**Universität Potsdam**
**Institut für Informatik**

# Praxis der Programmierung

## 9. Aufgabenblatt

1. Schreiben Sie eine Klasse `Point`, die zwei gekapselte Datenelemente für die $x$- und die $y$-Koordinate, einen Standard- und einen Initialisierungskonstruktor, zwei Getter (für beide Datenelemente), eine Methode zum Verschieben auf neue Koordinaten (absolutes Verschieben) und eine Methode zum Verschieben um einen Vektor (relatives Verschieben) besitzt.
   *Hinweis: Es lohnt sich ein Blick auf Ihre bisher geschriebenen Klassen.*

2. Erstellen Sie nun eine abstrakte Klasse `Figure`, die ebene geometrische Figuren repräsentiert.

   - Sie besitzt ein gekapseltes Datenelement vom Typ `Point`, das die Position der Figur in der Ebene bestimmt.
   - Ein Standardkonstruktor und ein Konstruktor mit zwei Parametern vom Typ `int` erzeugen jeweils entsprechende Exemplare von `Point` und initialisieren damit das Datenelement der Klasse. Die beiden Konstruktoren sind nur für die Unterklassen und Klassen aus demselben Paket sichtbar.
   - Es gibt für alle Klassen sichtbare Methoden zum absoluten und relativen Verschieben der Figur.
     *Nutzen Sie dazu die entsprechenden Methoden der Klasse* `Point`.
   - Es werden die Schnittstellen von vier weiteren Methoden vereinbart: zum Abfragen und zum Ändern der Größe der Figur (die jeweils durch einen `int`-Wert bestimmt sein wird), zur Berechnung des Flächeninhalts sowie des Umfangs der Figur.

3. Es gibt Klassen für zwei Arten von ebenen geometrischen Figuren und somit zwei (implementierte) Unterklassen von `Figure`:

   - `Square` hat ein zusätzliches gekapseltes Datenelement vom Typ `int`, das die Kantenlänge des Quadrats bezeichnet. Es gibt einen Standardkonstruktor und zwei Initialisierungskonstruktoren.

- `Circle` als Klasse von Kreisen mit einem zusätzlichen gekapselten Datenelement vom Typ `int` für den Radius des Kreises. Es gibt einen Standardkonstruktor und zwei Initialisierungskonstruktoren.

4. Schreiben Sie eine Applikation, mit der Sie die Klassen testen.

5. Entwickeln Sie eine alternative Klassendefinition von `Circle` mit dem Klassennamen `Circ`, wobei `Circ` jetzt von `Square` ableitet (Unterklasse von `Square` ist). Testen Sie!
   Halten Sie diese Klassenstruktur für sinnvoll? Warum oder warum nicht?

   ---

   ---

6. In einem Geschirrschrank einer Großküche können tiefe und flache Teller, Gläser, Kaffeebecher, Teetassen, Suppentassen, Unterteller für Tassen und Müslischalen gelagert werden. Alle Geschirrteile haben einen Durchmesser (in cm), eine Höhe (in cm) sowie ein Gewicht (in Gramm). Sie unterscheiden sich durch ihre Funktion: Getränkeaufnahme (`'g'`), Nahrungsaufnahme (`'n'`), Sonstiges (`'s'`). Alle Geschirre lassen sich im Geschirrschrank stapeln.

   (a) Entwickeln Sie eine Vererbungshierarchie für die verschiedenen Geschirrteile. Implementieren Sie die Klassen `Geschirr`, `Glas` und `Mueslischale` in Java. Die oberste Oberklasse soll `Geschirr` sein.
   Begründen Sie, ob und ggf. warum die Oberklasse `Geschirr` abstrakt sein sollte.

   ---

   ---

   (b) Implementieren Sie für alle drei Klassen sinnvolle Konstruktoren.

   (c) Begründen Sie, in welcher der Klassen die `get`- und `set`-Methoden des Attributs `gewicht` implementiert werden müssen.

   ---

   ---

   Implementieren Sie die beiden Methoden.

(d) Ergänzen Sie eine Klasse `Geschirrschrank`. Sie besitzt ein gekapseltes Datenelement vom Typ `int`, das die Kapazität des Geschirrschranks (in Gramm) bezeichnet. Ergänzen Sie sinnvolle Konstruktoren. Implementieren Sie eine Methode
`public void einraeumen(Geschirr pGeschirr)`, welche die Kapazität um das Gewicht von `pGeschirr` reduziert.

Erläutern Sie stichwortartig, warum mit der Methode `einraeumen` auch z.B. Teller eingeräumt werden können.

---

---

*basierend auf: "Informatik 2", T.Kempe, A.Löhr (Hrsg.), Schöningh, 2015*