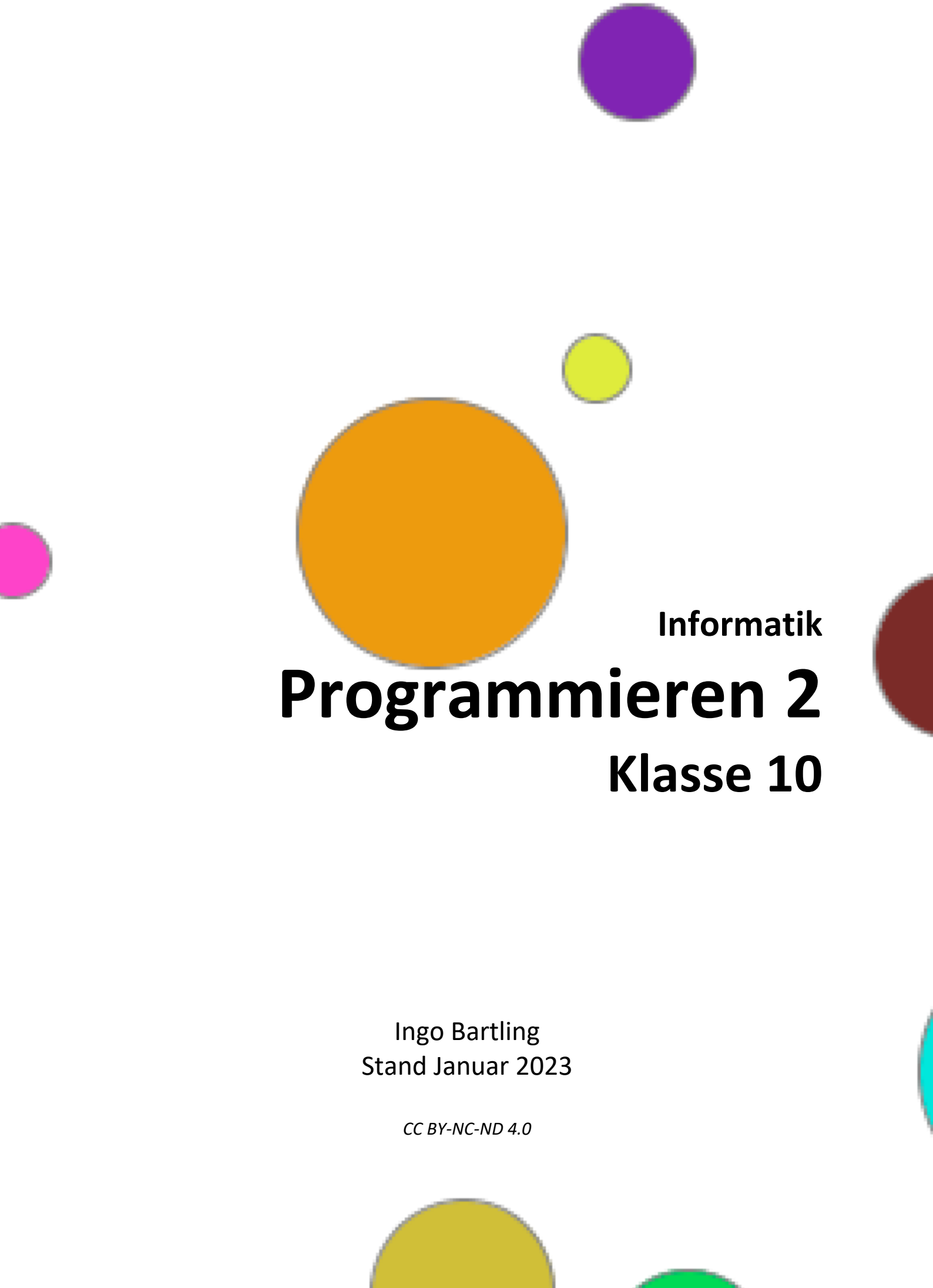


Informatik

Programmieren 2

Klasse 10

Ingo Bartling

Stand Januar 2023

Inhaltsverzeichnis

Einleitung 3
Lehr- & Lernprinzipien 3
Links 3
Aufgabe 1 – Weihnachtsbaum 4
Wiederholung / Einführung processing 6
Aufgabe 2: setup() und draw() 6
Aufgabe 3: Das Koordinatensystem 6
Aufgabe 4: Farben 6
Aufgabe 5: Grundlagen des Zeichnens 7
Aufgabe 6: Strichcode 8
Aufgabe 7: Mondrian 1 9
Aufgabe 8: Mondrian2 10
Aufgabe 9: Was du bis jetzt kennen solltest 11
Anhänge 12
Anhang 1: Einfache Klassendiagramme 12
Anhang 2: Wiederholung 9. JgSt. - Grundwissen 13
Anhang 3: Fach- & Grundbegriffe 16
Anhang 4: Vorgehen bei Programmieraufgaben 17
Anhang 5: Ausgewählte Musterlösungen 18

Einleitung

Dieses Arbeitsheft orientiert sich am LehrplanPLUS (www.lehrplanplus.bayern.de) des bayerischen Gymnasiums und konzentriert sich auf die Programmiersprache JAVA.

*Aufgaben mit Sternchen sind für Schüler*Innen gedacht, die sich noch tiefer mit dem Thema auseinandersetzen oder die benutzte Software besser kennenlernen wollen.*

Lehr- & Lernprinzipien

Programmieren hat aus meiner persönlichen Sicht sehr viel mit Fächern wie Kunst oder Musik zu tun: Die Theorie ist schnellt vermittelt, das bedeutet aber nicht, dass man auch Programmieren (Malen, Zeichnen, Musizieren) kann. Es gehört viel(!) Üben dazu. In der Unterrichtszeit, aber auch zu Hause.

Links

Ein paar Links, die regelmäßig im Unterricht genutzt werden:

Link	**Beschreibung**
processing.org/	Entwicklungsumgebung
http://bluej.org	Entwicklungsumgebung
www.youtube.com/@TheCodingTrain	TheCodingTrain mit den Ressourcen auf thecodingtrain.com/ inkl. dem Buch: https://natureofcode.com/book/
michaelkipp.de/processing/	Schöne Quelle zum Nachlesen und Üben.
www.ingo-bartling.de	Hier finden sich viele Arbeitsblätter zu meinen drei Fächern Mathematik, Physik und Informatik. Manche Arbeitsblätter haben auch hier Eingang gefunden.
openbook.rheinwerk-verlag.de/javainsel/	Komplexes Nachschlagewerk zu Java mit vielen Beispielen

Viel Erfolg und Spaß beim Erlernen des Programmierens.

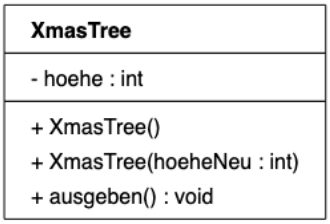
PS: Und um eine Diskussion gleich zu beenden – es gibt keine guten oder schlechten Programmiersprachen. Es gibt nur unpassende Programmiersprachen. Und hier geht es um die Grundlagen. Um einen bestimmten Denkansatz und Grundlagen, die es in so gut wie jeder Programmiersprache gibt.

Aufgabe 1 – Weihnachtsbaum

Bei dieser Aufgabe wiederholst du die für uns wichtigsten Elemente des Programmierens:

- **Modellieren**
- **Definition einer Klasse**
- **Definieren von Variablen**
- **Bedingte Verzweigung (if-else-Struktur)**
- **Zählwiederholung (for-Schleife)**

Erstelle ein Programm mit einer IDE deiner Wahl (BlueJ, processing), die einen Weihnachtsbaum ähnlich zum Nebenstehenden ausgibt.



Erstelle hierzu eine Klasse `XmasTree` gemäß der Klassenkarte. Achte auf eine sinnvolle Benennung von Variablen.

Beginne mit der Klasse und den Konstruktoren. Implementieren sodann die Methode `ausgeben()`.

Hierzu kannst du wie folgt vorgehen:

a) Ausgabe eines einzelnen Zeichens: „#“

b) Ausgabe einer Reihe mit Zählwiederholung: „##########“

c) Ausgabe eines Rechtecks mit 2 geschachtelten Zählwiederholungen:
„##########“
„##########“
„##########“

d) Ausgabe eines Rechtecks in Abhängigkeit des Attributs `hoehe`:
„##########“
„##########“
„##########“

e) Ausgabe eines Dreiecks mit ungerader Zahl:
„#“
„###“
„#####“

f) Zentrieren des Dreiecks mit Leerzeichen (hier als – dargestellt):
„––#“
„–###“
„#####“

g) Erzeugen des zentrieren Dreieckts in Abhängig des Attributs `hoehe`:

```
„       #“
„      ###“
„     #####“
„    #######“
„  #########“
„###########“
```

h) Ergänze einen Stamm:

```
„       #“
„      ###“
„     #####“
„    #######“
„  #########“
„###########“
„      ###     “
„      ###     “
```

i) Ergänze per Zufall[1] ”*“ und ”o” :

```
„       #“
„      *##“
„     ##o##“
„    #o###*#“
„  ##*###o##“
„      ###     “
„      ###     “
```

f*) Wenn du so weit bist, kannst du noch einen Text „HAPPY XMAS“ ergänzen, je nach IDE[2] Farben ergänzen, Animationen etc.

[1] BlueJ: Math.random() ∈ [0;1[ODER processing: random(unten,oben) ∈ [unten;oben[

[2] Integrated Development Environment : Programme (Entwicklungsumgebung) zum Programmieren.

Wiederholung / Einführung processing

Aufgabe 2: setup() und draw()

So gut wie alle processing-Projekte werden mit Hilfe zweier Methoden gesteuert. Erläutere!

Befehl	Fragen	Deine Erläuterung

`void setup() {`
 	
`}`
	Wie oft wird diese Methode aufgerufen?
	_____	

`void draw() {`
 	
`}`
	Wie oft wird diese Methode aufgerufen?
	_____	

Mit Hilfe welchen Befehls kann ein mehrfaches Wiederholen der Methode `draw()`

abgebrochen werden?_____

Aufgabe 3: Das Koordinatensystem

Erläutere das Koordinatensystem (Ursprung, Achsen) in processing!

_____

_____

Aufgabe 4: Farben

In welcher Einheit wird der Bildschirm unterteilt? _____

Erläutere die Bedeutung des Befehls `strokeWeight(x)` und `fill(x)`. Aus welchen Wertebereich darf x gewählt werden?

_____

_____

_____

Erläutere das RGB-Farbmodell am Beispiel `fill(255,0,255)`.

_____

_____

_____

Aufgabe 5: Grundlagen des Zeichnens

Zur Darstellung von Objekten in processing musst du ein paar wichtige Methoden und Funktionen kennen. Ergänze die Tabelle jeweils um eine knappe Erklärung oder gib den Befehl an. Beantworte auch die Fragen. Informationen findest du auf processing.org.

Befehl	**Frage**	**Erläuterung**
size(600,400)		
		Das Zeichenfenster soll genauso groß sein wie der Bildschirm.
frameRate(30)	Wie hoch ist der Standardwert?	
background(0)		
rect(100,200,300,50)	Auf welche Ecke des Objekts bezieht sich die Positionsangabe?	
ellipse(20,50,100,100)		
		Es soll eine Linie vom Punkt (100|200) zum Punkt (300|400) gezogen werden
		x-Koordinate, y-Koordinate der Maus
height width		

Aufgabe 6: Strichcode

a) Lege ein neues processing-Projekt mit dem Namen „Strichcode“ an.

b) Erzeuge ein Fenster in Bildschirmgröße mit schwarzem Hintergrund.

c) Zeichen ein weißes Rechteck über die gesamte Bildschirmhöhe, das 100px breit ist und einen 10px breiten schwarzen Rand besitzt sowie die x-Position 50% von der Bildschirmbreite besitzt.

   Warum ist das Rechteck dennoch nicht genau in der Mitte des Bildschirms?

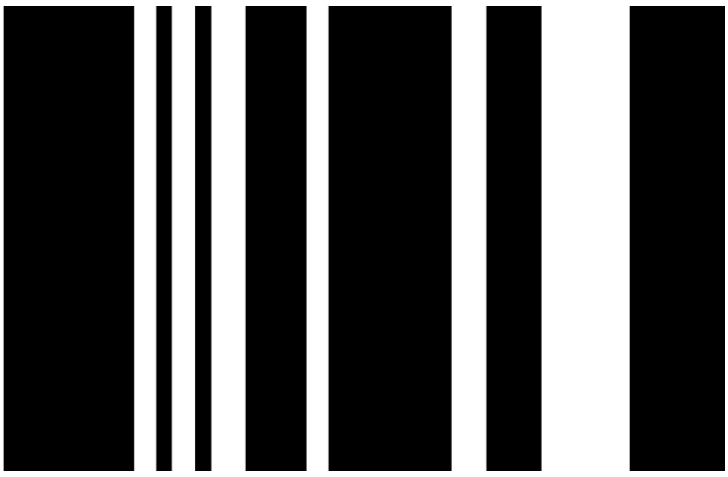
d) Reduziere die `frameRate` auf 5 und lasse bei jedem neuen Bildaufbau ein bildschirmhohes, zufällig x-positioniertes, weißes Rechteck zeichnen.

e) Verändere das Programm so, dass die Rechtecke mit Hilfe der Maus positioniert werden können, aber immer noch bildschirmhoch sind. Die x-Koordinate des Rechtecks entspricht also `mouseX`.

   Ergänze hierzu die Prozedur `void mouseReleased()`, die immer aufgerufen wird, wenn die links Maustaste losgelassen wird.

f) Verändere das Programm so, dass mit 80%iger Wahrscheinlichkeit ein weißes, sonst aber ein schwarzes Rechteck gezeichnet wird.

   Tipp: *random(1,10)<8*
   random(1,10) liefert einen Wert aus dem Intervall [1,10[



Aufgabe 7: Mondrian 1

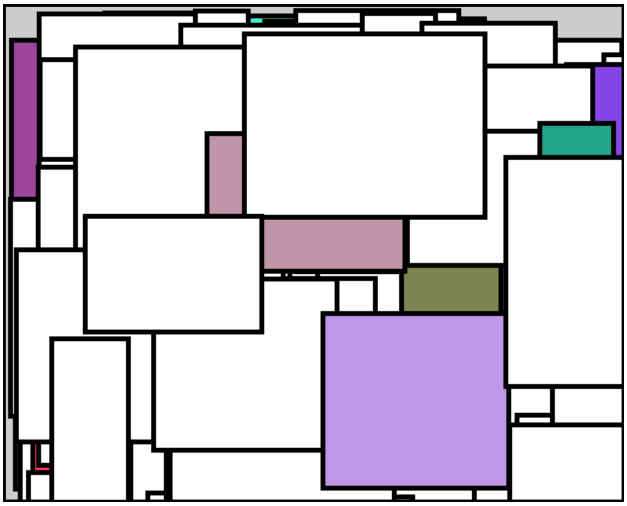
a) Lege ein neues processing-Projekt mit dem Namen „Mondrian1“ an.
b) Erstelle ausschließlich mit den Methoden *setup()* und *draw()* ein Programm mit dem Bilder ähnlich zu Mondrians Bildern erzeugt werden können. Dabei soll ein Fenster der Größe 800x400 mit Rechtecken gefüllt werden, die eine Zufallsfüllfarbe und einen Zufallsbreite sowie –höhe haben. Alle Rechtecke besitzen einen 10px breiten, schwarzen Rand.
c) Die Rechtecke sollen automatisch erzeugt werden, wobei ungefähr pro Sekunde nur 1 Rechteck erzeugt wird.
d) Statt Rechtecke werden Quadrate mit einer Zufallsgröße erzeugt.

Entscheide dich für eine der e) Aufgaben :

e.1) Durch Linksklick mit der Maus soll das Bild gelöscht werden.

ODER

e.2) Durch Klicken mit der linken Maustaste werden Rechtecke / Quadrate an der Mausposition erzeugt.

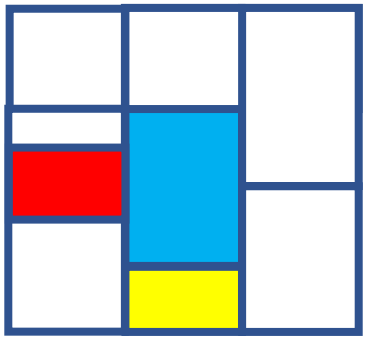


Sternchenaufgaben dienen dazu dein Problemlösungsdenken zu testen und zu schulen. Diese Aufgaben haben ihre Schwierigkeit oftmals in der Frage: *In welchen Schritten löse ich das Problem am einfachsten?* Oftmals hilft es sich zu vorzustellen, wie man in der Realität das Problem lösen könnte: „Erst dann Rand malen und dann die Rechtecke. Oder erst die Rechtecke und dann zum Schluss den Rand übermalen?“

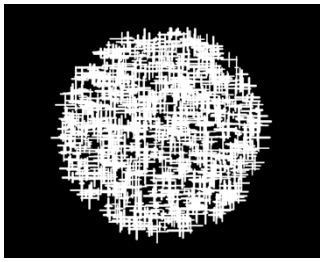
f.1*) Das Bild soll einen schwarzen Rand mit einer Breite von 10px haben.
f.2*) Die Rechtecke solle nur die Farben Weiß, Rot, Gelb und Blau haben.

Sehr schwierige Aufgabe für Profis:

g***) Die Rechtecke werden wie bei einem echten Mondrian nicht nur zufällig gesetzt.
Die Randlinien müssen sich immer treffen.



Aufgabe 8: Mondrian2



Überlege zunächst welche Information du zum Zeichnen eines Kreuzes benötigst. Programmieren sollte immer mit Papier und Bleistift beginnen und nicht am Computer!

a) Lege ein neues processing-Projekt mit dem Namen „Mondrian2“ an.

b) Lege eine neue Klasse „Kreuz“ an. Jedes Kreuz soll als Attribute x und y haben. Jedes Kreuz besteht aus 10px-breiten schwarzen Linien, deren Länge (horizontal wie vertikal) zwischen 10 und 50 Pixeln liegen kann. Definiere passende Attribute für diese zwei Linien.

c) Implementiere einen Standardkonstruktor und einen Nicht-Standardkonstruktor.

d) Implementiere eine Methode gibAus(), die alle Attribute-Werte in der Konsole ausgibt.

e) Implementiere eine Methode show(), die das Kreuz auf Basis der Attribut-Werte zeichnet.

f) Lege in der Projekt-Klasse die Methode setup() und draw() an. Bei Klick mit der linken Maustaste soll an der Maus-Position ein Kreuz gezeichnet werden.

g*) Die Position der Kreuze muss so eingeschränkt werden, dass ähnlich zu nebenstehendem Bild ein Kreis gebildet wird.

h**) Je weiter die Kreuze von Mittelpunkt des Fensters entfernt sind, desto durchsichtiger sollen die Kreuze erscheinen. Benutze hierfür den sogenannten Alpha-Kanal einer Farbe, also z.B. `stroke(255,100)`. 100 bestimmt in diesem Fall die Transparenz. Der Wert geht von 0 bis 255. 0 ist vollständig durchsichtig. 255 ist überhaupt nicht mehr durchsichtig.

i**) Verändere die Transparenz oder Farbe mit der Anzahl der durchlaufenen Frames.

Benutze die processing-Funktion `line(x1,y1,x2,y2)` mit der eine Linie vom Punkt (x1|y1) zum Punkt (x2|y2) gezogen wird.

Für die Teilaufgabe g*) kann die Funktion „dist(x1,y1,x2,y2)“ benutzt werden. Diese berechnet den Abstand zwischen dem Punkt (x1|y1) und dem Punkt (x2|y2).

Für die Teilaufgabe h*) kann zusätzlich die Funktion map() benutzt werden.

Bei der Teilaufgabe i**) kann mit dem Modulooperator % gearbeitet werden, der den Rest einer Division zurückliefert: zum Beispiel:15%256 = 15, 255%256 = 255, 256%256 = 0.

Aufgabe 9: Was du bis jetzt kennen solltest

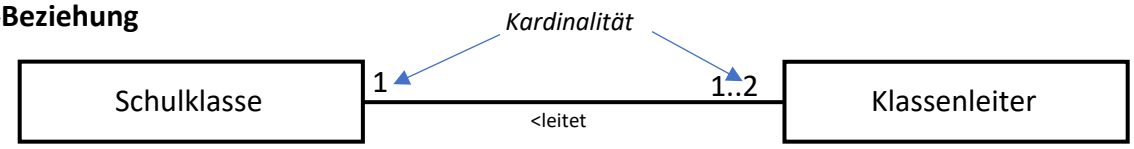
Bei den vorangegangenen Aufgaben hast du einige neue processing-Befehle kennengelernt. Fülle die Tabelle aus, um eine Art Nachschlagewerk zu erhalten. Hilfe und weitere Informationen findest du auf processing.org.

Befehl	**Frage**	**Erläuterung**
random(10)	Welchen Datentyp liefert die Funktion zurück?	
		Eine Zufallszahl zwischen [10,21[
int()		
int(random(1,7))	Zeichne das Datenflussdiagramm:	
map(mouseX,0,width,1,100)		
dist(x1,y1,x2,y2)		

Anhänge

Anhang 1: Einfache Klassendiagramme

1:1-Beziehung

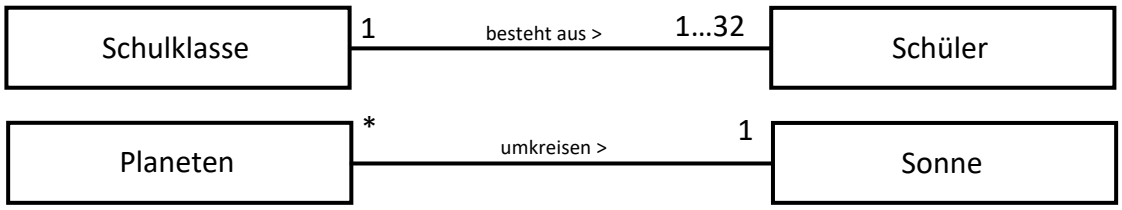


Umsetzung: Schulklasse = {name:String, leitung:Klassenleiter}
Klassenleiter = {name:String, unterrichtsfach:String }

oder

Schulklasse = {name:String }
Klassenleiter = {name:String, unterrichtsfach:String, schueler:Schulklasse }

1:n-Beziehung



Umsetzung: Planet={name:String }
Sonne={name:String, planeten:Planet[] }

Das Array/Liste oä. kommt immer dahin, wo „1“ war!

Anmerkungen:

- Werden Beziehungen komplexer (z.B. eine n:m-Beziehung), so sollte der Einsatz einer DB in Betracht gezogen werden. Es ließe sich aber auch hier eine Art „Vermittlungstabelle“ umsetzen, was aber eher unüblich ist.
- Die Kardinaliäten werden hier oftmals sehr viel genauer angegeben: 0…1 oder 1…32. Das liegt daran, dass es teilweise verschieden Implementierungsvarianten gibt: 1…32 wäre ein Array. 1…* wäre eher eine ArrayList.

Anhang 2: Wiederholung 9. JgSt. - Grundwissen

Um eine gemeinsame Basis zu schaffen, wird das Wissen der neunten Jahrgangstufe hier nochmal zusammengefasst geübt. Auch hier gilt: Es ist wichtig, dass **DU** dich mit dem Thema auseinandersetzt. Das bedeutet aber nicht, dass du es allein machen musst. Frage deine Platznachbarn oder recherchiere im Internet. Diskutiere und vergleiche deine Ergebnisse mit anderen. Informatik ist viel Teamwork, denn Programme entstehen in der Regel immer in Gruppen.

Folgendes Grundwissen solltest du mitbringen:

Modellierung	Analysieren und Übertragen einer Aufgabe in die Informatik: Substantiv -> Klasse oder Attribut Adjektiv -> Attribut (Eigenschaft) Verb -> Methode (Fähigkeit)
„Agiles“ Programmieren	Programme baut man immer schrittweise auf und mit einer kleinen Funktion. Im Laufe der Stunden werden die Programme dann immer komplexer, bekommen mehr Funktionen.
Klassendiagramm	Klassendiagramm bilden den Zusammenhang zwischen Klassen ab. Klassen stehen immer in Rechtecken. Im Themenbereich Programmieren gibt es **einfache Klassendiagramme,** bei denen nur die Klassennamen zzgl. Kardinalität bzw. Vielfachheit und Beziehungsname inkl. Leserichtung angegeben werden. Bei **erweiterten Klassendiagrammen** werden zuzüglich Attribute und Methoden mit Sichtbarkeit, Datentypen und Rückgabewerte in UML-Notation angegeben.
Datentypen	Die drei wichtigsten primitiven Datentypen lauten: **int (ganze Zahl), float (Dezimalbruch),** **boolean (Wahrheitswert),** Ein wichtiger nicht-primitiver Datentyp ist **String (Zeichenkette)** Es gibt weitere Datentypen: double, Date, char,…
Namenskonventionen	Da du im Laufe der Programmierung mehr Lesen als Programmieren wirst, ist es ausgesprochen wichtig, dass dein Quelltext gut lesbar ist. Achte daher auf folgende Punkte: - Benutze keine Umlaute oder Sonderzeichen - Benutze kein Leerzeichen in Namen, benutze ein _ - Klassennamen beginnen mit einem Großbuchstaben - Attribute und Methoden beginnen mit einem Kleinbuchstaben - Nur Konstanten wie PI werden ausschließlich mit Großbuchstaben geschrieben.
Kommentare	Einzeiliger Kommentar: `//Kommentar` Mehrzeiliger Kommentar: `/* Kommentare */`

Sichtbarkeit	Attribute und Methoden können sein: **public** + : Jeder kann darauf zugreifen **private** - : Nur die Klasse selbst kann darauf zugreifen **protected** # : Nur Klassen des selben Pakets können darauf zugreifen In der Regel wird bei uns alles public gesetzt, da man sich dadurch viel Aufwand spart und nur die wirklich „gefährlichen“ Attribute bzw. Methode werden geschützt (vgl. Python).

Klassen	Klassen werden wie folgt definiert:
	`class KlassenName { //Klassenkopf`
	`  //Klassenrumpf`
	`}`

Attribute	Nach dem Klassenkopf werden im Klassenrumpf zunächst alle Attribute (Eigenschaften) aufgelistet.
Konstruktor	Konstruktoren sind spezielle Methoden: Der Name ist identisch mit dem Klassenname und es gibt keinen Rückgabewert. Man unterscheidet zwischen **Standardkonstruktoren**, die eine leere Parameterliste haben und **Nicht-Standardkonstruktoren**, die Parameter haben (s. Aufgabe 1) Konstruktoren werden direkt im Anschluss an die Attribute angegeben.
Methoden	Methoden haben immer einen Rückgabewert (void, int,…) und können Parameter besitzen. Sie haben eine Methodenkopf und einen Methodenrumpf. Der Name sollte ein Verb sein, aus dem man die Aufgabe der Methode schließen kann.

Prozedur	Eine Prozedur ist eine Methode, die als „Rückgabetyp“ *void* („Leere“) besitzt.
	`void ausgeben() {`
	`  System.out.println(„Hallo“);`
	`}`

Funktion	Eine Funktion ist eine Methode, die als Rückgabetyp etwas anderes als *void* hat und daher in der letzten Zeile mit return beginnt.
	`int addieren(int zahl1, int zahl2) {`
	`  return zahl1+zahl2;`
	`}`
	`zahl1` und `zahl2` wären hier **Parameter**.

Variablen	Variablen werden **deklariert** und dann **initialisiert**. Beides zusammen nennt man **definieren**. Primitive Datentypen werden bei der Deklaration bereits initialisiert, da ein sinnvoller Standard-Wert vergeben werden kann – meist 0. Andere Datentypen haben **null**, also „Nichts“ als Startwert.

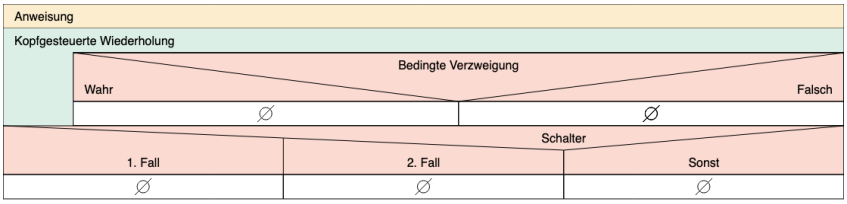
Bedingte Verzweigung	`if (Bedingung) {`
	`  //Bedingung ist wahr`
	`} else {`
	`  //Bedingung ist falsch`
	`}`
	Bedingungen könne mit && (und) bzw. || (oder) kombiniert werden. && bindet dabei stärker. Der else-Zweig muss nicht aufgeführt werden.

Bedingte Verzweigung	`switch (variable) {`
	`  case X:  //variable == X`
	`  break;`
	`  case Y:  //variable == Y`
	`  break;`
	`  default:  //Standardreaktion`
	`  default;`
	`}`
	Jeder Fall (case) wird in der Regel mit break; beendet, um weitere Prüfungen zu verhinden – insbesondere den default-Fall.

Zählwiederholung	Kopfgesteuerte Wiederholung
	
	`for (Def. Zählvariable; Bedingung; Schrittweite) {`
	`  //Schleifenrumpf`
	`}`

Bedingte Wiederholung	Kopfgesteuerte Wiederholung
	
	`while (Bedingung) {`
	`  //Schleifenrumpf`
	`}`

Operatoren	Mathematische Operatoren:
	`+, -, *, /, % (Modulo: 7%5=2), ++, --, -=, +=`
	
	Vergleichsoperatoren:
	`==, ! (Nein oder Nicht), !=, <, >, <=, >=`

Struktogramme[3]	Um Programmierabläufe bzw. Algorithmen sprachunabhängig darzustellen kann man sich sogenannter Struktogramme bedienen. Diese können fast beliebig in- und aufeinander geschachtelt werden. 

[3] Online-Programm zum Erstellen von Struktogrammen: https://dditools.inf.tu-dresden.de/struktog/index.html

Anhang 3: Fach- & Grundbegriffe

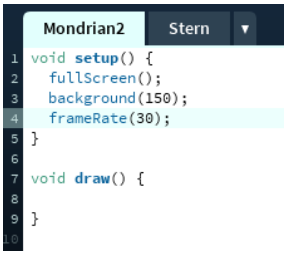
Ergänze hier nach eigenem Ermessen dir unbekannte Begriffe und wichtige neue Vokabeln.

Fachbegriff	**Erklärung**
Modellierung	Analysieren und Übertragen einer Aufgabe in die Informatik: Substantiv -> Klasse oder Attribut Adjektiv -> Attribut Verb -> Methode
Klassendiagramm	Klassendiagramm bilden den Zusammenhang zwischen Klassen ab. Klassen stehen immer in Rechtecken. Im Themenbereich DB werden idR. nur Klassennamen angegeben zzgl. Kardinalität und Beziehungsname inkl. Leserichtung
Relation / Klasse	„Relation“ kann synonym zu „Tabelle“ benutzt und als Karteikasten mit Karteikarten (Datensatz) vorgestellt werden. Eine Datenbank besteht aus mindestens einer Relation. Diese muss einen Primärschlüssel besitzen. Klassen habe zusätzlich Methoden.
Relationenschema	Eine Relationenschema gibt den Relationenname zzgl. der Feld-/Attributnamen eventuell mit Datentyp an: Tier = {nr:int, name:String, gehege:Gehege}
Datentypen	Die vier wichtigsten Datentypen lauten: int (ganze Zahl), float (Dezimalbruch), boolean (Wahrheitswert), String (Zeichenkette) Es gibt weitere Datentypen: double, Date, char,…
Referenzattribut	Entspricht in etwa dem Fremdschlüssel bei Datenbanken.
1:1-Beziehung	Referenzattribut kann entweder bei „Spieler“ oder bei „Gegenstand“ stehen.
1:n-Beziehung	Das Referenzattribut steht idR immer auf der Seite des „1“ – anders als bei Datenbanken! Es wird durch Arrays, ArrayLists oder Listen umgesetzt, die mehrere Objekte speichern können.
n:m-Beziehung	Werden genauso wie im Themenbereich Datenbanken die Vermittlungsrelationen hier über sogenannte Assoziationsklassen gelöst. Ist aber eher ungewöhnlich. Man sollte in diesen Fällen den Einsatz einer DB in Erwägung ziehen.

Anhang 4: Vorgehen bei Programmieraufgaben

1. Lese die ganze Aufgabenstellung.
2. Lies die Aufgabenstellung nochmals und markiere alle Substantive, Adjektive und Verben unter folgendem Aspekt:
   Substantiv – mögliche Klasse
   Adjektiv – mögliches Attribut einer Klasse
   Verb – mögliche Methode einer Klasse
3. Lege ein neues processing-Projekt an und definiere die Methoden setup() und draw() im ersten Reiter des Projekts.
4. Ähnlich zum Kochen erfolgt nun das sogenannte "Mise-en-place". Ergänze die setup()-Methode so, dass die äußeren Rahmenbedingungen für dein Projekt gegeben sind:
   - Fenstergröße
   - (Hintergrund)-Farbe(n)
   - frameRate(30) etc.

   Lege für jeden Referenz-Datentyp eine neue, aber noch leere Klasse in einem eigenen Reiter an.



5. Beginne nun die eigentliche Aufgabe zu lösen, indem du möglichst einfach beginnst und schrittweise dein Programm erweiterst, verallgemeinerst und um weitere Funktionen ergänzt.

   Beachte: - Wechsle erst in eine neue Zeile wechselst, wenn die aktuelle Zeile korrekt ist.
   - Achte darauf, dass dein Programm immer funktioniert.

Beispiel an der Aufgabe „Mondrian 2"

1. Definiere die Methode setup() mit Methodenrumpf und draw(). Lege eine Klasse „Kreuz" an.
2. Erzeuge innerhalb der Methode draw() ein Kreuz an einer speziellen Stelle z.B. (100|100)
3. Lass dieses Kreuz an der Stelle eines Mausklicks erzeugen.
4. Passe nun die Klasse Kreuz so an, dass mit Hilfe des Standardkonstruktors „Kreuz()" ein Kreuz an der Stelle (100|100) erzeugt wird. Ergänze hierzu die Klasse um die benötigten Attribute, den Standardkonstruktor und eine Methode show(). show() ist dabei nahezu identisch zu dem Programmcode aus Punkt 2.
5. Passe nun die draw()-Methode an:

```
void draw(){
    Kreuz k1 = new Kreuz();
    k1.show();
}
```

Anhang 5: Ausgewählte Musterlösungen