Die Vorbereitung und die GUI:

1. Erkundige Dich über das Spiel Doodle-Jump. Wir möchten dieses Spiel in seinen Grundzügen für den PC nachprogrammieren. Erstelle eine kurze Spielbeschreibung! Beantworte die folgenden Fragen: Welche Positionen im Spielfeld erreicht die Figur. Wie hoch springt die Figur? Wie viele Stufen gibt es in im Spielfeld. Wie schnell bewegt sich die Spielfigur? Usw.
2. Überlege Dir eine Spielfeldgröße in Pixeln in Breite und Höhe, z. B. 400 Pixel breit, 800 Pixel hoch! Erstelle in Deinem Heft eine maßstabsgetreue Skizze des Spielfeldes, der Hauptfigur und mehrerer Stufen.
3. Male mit einem Zeichenprogramm ein Bild für den Hintergrund. Speicher dieses Bild im Dateiformat .jpg oder .gif.
4. Zeichne mit einem Zeichenprogramm ein Bild für eine Spielfigur. Überlege Dir hier auch die Größe der Spielfigur, z. B. 32 x 64 Pixel. Speichere dieses Bild ebenfalls im Dateiformat .jpg oder .gif ab. Manche Zeichenprogramme erlauben, ein Bild ohne Hintergrund (mit transparentem Hintergrund) zu erstellen. Dafür ist es erforderlich, das Bild als .gif-Bild abzuspeichern.
5. Zeichne mit einem Zeichenprogramm ein Bild für eine Stufe, auf der die Person springen soll. Überlege Dir auch hier eine passende Größe der Stufe. Auf den Stufen soll unsere Hauptfigur später nach oben springen.
6. Öffne das Programmbeispiel „Schiller_Jump01“. Inspiziere den Quelltext. In der Klasse Fenster wird das Fenster erzeugt, der Fensterinhalt ergibt sich aus einem Objekt der Klasse Picturepanel. In der Methode paintComponent() der Klasse Picturepanel kannst du zeichnen. Probiere es aus.
7. Kopiere die Bilder, die du selbst erstellt hast, in den Programmordner und passe die Namen der Bilder in der Klasse Picturepanel an, so dass Deine drei Bilder geladen werden.

Unsere Grafik soll von der Logik unabhängig sein. Wir erzeugen also eine Klasse Doodle, so nennen wir unsere Hauptfigur. Von unserem Doodle wissen wir, dass er seine Position verändert. Also benötigen wir zwei Attribute x und y, um seine Koordinaten zu speichern. Die Methode bewegen soll seine y-Koordinaten verändern. Doodle soll in hie Höhe springen und dann wieder nach unten fallen. Die Sprunghöhe geben wir als Attribut an. In diesem Beispiel nennen wir dieses Attribut energie.

Wir starten mit einem Wert für das Attribut von z. B. 100, das soll bedeuten, dass Doodle 100 Pixel in Y-Richtung springen kann. Danach soll Doodle wieder nach unten fallen, bis der Boden erreicht ist. Momentan soll dabei das Spiel noch nicht zu Ende sein, sondern Doodle erhält erneut eine Energie von 100.

8. Erzeuge einen Klasse Doodle mit den Attributen int x, int y und int energie. Gib dem Doodle im Konstruktor Startwerte für die Attribute. Schreibe die Set- und Get-Methoden für den Doodle und zusätzlich eine Methode public void bewegen(). In dieser Methode prüfst Du, ob unser Doodle eine positive Energie hat und bewegst ihn einen kleinen Schritt (z. B. 5 Pixel) nach oben bzw. nach unten.
9. Rufe in der Programmschleife der Klasse Fenster die Methode bewegen() des Doodles hinzu. Dazu musst du ein Referenzattribut anlegen und das Doodle-Objekt mit new erzeugen.
10. Unser Doodle würde nun immer weiter fallen. Ergänze in der Programmschleife eine Anweisung, mit dem Du unserem Doodle wieder Enegie gibst, wenn er den unteren Rand des

Fensters erreicht hat. Damit beginnt unser Doodle immer wieder zu hüpfen, wenn er den Boden erreicht hat.

11. Schreibe in der Klasse Fenster eine Methode **`public Doodle getDoodle()`**, mit der Du eine Referenz auf das Doodle-Objekt erhältst. Diese Methode benötigen wir, um die nächste Aufgabe zu lösen.
12. Ergänze in der Klasse Picturepanel die Methode paintComponent, dass unser Doodle jetzt immer an der richtigen Y-Koordinate gezeichnet wird.
13. Ergänze in der Klasse Picturepanel die Methode paintComponent, dass unser Doodle jetzt immer an der richtigen X-Koordinate gezeichnet wird. Diese soll mit der Maus-Koordinate in X-Richtung identisch sein.
14. Teste das Programm ausgiebig und verändere verschiedene Attributwerte und Parameter. Die Lösung kannst du mit dem Programm Schiller_Jump02 vergleichen.

Ähnlich wie unseren Doodle wollen wir auch Stufen erzeugen. Zunächst genügt uns eine Stufe, die sich von oben nach unten bewegt. Unten angekommen soll sie einfach stehen bleiben. Die X-Position der Stufe soll allerdings zufällig gewählt werden. Dazu erstellen wir eine Klasse Zufallsgenerator. In ihr wird eine Referenz auf eine Objekt der Java-Klasse Random erzeugt. Suche in der Dokumentation zur Java-Klassenbibliothek die Klasse Random und informiere Dich, wie eine zufällige ganze Zahl zwischen 1 und der Fensterbreite erzeugt wird.

15. Erstelle die Klasse Zufallsgenerator. Importiere in dieser Klasse die Klasse Random und erstelle ein Objekt der Klasse Random. Erstelle eine Methode public int getZufallStufeX(), die einen Wert zwischen 1 und Fensterbreite liefert. Reduziere den Zufallswert noch um die Breite der Stufe, so dass die Stufe immer vollständig zu sehen ist.
16. Erzeuge in der Klasse Fenster eine Referenz auf ein Objekt der Klasse Zufallsgenerator und erzeuge ein Obekt im Konstruktor.
17. Erstelle eine Klasse Stufe. Ähnlich wie bei unserem Doodle benötigen wir die Koordinaten der Stufe, die wir in den Attributen x und y speichern. Erstelle alle Set- und Get-Methoden und zusätzlich eine Methode bewegen(), mit der wir die Stufe ein kleines Stück nach unten bewegen, der Wert der Y-Koordinate also zunimmt. Der Konstruktor soll einen Parameter des Datentyps int haben, mit der die X-Position der Stufe festgelegt wird.
18. Programmiere eine Referenz in der Klasse Fenster auf ein Objekt der Klasse Stufe und erzeuge ein Objekt der Klasse Stufe im Konstruktor.
19. Ähnlich wie bei unserem Doodle benötigen wir eine Methode in der Klasse Fenster, mit der wir eine Referenz auf die Stufe erhalten. Verwende die Signatur **`public Stufe getStufe();`**
20. Zeichne in der Methode paintComponent() der Klasse Picturepanel nun auch die Stufe an ihrer aktuellen Position.
21. Rufe in der Programmschleife die Methode bewegen() der Stufe auf und stelle sicher, dass wenn die Stufe das untere Ende erreicht hat, eine neue Stufe erzeugt wird.
22. Teste das Programm ausgiebig und verändere verschiedene Attributwerte und Parameter. Die Lösung kannst du mit dem Programm Schiller_Jump03 vergleichen.

In unserem Spiel soll es nicht nur eine Stufe geben, sondern mehrere. Daher benötigen wir einen Behälter, in denen mehrere Stufen hineinpassen. Wir wollen die Java-Klasse ArrayList verwenden.

Dazu müssten wir unser Attribut **`Stufe stufe`** ändern zu **`ArrayList<Stufe> stufen`**. Als Importanweisung benötigen wir import java.util.ArrayList. Zudem müssen nun überall, wo wir eine Methode der Klasse Stufe aufgerufen haben, diese Methode für alle Stufen aufrufen. Dies ist mit einer For-Schleife möglich.

23. Ändere in der Klasse Fenster den Datentyp **`Stufe stufe`** in **`ArrayList<Stufe> stufen`**. Gebe die notwenige Import-Anweisung an.
24. Erzeuge im Konstruktor zehn Stufen, die gleichmäßig über das Spielfeld verteilt sind. Die X-Koordinate soll dabei wieder zufällig sein. Dazu müssen wir im Konstruktor der Klasse Stufe einen zweiten Konstruktor implementieren, der als Parameter zwei Integer-Werte für x und y verwendet.
25. Ändere den Aufruf **`stufe.bewegen()`** in eine Sequenz um, mit der du für alle Stufen die Methode bewegen() aufrufst.
26. Hat eine Stufe das untere Ende des Fensters erreicht, muss sie aus der ArrayList gelöscht werden, dies erfolgt mit der Methode remove(int). Erzeuge sodann eine neue Stufe und füge sie der ArrayList mit der Methode add(Stufe) hinzu.
27. Ändere in der Klasse Fenster die Methode getStufe() in getStufen() um.
28. Ändere in der Klasse Picturepanel die Anweisung in der Methode paintComponent so um, dass alle Stufen gezeichnet werden.

Jetzt soll unser Doodle endlich auch auf den Stufen die Möglichkeit erhalten zu springen. In der Endlosschleife müssen wir also in jedem Schleifendurchlauf prüfen, ob der Doodle auf einer Stufe steht, um ihm dann die nötige Energie für seinen Sprung zuweisen zu können.

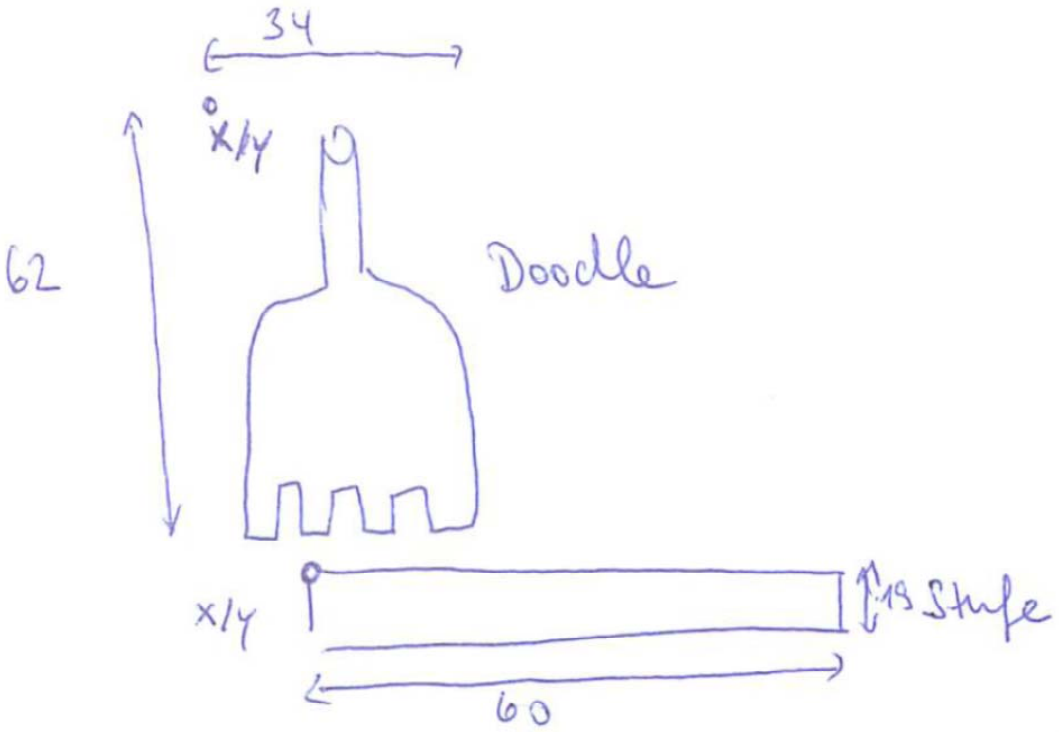
```
if (istAufStufe()) doodle.setEnergie(100);
```

Beachten sollten wir noch, dass die Stufe nur dann zu einem Sprung führen soll, wenn unser Doodle sich von oben nach unten beweget.

```
if (istAufStufe() && (doodle.getEnergie()==0)) doodle.setEnergie(100);
```

Die Methode public boolean istAufStufe() muss allerdings auch implementiert werden. Die Methode soll **`true`** liefern, wenn der Abstand des Doodle, also die Entfernung der x-Koordinate und der y-Koordinate einer Stufe sich in einem bestimmten Bereich befindet.

Auf der Abbildung der nächsten Seite sind die Maße meines Beispiel-Doodles und der Beispiel-Stufe angegeben. Es soll der Doodle mindestens zu 20% auf der Stufe sein (x-Richtung) und weniger als zehn Pixel in Y-Richtung entfernt sein, um „als auf der Stufe sthend“ zu gelten.



29. Füge die Anweisung in die Programmschleife hinzu, dass nach jeder Bewegung geprüft wird, ob Doodle auf einer Stufe steht, um ihm eine „neue Energie“ zuzuweisen.
```
if (istAufStufe() && (doodle.getEnergie()==0))
doodle.setEnergie(100);
```
30. Ändere die Anweisung für das Erreichen des Bodens in der Art, dass das Attribut ende den Wert true erhält.
31. Um überhaupt auf einer Stufe landen zu können, müssen die Stufen stehenbleiben. Sie dürfen sich erst dann bewegen, wenn sich unser Doodle sich in der oberen Hälfte des Fensters befindet. Ergänze diese Einschränkung beim Aufruf der Methode bewegen() der Stufen.