Java Fundamentals

Verwendung von Variablen

Variablen Deklaration

- Variablen müssen deklariert werden bevor sie verwendet werden
- Variablen, die in Methoden deklariert werden, nennt man lokale Variablen
- Lokale Variablen müssen initialisiert werden bevor sie ausgelesen werden

```
package at.java;

public class IntegerExample {
    public static void main(String[] args) {
        int a;
        int b;
        int c;

        a = 5;
        b = 3;
        c = a + b;

        System.out.println(c);
    }
}
```

Variablen Deklaration

- Deklaration und Initialisierung können kombiniert werden

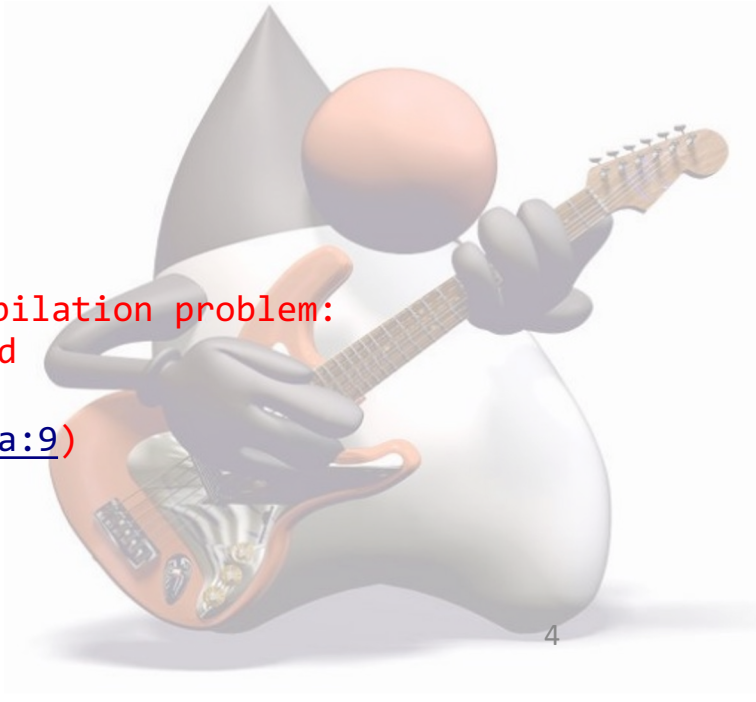
```
package at.java;

public class IntegerExample {
	public static void main(String[] args) {
		int a = 5;
		int b = 3;
		int c = a + b;

		System.out.println(c);
	}
}
```

Variablen Deklaration

- Nicht initialisierte lokale Variablen verursachen Compiler Fehler

```
package at.java;

public class IntegerExample {
	public static void main(String[] args) {
		int a;
		int b;
		int c;

		System.out.println(c);
	}
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
	The local variable c may not have been initialized

	at at.java.IntegerExample.main(IntegerExample.java:9)
```



Arithmetische Operationen sind immer 64 oder 32 bit Operationen

```
package at.java;

import java.util.Scanner;

public class PrimitiveDataTypes {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.println("Please input a byte value:");
		byte b1 = scanner.nextByte();
		System.out.println("Another one please:");
		byte b2 = scanner.nextByte();

		byte result = b1 + b2; //Compile error!!! Arithmetic is always at least in 32 bit
		System.out.println("b1 + b2 = " + result);
	}
}
```

Konvertierung zwischen Datentypen (type casting)

```
package at.java;

import java.util.Scanner;

public class PrimitiveDataTypes {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.println("Please enter a byte value:");
		byte b1 = scanner.nextByte();
		System.out.println("Another one please:");
		byte b2 = scanner.nextByte();

		byte result = (byte)(b1 + b2); //now it works → this is called type casting
		System.out.println("b1 + b2 = " + result);
	}
}
```

Achtung bei Überlauf – keine Fehlermeldung!

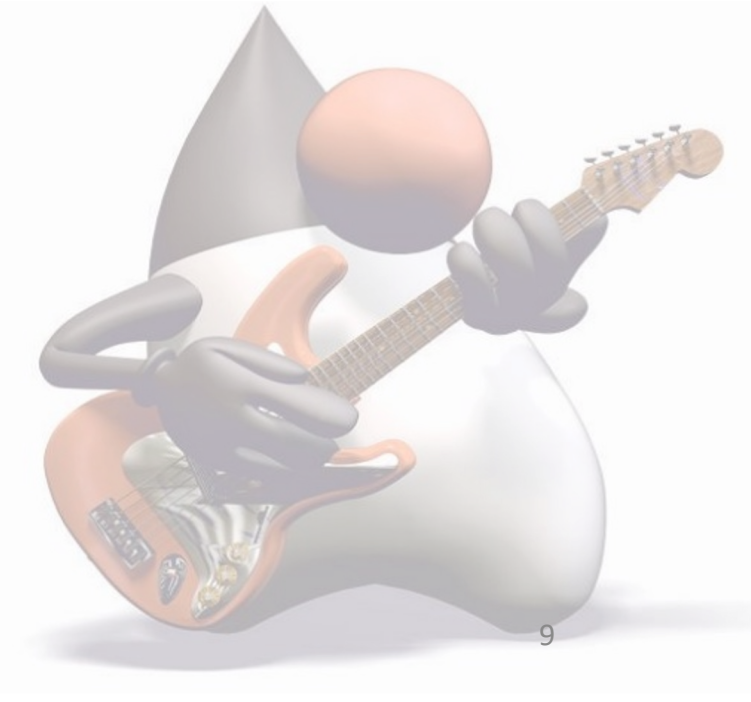
```
package at.java;

import java.util.Scanner;

public class PrimitiveDataTypes {
	public static void main(String[] args) {
		byte b1 = 125; //ok
		byte b2 = 126; //ok

		byte result = (byte)(b1+b2);
		System.out.println(b1 + " + " + b2 + " = " + result); //overflow
	}
}
```

Wertebereich numerische Datentypen

```
byte [-128,127]
short [-32768,32767]
int [-2147483648,2147483647]
long [-9223372036854775808,9223372036854775807]
float[1.4E-45,3.4028235E38]
double[4.9E-324,1.79769313486231570E308]
```

Fließkommazahlen

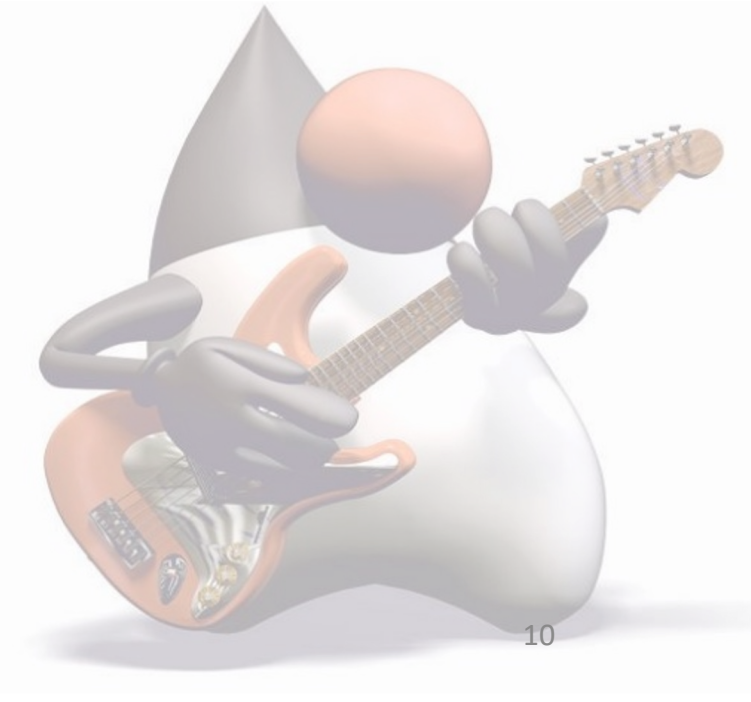
```
package at.java;

import java.util.Scanner;

public class PrimitiveDataTypes {
	public static void main(String[] args) {
		float f = 5.24F;
		double d = 3.23;
	}
}
```



Alphanumerische Zeichen

```
package at.java;

import java.util.Scanner;

public class PrimitiveDataTypes {
	public static void main(String[] args) {
		char a = 'a';
		char b = (char)(a + 1); // is 'b'

		// hexadecimal unicode
		char greekCharacter = '\u003c';

	}
}
```

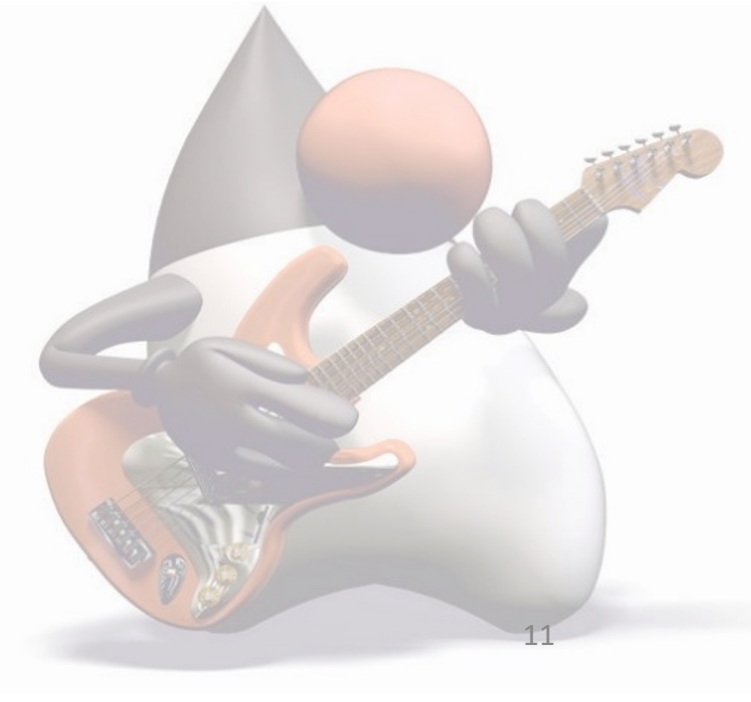
- Characters können Teil einer arithmetischen Operation sein
- Characters können auch über Unicodes definiert werden



Booleans

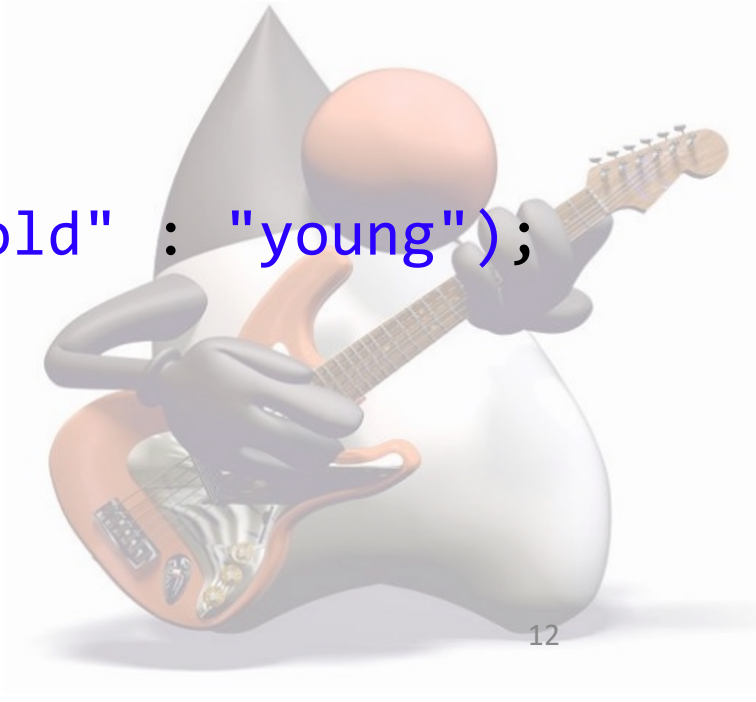
- Booleans können nicht Teil einer arithmetischen Operation sein
- Booleans können nur zwei Werte halten, true oder false
- Booleans werden mit dem == Operator verglichen

```
public class PrimitiveDataTypes {
        public static void main(String[] args) {
                boolean b1 = false;
                boolean b2 = 1; //COMPILE ERROR

                boolean b3 = true;

                boolean b4 = (1 < 2); // true
                boolean b5 = (1 == 4); // false
                boolean b6 = (4 >= 8); // false

                int a = 5;
                int b = 14;
                boolean b7 = a == b; // false
        }
}
```



Operatoren

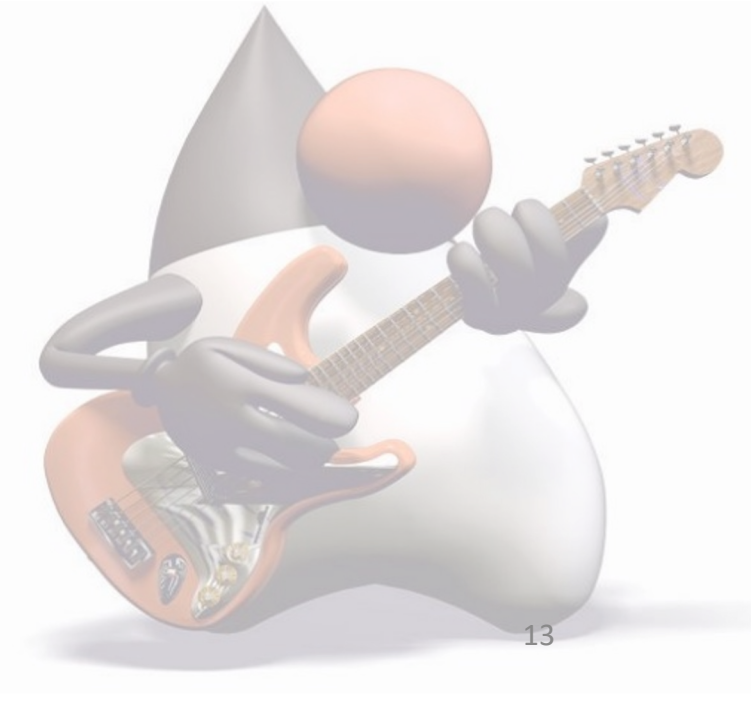
```
public class PrimitiveDataTypes {
	public static void main(String[] args) {
		int a = 5;

		a += 4; // same as a = a + 4;
		a++; // same as a = a + 1;
		a *= 3; // same as a = a * 3;

		int age = 40;
		String youngOrOld = (age >= 40 ? "old" : "young");

	}
}
```



Strings

String ist eine Klasse, daher zählt der String zu den Referenz-Datentypen.

```
public class PrimitiveDataTypes {
	public static void main(String[] args) {
		String name = "Michael";
		// String lastname = 'Schaffler'; -> COMPILE ERROR

		String lastname = "Schaffler";

		//Strings can be concatenated
		String alltogether = name + " " + lastname;
		System.out.println(alltogether);
	}
}
```

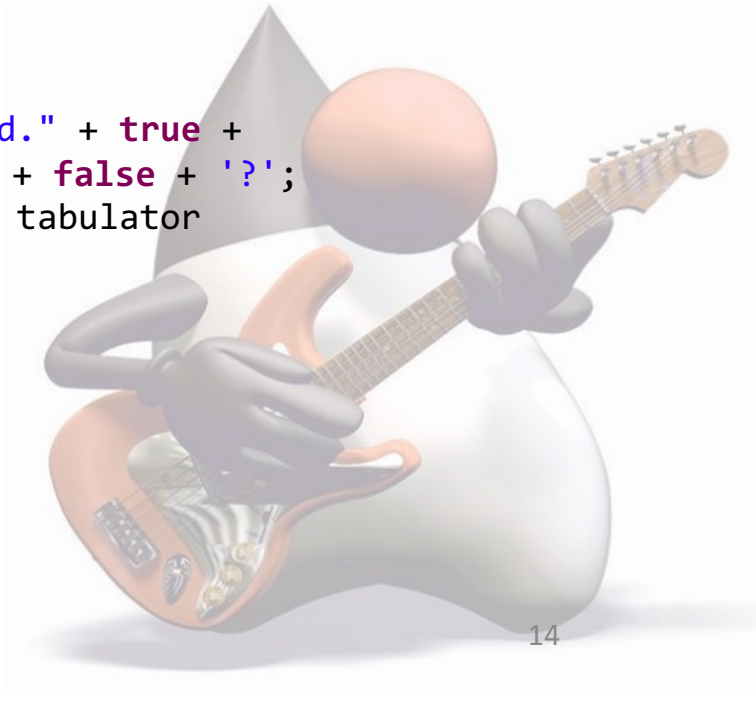


Strings

Alle Datentypen können zu Strings konvertiert werden

```
public class PrimitiveDataTypes {
        public static void main(String[] args) {
                String number = String.valueOf(42);
                String booleanString = String.valueOf(false);
                String newline = "\n";
                String tabulator = "\t";
                String backslash = "\\";
                String doublequote = "this is a double quote: \" ";

                String concatenated = "Michael is " + 30 + " years old." + true +
                                                                " or " + false + '?';
                System.out.println(number + booleanString + newline + tabulator
                                                        + concatenated);
                System.out.println(doublequote);
        }
}
```

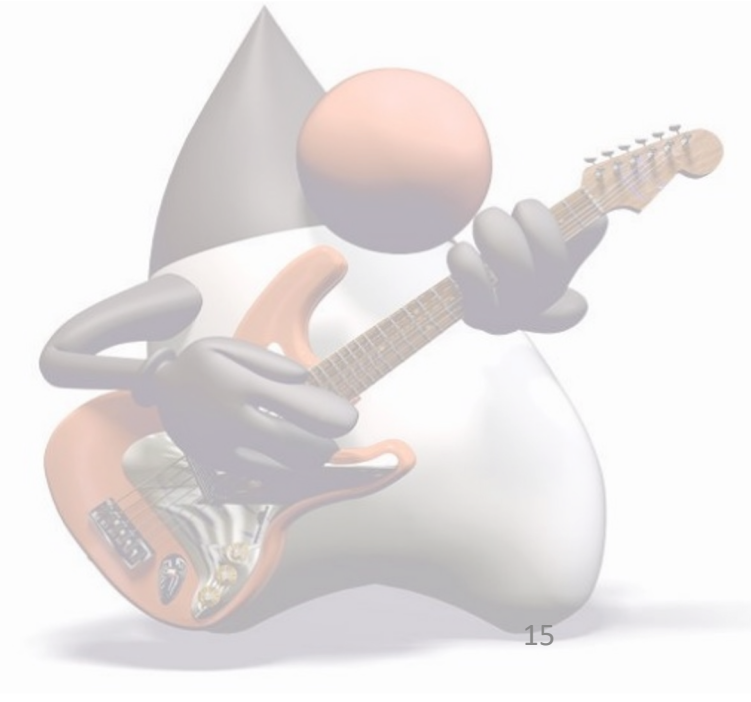
```
42false
        Michael is 30 years old.true or false?
this is a double quote: "
```



Strings

Strings können in Nummern konvertiert werden

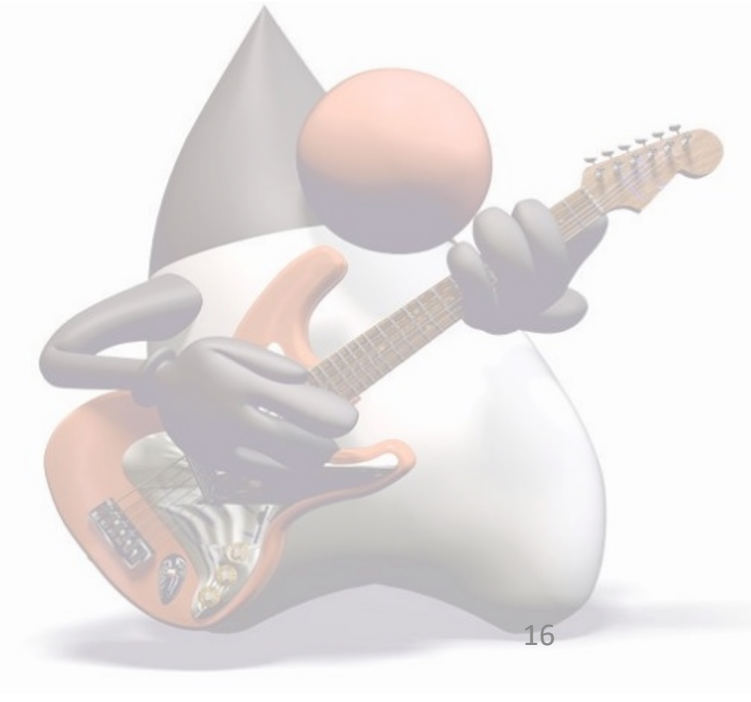
```
public class PrimitiveDataTypes {
        public static void main(String[] args) {
                String intString = "5";
                String floatString = "3.2";
                String booleanString = "true";

                int i = Integer.parseInt(intString);
                long l = Long.parseLong(intString);

                float f = Float.parseFloat(floatString);
                double d = Double.parseDouble(floatString);

                boolean b = Boolean.parseBoolean(booleanString);
        }
}
```



Strings als Textblöcke

Strings können sich auch mehrzeilig als Textblöcke deklariert werden

```
String brief = """
        Lieber Klaus,

        wann wirst Du mir das Buch zurückbringen,
        das Du Dir vor mehr als 2 Jahren von mir ausgeliehen hast?

        Ich bitte gringendst um Rückmeldung
        Roja
      """;
System.out.println(brief);
```



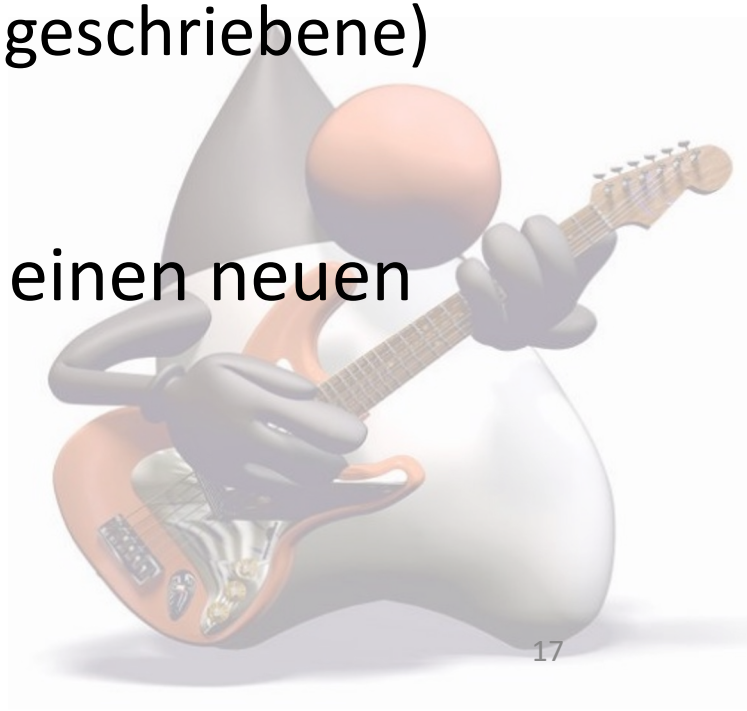
Gruppen von Datentypen in Java

Es gibt in Java zwei Arten von Datentypen:

1. Primitive Datentypen (byte, short, int, long, float, double, char, boolean)

2. Referenz Datentypen (alle Klassen, inklusive selbst geschriebene)

Merke: Beim Erstellen einer Klasse, definiert man einen neuen Datentyp

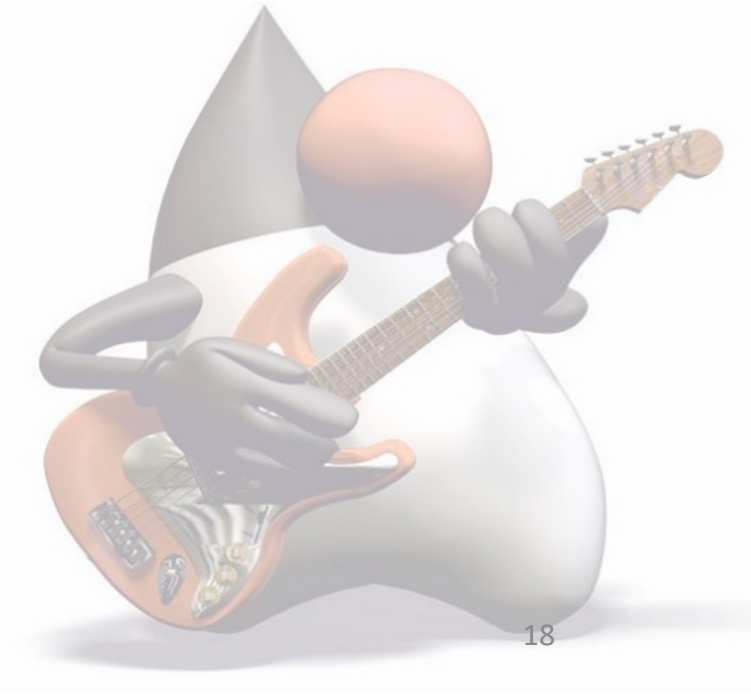


Übung

Schreiben sie ein Programm das den Radius eines Kreises von der Kommandozeile liest und den Flächeninhalt berechnet

Die Syntax für die Formel der Fläche in Java lautet

a = r * r * Math.PI



Übung

Öffne das Java API Doc (suche in Google nach ‘Java 17 API documentation‘)

Suche nach der Math Klasse

Suche nach der Methode „round“

Aufgabe:

Der User soll eine Dezimalzahl eingeben diese soll dann gerundet ausgegeben werden.