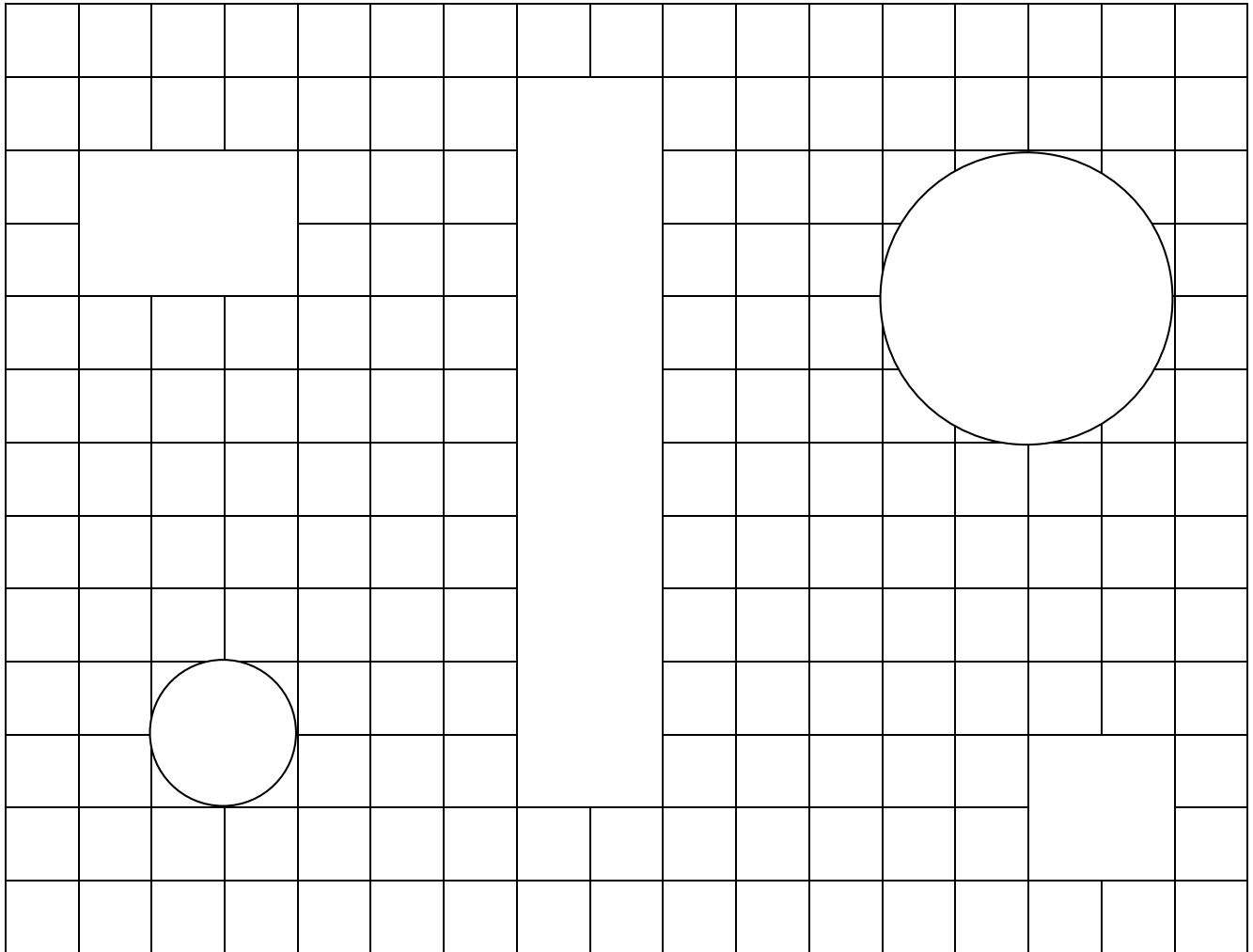


Abstrakte Klassen

Beispiel: Geometrische Figuren

Ein Programm stellt verschiedene geometrische Figuren (Rechtecke, Dreiecke, Kreise) dar.



1. Wieviele Objekte siehst Du? 5

2. Versuche die Objekte möglichst genau zu beschreiben.

3. Untersuche die Objekte auf gemeinsame Eigenschaften / Verhalten und fassen sie zu Klassen zusammen. Erstelle UML-Diagramme zu den Klassen und Objekten.

4. Untersuche jetzt die Klassen auf Gemeinsamkeiten und erstelle eine passende Klassenhierarchie.

Abstrakte Methoden

Alle von der Oberklasse `GeometrischeFigur` abgeleitete Klassen erben von dieser die Methode `berechneFlaecheninhalt()`: FKZ. Da die Formel zur Berechnung des Flächeninhalts jedoch von der Art der geometrischen Figur abhängt, ist es notwendig, dass jede Unterklasse diese Methode **überschreibt**.

Um sicherzustellen, dass jede Unterklasse dies auch tut, wird die Methode in der Klasse `GeometrischeFigur` als *abstrakt* gekennzeichnet. Dies **zwingt** jede Unterklasse dazu, die Methode zu **überschreiben**.

```
from abc import ABC, abstractmethod

class GeometrischeFigur(ABC): ← Abstrakte Klasse
    @abstractmethod ← Abstrakte Funktion
    def berechneFlaecheninhalt(self):
        pass

class Rechteck(GeometrischeFigur):
    def berechneFlaecheninhalt(self):
        return self.hoehe * self.breite
```

Eine **abstrakte Methode** besitzt keinen Methodenrumpf und auch keine Anweisungen. Sie zwingt jede Unterklasse dazu die Methoden zu überschreiben. Damit ist sichergestellt, dass alle Unterklassen eine Methode mit dieser Signatur implementieren.

Eine Klasse, die mindestens eine **abstrakte Methode** enthält, muss selbst als **abstrakte Klasse** gekennzeichnet werden.

Im **UML-Klassendiagramm** wird eine abstrakte Methode entweder *kursiv* dargestellt oder um die Eigenschaft **{abstract}** ergänzt.





Übungsaufgabe

Sie sind ein Softwareentwickler und sollen ein Programm zur Verwaltung von verschiedenen elektronischen Geräten erstellen. Verwenden Sie dazu abstrakte Klassen, um eine gemeinsame Schnittstelle für alle Gerätetypen zu definieren. Definieren Sie eine abstrakte Klasse `Geraet`, die die Attribute `marke (String)`, `modell (String)` und `preis (Float)` enthält. Diese Klasse soll auch eine abstrakte Methode `beschreibung` haben, die eine Beschreibung des Geräts zurückgibt.

Erstellen Sie zwei abgeleitete Klassen `Laptop` und `Smartphone`, die von der abstrakten Klasse `Geraet` erben. Die Methode `beschreibung` soll in diesen beiden Klassen implementiert werden. Die Klasse `Laptop` soll die spezifischen Attribute `prozessor (String)`, `ram (Integer)` und `speicher (Integer)` haben. Die Klasse `Smartphone` soll die spezifischen Attribute `kamera (Integer)` und `akku (Integer)` haben. Zusätzlich sollen Sie eine weitere abgeleitete Klasse `Tablet` erstellen und die Methode `beschreibung` entsprechend implementieren. Die Klasse `Tablet` soll die Attribute `displayGroesse (Float)` und `akku (Integer)` haben.



Arbeitsauftrag:

Zeichnen Sie ein Klassendiagramm zu der oben beschriebenen Situation. Vervollständigen Sie die Klassen mit den `get`- und `set`-Methoden.

