Zahl gefunden werden.

Beispiel: In dem Array { 1, 5, 9, 1, 7, 9, 12, 14, 21, 49, 21 } kommen die Zahlen 1, 9 und 21 jeweils doppelt vor. Davon ist 21 die größte Zahl und somit der gewünschte Rückgabewert.

- a) Schreiben Sie eine Hilfsmethode, welche die eigentliche Suche implementiert. Sollte in dem als Parameter übergebenen Array keine Zahl mehr als einmal vorkommen, soll die Methode eine `IllegalArgumentException` werfen. (*12 Punkte*)

- b) Schreiben sie eine zugehörige `main`-Methode. In dieser sollen die drei folgenden `int` Arrays fix deklariert sein:

```
int[] numbers1 = { 1, 9, 1, 7, 9, 12, 21, 49, 21 };
int[] numbers2 = { 1, 49, 21 };
int[] numbers3 = { 49, 7, 21, 14, 49, 21 };
```

Der Nutzer soll Beim Aufruf des Programms eine Zahl zwischen 1 und 3 als Kommandozeilen-Parameter übergeben. Basierend auf dieser Zahl soll eins der gegebenen Arrays ausgewählt werden. Dieses soll dann als Parameter an die in Aufgabe a) implementierte Hilfsfunktion übergeben werden. Zum Schluss soll das Ergebnis auf der Kommandozeile ausgegeben werden.

Wenn keine bzw. keine gültige Zahl übergeben wurde, geben Sie eine sinnvolle Fehlermeldung aus. Fangen Sie außerdem die Exception der Hilfsmethode und geben Sie hier ebenfalls eine Fehlermeldung aus. (*7 Punkte*)

**Für beide Teilaufgaben gilt:**

Sie erhalten nur dann Punkte, wenn Sie ein Java-Programm schreiben das erkennbar geeignet ist die Problemstellung zu lösen. Sie dürfen (weitere) eigene Hilfsmethoden schreiben und verwenden, aber weder Klassen noch Methoden importieren. **Verwenden Sie aussagekräftige Variablen- und Funktionsnamen und kommentieren Sie Ihren Code!**

**Lösung Aufgabe 6:**