

**KLAUSUR**  
**PROGRAMMIERUNG 2**

**21. JULI 2021**

Bedingungen der Klausur:

1. Fragen stellen Sie bitte im Zoom-Chat.
2. Bitte im Source-Code nicht Ihren Namen vermerken (also nicht `@author`-Tag o.ä.). Die Klausuren werden anonym kontrolliert.
3. Am Ende der Prüfung: Öffnen Sie den Dateexplorer/Finder und wechseln Sie in Ihren Workspace. Laden Sie entweder alle `*.java`-Dateien aus dem package `klausur` hoch oder zippen Sie den package-Ordner und laden Sie die zip-Datei in Moodle hoch (bei Aufgabe Klausur1PZ)!
4. Es sind insgesamt 59 Punkte zu erzielen (Teil 1: 15 Pkt., Teil 2: 15 Pkt., Teil 3: 15 Pkt., Teil 4: 10 Pkt., fehlerfreies Programm: 4 Pkt.).
5. Schreiben Sie Ihre Klassen im package `klausur`!

Notenspiegel:

Note	1,0	1,3	1,7	2,0	2,3	2,7	3,0	3,3	3,7	4,0	5,0
Punkte	>=	55.5	52.5	49.5	46.5	43.5	40.5	37.5	34.5	31.5	<=
	56.0	– 53.0	– 50.0	– 47.0	– 44.0	– 41.0	– 38.0	– 35.0	– 32.0	– 29.5	29.0

**Teil 1 (Fensterklasse)**

**15 Punkte**

Erstellen Sie eine Klasse <code>Klausur1PZ</code> . Diese Klasse stellt Ihre Fensterklasse dar, d.h. sie erbt von <code>JFrame</code> . Es muss gezeichnet werden. Die zu erstellende Oberfläche sieht wie folgt aus:	4 Pkt.
---	--------



Dabei sind folgende Dinge zu beachten (nächsten drei Tabellenzeilen):

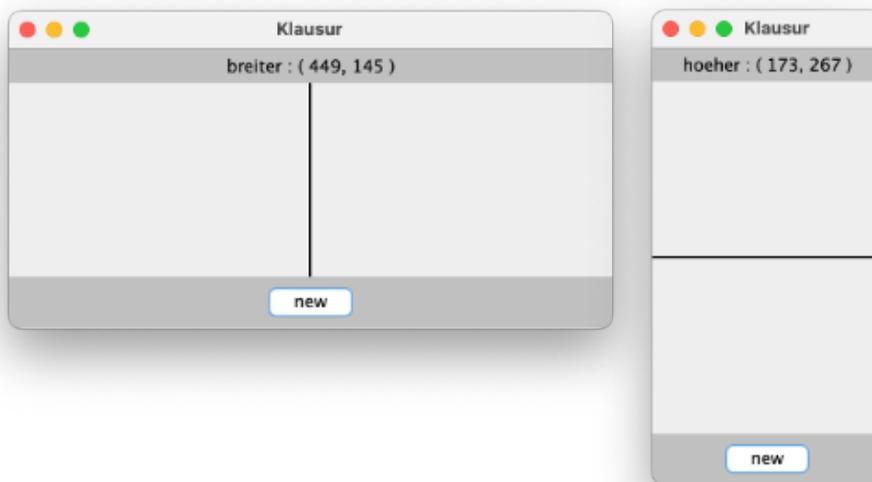
Oben im Fenster ist ein **JPanel**,

- dessen Hintergrundfarbe **LIGHT\_GRAY** ist und
- das ein **JLabel** enthält.

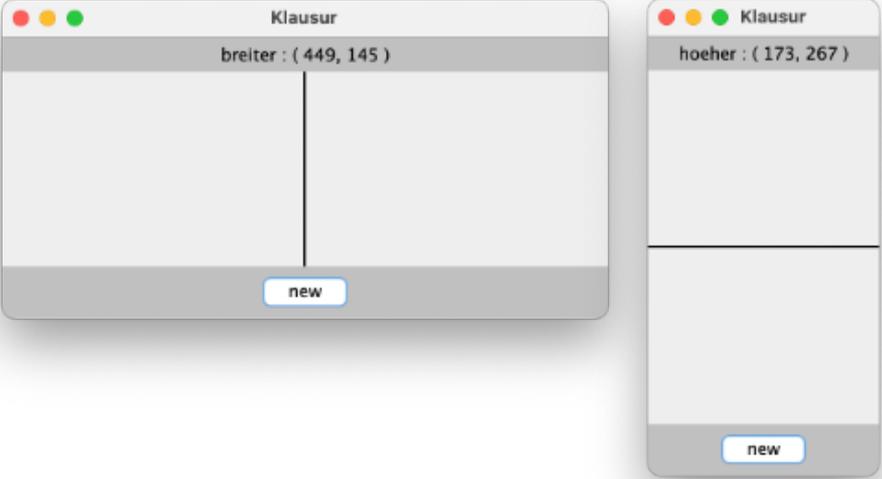
Die Beschriftung des **JLabels** sollten Sie in der **paintComponent()**-Methode vornehmen, denn das **JLabel** zeigt

- an, ob die Zeichenfläche (Canvas) breiter als hoch ist (**breiter**) oder umgedreht (**hoeher**) und
- die Breite der Canvas sowie die Höhe (**breite**, **hoehe**)

Es gibt also diese beiden Möglichkeiten:



5 Pkt.

<p>Achten Sie auch darauf, dass der Text des <b>JLabels</b> angepasst wird, wenn Sie die Größe des Fensters ändern (ergibt sich aber automatisch, wenn Sie den Text des <b>JLabels</b> in der <b>paintComponent()</b>-Methode setzen).</p>	
<p>In der Mitte des Fensters ist die Zeichenfläche (Canvas).</p> <ul style="list-style-type: none"> <li>• Es wird eine Linie dargestellt.</li> <li>• Diese Linie ist in Strichstärke 2.0f.</li> <li>• Wenn die Canvas breiter als hoch ist, dann verläuft die Linie in der Mitte der Breite vertikal.</li> <li>• Wenn die Canvas höher als breit ist, dann verläuft die Linie in der Mitte der Höhe horizontal.</li> <li>• Der Fall Höhe==Breite muss nicht extra behandelt werden.</li> </ul> <p>Es gibt also diese beiden Möglichkeiten:</p> <div style="text-align: center;">  </div>	<p>4 Pkt.</p>
<p>Unten im Fenster ist ein <b>JPanel</b>,</p> <ul style="list-style-type: none"> <li>• dessen Hintergrundfarbe <b>LIGHT_GRAY</b> ist und</li> <li>• das ein <b>JButton</b> mit dem Text <b>new</b> enthält.</li> </ul>	<p>2 Pkt.</p>

**Teil 2 (Der new-Button)**

**15 Punkte**

<p>Implementieren Sie eine Klasse <b>Figure</b> mit drei <b>privaten</b> (!) Objektvariablen</p> <ul style="list-style-type: none"> <li>• <code>int x</code></li> <li>• <code>int y</code></li> </ul>	<p>5 Pkt.</p>
---	---------------

- `int length`

Schreiben Sie einen parametrisierten Konstruktor `Figure(int x, int y, int length)`, der die Objektvariablen mit den Parameterwerten initialisiert.

Schreiben Sie für alle drei Objektvariablen jeweils `Getter` und `Setter`.

Implementieren Sie für den Button den `ActionListener` so, dass durch den Klick auf den Button

- in die eine Hälfte der Canvas ein gelbes Quadrat (Farbe ist YELLOW) und
- in die andere Hälfte der Canvas ein grüner Kreis /Farbe ist GREEN) gezeichnet wird.
- Beide Objekte sind vom Typ `Figure`.
- `length` des Quadrates entspricht der Seitenlänge, `length` des Kreises entspricht dem Durchmesser.
- `x` und `y` sind jeweils die Koordinaten der linken oberen „Ecke“.

Beachten Sie, dass `length` bei Kreis und Quadrat gleich sind und dass sich die Länge möglichst gut (**ca. 90%**) in die Hälfte der `Canvas` einpasst, d.h. Sie müssen schauen, dass das Quadrat und der Kreis stets vollständig in ihre Hälfte passen, aber bestmöglich.

Außerdem sollen die beiden Figuren möglichst mittig in ihrer jeweiligen Hälfte angeordnet sein.



Wenn Sie die Größe des Fensters ändern, dann müssen sich Quadrat und Kreis nicht mitändern! Wenn Sie dann aber wieder auf den **new**-Button klicken, dann werden die beiden Figuren wieder an die neuen **Canvas**-Dimensionen angepasst.

### Teil 3 (MouseListener und MouseMotionListener)

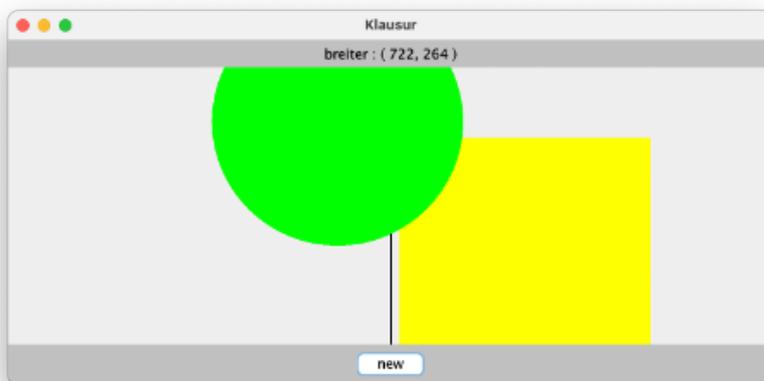
15 Punkte

Implementieren Sie **MouseListener** und **MouseMotionListener** so, dass Sie entweder den Kreis oder das Quadrat bei gedrückter Maustaste bewegen können, je nachdem, ob Sie auf das Quadrat oder auf den Kreis mit der Maus geklickt haben.

15 Pkt.

Wenn Sie weder das Quadrat noch den Kreis durch den Mausklick getroffen haben, dann soll sich auch nichts bewegen. Bei Kreis betrachten Sie das Tangentenquadrat um den Kreis, um zu prüfen, ob Sie den Kreis getroffen haben (also genauso, wie beim Quadrat).

Sie können Kreis und Quadrat jeweils auch mehrmals hintereinander bewegen und/oder abwechselnd. Sie können nur nie beide Figuren zugleich bewegen (selbst wenn sie übereinander sind). In der folgenden Abbildung wurde sowohl der Kreis als auch das Quadrat bereits (evtl. mehrfach) bewegt:

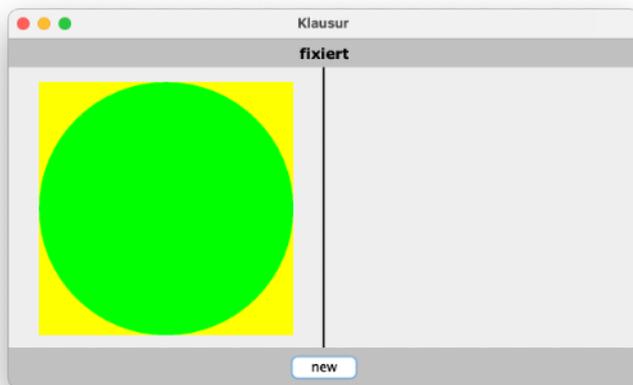


### Teil 4 (Fixieren)

10 Punkte

Wenn Sie den Kreis fast vollständig über das Quadrat bewegt haben oder das Quadrat fast vollständig über den Kreis, dann wird das erkannt und der Kreis wird exakt in das Quadrat fixiert. Das bedeutet, es entsteht z.B. folgendes Bild:

10 Pkt.



- **Fast** vollständig bedeutet, dass sich die **x**- und **y**-Koordinaten der beiden Figuren um jeweils höchstens 30 Punkte unterscheiden.
- Sind die Figuren innerhalb dieses Abstandes, dann werden Sie automatisch exakt übereinandergelegt.
- Sind die beiden Figuren exakt übereinander, kann keine der beiden Figuren mehr bewegt werden.
- Es kann nur noch der Button **new** geklickt werden, um die Ausgangssituation wieder herzustellen.
- Im Label oben erscheint „**fixiert**“. Der Text ist fettgedruckt (bold).

**Viel Erfolg!**