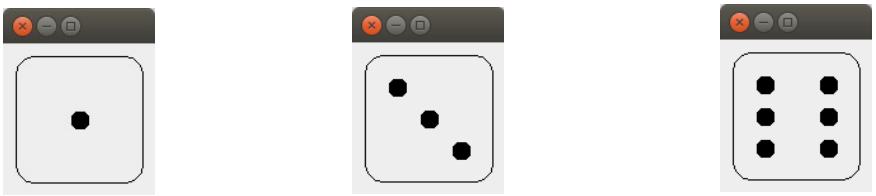
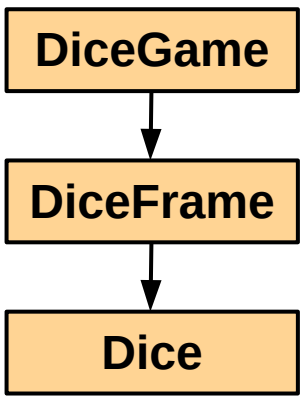
2D-Grafiken

1. Würfel

1 Darstellung



1. Ziel dieser Übung ist es, ein Programm zu schreiben, welches beim Ausführen einen grafischen Würfel darstellt. Die Augenzahl des Würfels soll bei jedem Start zufällig gewählt werden.
2. Für den Aufbau dieses Projektes soll folgende Struktur verwendet werden:



DiceGame: Enthält die Main-Methode. Instanziiert *DiceFrame*.

DiceFrame: Erbt von *JFrame*. Über den Konstruktor wird die Grösse und Sichtbarkeit des Fensters gesetzt. Zudem wird *Dice* instanziiert. Besitzt die Methode *paint(Graphics g)*.

Dice: Über den Konstruktor wird die Position des Würfels gesetzt. Besitzt mehrere Methoden um den Würfel zu zeichnen.

3. Programmieren Sie die Klasse *DiceGame* wie beschrieben aus.
4. Programmieren Sie die Klasse *DiceFrame* wie beschrieben aus. Den Inhalt von der Methode *paint(Graphics g)* können Sie noch leer lassen.
5. Programmieren Sie die Klasse *Dice* wie beschrieben aus. Schreiben Sie eine Methode *drawDice(Graphics g)*, lassen Sie auch hier momentan den Inhalt aus. Rufen Sie diese Methode in der Klasse *DiceFrame* in der *paint(Graphics g)* auf und übergeben Sie das Objekt *g*.
6. Schreiben Sie in der Klasse *Dice* die Methode *drawBody(Graphics g)*. Diese Methode soll den Würfel ohne die Augen (nur den Rahmen) zeichnen. Diese Methode können Sie in der Methode *drawDice(Graphics g)* aufrufen.
7. Schreiben Sie in der Klasse *Dice* die Methode *drawDots(Graphics g, int number)*. Diese Methode soll die Augen auf dem Würfel zeichnen. Dabei soll mit *number* die Augenanzahl gewählt werden können. Diese Methode können Sie, wie die Methode *drawBody(Graphics g)*, in der Methode *drawDice(Graphics g)* aufrufen.
8. Schreiben Sie in der Klasse *Dice* die Methode *rollDice()* welche eine Zahl zwischen 1 und 6 generiert und diese für das Darstellen des Würfels zwischenspeichert. Implementieren Sie diesen **zwischengespeicherten** Wert **beim Aufruf** der Methode *drawDots()*.

   Tipp: Mit der folgenden Methode können Sie einen Zufallswert generieren:

   ```
   public int getRandom(int min, int max){
         return min + (int)(Math.random() * ((max - min) + 1));
   }
   ```

9. Rufen Sie die Methode *drawDice(Graphics g)* in der Methode *paint(Graphics g)* in der Klasse *DiceFrame* auf. Testen Sie Ihre implementiertung.

2 Eingabe

1. Nun soll Ihr Programm nach dem Starten bei der Eingabe einer Taste einen neuen Würfelwert generieren.

   Erweitern Sie Ihr Programm so, dass der Würfel jedes mal beim Aufrufen von *paint* einen weissen Hintergrund zeichnet. Dies verhindert, dass mehrere Würfel übereinander gezeichnet werden:

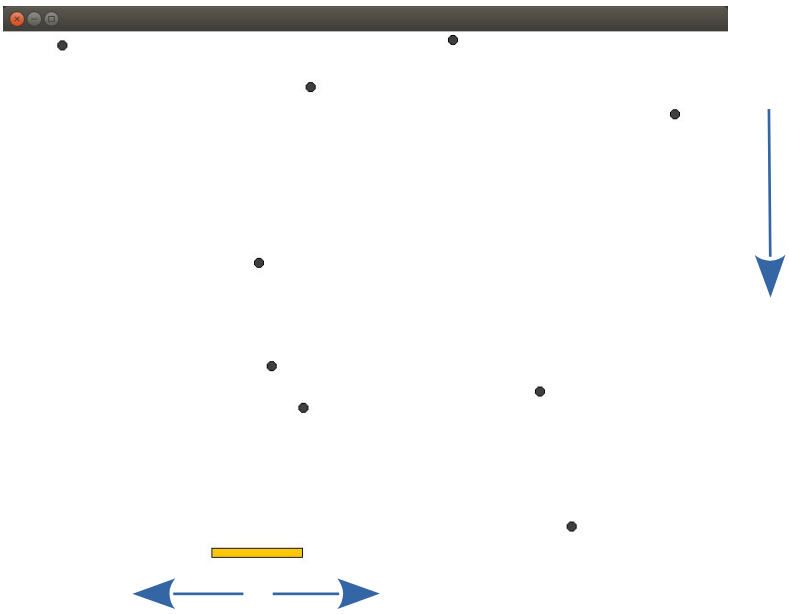
   

2. Erstellen Sie eine neue Klasse *Keys* und erben Sie von der zur Verfügung gestellten Klasse *SimplifiedListener*. Diese Klasse erlaubt es Ihnen die Methoden zum Abfangen der Tasteneingabe zu überschreiben.
3. Setzten Sie in der Klasse *DiceGame* Ihr „frame“ (Fenster) in eine statische Variable die ausserhalb der Main-Methode deklariert wurde. Dadurch können Sie dann später auf Ihr Fenster von einem beliebigen Punkt zugreifen (globaler Zugriff).
4. In der Klasse *Keys* können Sie nun die Methode *public void keyReleased(KeyEvent e)* überschreiben. Diese Methode wird nach der entsprechenden Registrierung jedes mal dann aufgerufen, sobald eine Taste nach dem Drücken wieder losgelassen wird. Im Parameter *e* ist die losgelassene Taste hinterlegt. Mit *e.getKeyCode()* erhalten Sie die ID der Taste (Zahlenwert). Diese ID können Sie direkt mit den statischen Variablen aus der Klasse *KeyEvent* vergleichen. Z.B. ist die ID der Taste „Space“ in der Variable *KeyEvent.VK_SPACE* hinterlegt. In der entsprechenden Java-Doc können Sie alle anderen vorgefertigten Variablen entnehmen. Beachten Sie, dass die Klasse *KeyEvent* unter dem Namensraum *java.awt.event.KeyEvent* abgelegt ist.
5. Programmieren Sie nun mithilfe des Parameters *e* die Funktionalität ein, dass nach dem Drücken der Taste „Space“ der Würfel neu gewürfelt wird. Nach dem Würfeln müssen Sie über Ihr Fenster die geerbte Methode *repaint()* aufrufen. Diesen Aufruf können Sie direkt in der Tastenabfrage durchführen. Dadurch wird die *paint*-Methode neu aufgerufen.

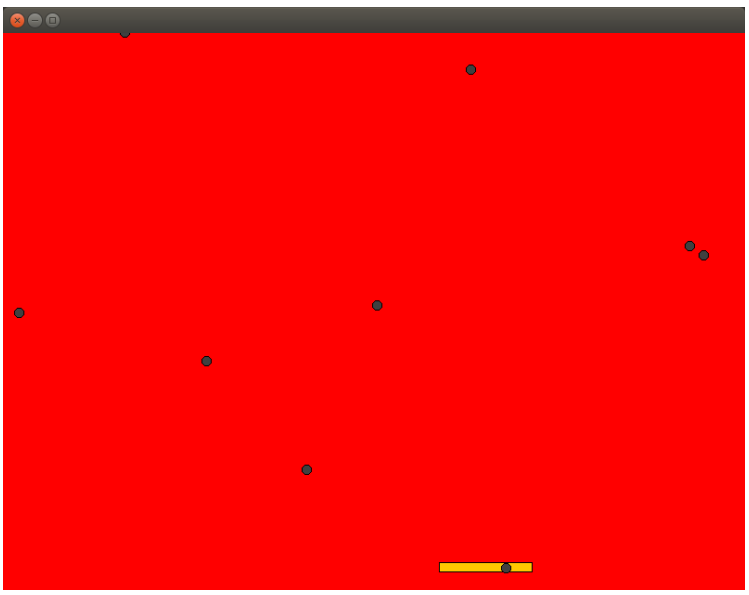
   Tipp: Durch den Punkt 4 können Sie über Ihre *DiceGame*-Klasse auf Ihr Fenster und den entsprechenden Würfel zugreifen: *DiceGame.frame.dice…*
6. In Ihrem Fenster können Sie nun mit der geerbten Methode *addKeyListener* ein instanziiertes Objekt der Klasse *Keys* übergeben (registrieren). Ab diesem Schritt werden Ihre überschriebenen Methoden aus *Keys* beim Loslassen einer Taste aufgerufen.
7. Testen Sie Ihre Implementierung.

2. Meteoriten-Spiel

Programmieren Sie mit dem bisherigen Wissen ein Meteoriten-Spiel. Der Spieler wird als ein Rechteck dargestellt und muss herunterfallende Meteoriten (Kreise) ausweichen. Der Spieler kann sich dabei nur durch Tasteneingabe nach rechts oder links bewegen.



Wird der Spieler von einem Meteoriten getroffen, wird der Hintergrund rot und die Meteoriten angehalten:



Hinweise

1. Die Positionierung der Meteoriten sollte zufällig gewählt werden. Sie können dazu die Random-Methode aus dem Würfel-Programm verwenden.
2. Wenn ein Meteorit unter dem Fensterrand fällt, kann dieser wieder über dem Fensterrand mit einem zufälligen X-Wert platziert werden.
3. Sie werden für das Fallen und das Wieder-Zeichnen eine endlose Schleife benötigen.
4. Ihre Endlos-Schleife sollte nach jedem Durchlauf eine Pause einlegen. Ansonsten wird diese zu schnell ausgeführt. Nutzen Sie dazu die Vorgefertigte Klasse *SimplifiedDelay*. Diese beinhaltet eine statische Methode *delay(int milliseconds)*, welches Ihr Programm entsprechend pausieren wird.
5. In der Klasse *SimplifiedListener* ist die Methode *public void keyPressed(KeyEvent e)* vorhanden, welche Sie für die Tasteneingabe nutzen können.