Fachbereich Ingenieurwissenschaften
Studienbereich U+D, Informatik
Andreas Zinnen
Peter Dannenmann
Siarhei Barodzich
Sommersemester 2019

Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Informatik 2 / OOP

Übungsblatt 7

05.06 und 06.06.2019

P 7.1 Smells like coffee spirit...

a) In der `.h`-Datei finden Sie die Deklaration einer Klasse `Produkt`. Implementieren Sie die fehlenden Funktionen `getBruttoPreis` und `erhoeheNettopreis`.

b) Schreiben Sie den Ausgabe-Operator `<<`, um *Name*, *Produktnummer*, *Bruttopreis* von Produkten mit `cout` auszugeben. Die Attribute sollen `protected` bleiben.

c) Legen Sie folgende Instanz der Klasse `Produkt` statisch an:

- `87097RT34, Kaffee, 7.5` (EUR)

Geben Sie das Produkt aus, falls der Operator `<<` implementiert wurde.

d) Erweitern Sie die Klasse `Produkt` um eine Methode, die die *Kategorie* des Produktes als `string` über `return` zurückgibt:

- GEHOBEN: Falls der Nettopreis kleiner als 7 (EUR) ist
- HOCH: sonst

Testen Sie die Methode auf dem Kaffee mit Produktnummer `87097RT34` und geben Sie die entsprechende Kategorie aus.

e) Schreiben Sie eine Methode `prepare` zur Zubereitung von Produkten. Die Methode führt folgende Schritte aus:

- Ausgabe von *Name*, *Produktnummer*, *Preis* in einer Zeile (durch Leerzeichen getrennt), z.B. *Kaffee 87097RT34 7.5*
- Ausgabe: *starte Zubereitung von Produkt...*
- warte 2 Sekunden (`sleep`)
- Ausgabe: *"fertig..."*

Testen Sie die Funktion auf dem Produkt `87097RT34`.

f) Erweitern Sie die Deklaration der Klasse `Kaffee`, damit sie von der Klasse `Produkt` erbt. Zusätzlich zu den geerbten Eigenschaften soll die Klasse `Kaffee` noch Attribute für den Röstgrad als *string* sowie die Region als `string` haben, aus der der Kaffee kommt.

g) Ein Konstruktor mit Parametern für alle Attribute soll Instanzen der Klasse `Kaffee` erstellen. Nutzen Sie hier eine Konstruktorkette in der Implementierung.

h) Überschreiben Sie die Methode `prepare` zur Zubereitung von Kaffee. Statt der Ausgabe: *starte Zubereitung von Produkt...* soll die Ausgabe: *starte Zubereitung von Kaffee...* ausgegeben werden.
Achten Sie bitte darauf, dass bei polymorphen Objekten (Vielfältigkeit) die spezielle Methode aufgerufen wird.

i) Legen Sie folgende Instanzen der Klasse `Kaffee` dynamisch an und fügen Sie diese in die existierende Liste mit symbolischen Namen `kaffeeListe` (in der `main`-Funktion) hinzu:

   - 76498CF45, Kaffee, 4.5, BLOND, Kenya
   - 87097RT34, Kaffee, 5.5, DARK, Colombia

j) Laufen Sie über die Liste der Kaffees (`kaffeeListe`) und geben Sie alle Kaffee-Sorten der Kategorie GEHOBEN aus.

k) Implementieren Sie den Operator `+=` auf der Klasse `ProduktListe`. Der Operator fügt einen Pointer auf ein Produkt zur ProduktListe hinzu.

l) Implementieren Sie den Destruktor der Klasse `ProduktListe`. Der dynamische Speicher für alle in der Liste gespeicherten Produkte muss hier wieder freigegeben werden.
Hinweis: Schleife + Freigabe über den Pointer auf den Datentyp.

m) Deklarieren und implementieren Sie eine Methode `prepareProducts` auf `ProduktListe`. Die Methode ruft für alle Produkte in der Liste die Methode `prepare` auf.
Erweiterung: Nutzen Sie *Threads* und die Funktion `detach`, damit alle Produkte parallel zubereitet werden können (`prepare` wird in eigenem Thread aufgerufen).

n) Legen Sie dynamisch eine Instanz der Klasse `Kakao` an:

   - 76498CF55, Kakao, 4.5

   Erzeugen Sie eine Instanz von `ProduktListe` und fügen Sie alle dynamischen(!) Instanzen der Klassen `Kakao` und `Kaffee` hinzu (insgesamt 3 Instanzen). Rufen Sie im Anschluss die Methode `prepareProducts` auf und überprüfen Sie, ob alle Produkte zubereitet werden.