5.2.A Einführung und Klassen/Zugriffsschutz für Komponenten – Arbeitsblatt

Aufgabe 1: Zugriffsspezifizierer und neue Methoden für `punkttyp`

In der Klasse `punkttyp` sollen die Attribute einen Zugriffsschutz bekommen, so dass sie von außen (von der main()-Funktion und von anderen Klassen aus) nicht mehr direkt zugreifbar sind. Damit man mit der Klasse gut arbeiten kann, müssen außerdem weitere Methoden erstellt werden.

a) (`punkttyp2.cpp`)
Ergänzen Sie die Zugriffsspezifizierer!

b) Ergänzen Sie die Funktionen

```
1 double get_x(void);
2 double get_y(void);
```

Sie sollen die x- bzw. y-Komponente als Rückgabewert ausgeben.

c) Ergänzen Sie die Funktionen

```
1 void set_x(double neu_x);
2 void set_y(double neu_y);
```

Sie sollen die x- bzw. y-Komponente des Objekts auf den Wert setzen, der im Parameter steht.

d) Ergänzen Sie die Funktion

```
1 void init(void);
```

Sie soll beide Attribute auf den Wert `0.0` setzen.

e) Fügen Sie alles so zusammen, dass die Funktionalität aus `punkttyp1.cpp` weiter erhalten bleibt!

f) Zeichnen Sie das UML-Diagramm für die veränderte Klasse (mit Zugriffsspezifizierern)!

Aufgabe 2: Zugriffsspezifizierer und neue Methoden für `tageszeittyp`

Auch in der Klasse `tageszeittyp` sollen die Attribute einen Zugriffsschutz bekommen, so dass sie von außen nicht mehr direkt zugreifbar sind.
Auch hier sollen neue Methoden erstellt werden.

a) (`tageszeittyp2.cpp`)
Ergänzen Sie die Zugriffsspezifizierer!

b) Ergänzen Sie die Funktionen

```
1 unsigned char get_std(void);
2 unsigned char get_min(void);
3 unsigned char get_sek(void);
```

Sie sollen den Wert des entsprechenden Attributs als Rückgabewert ausgeben.

c) Ergänzen Sie die Funktionen

```
1 void set_x(double neu_x);
2 void set_y(double neu_y);
```

d) Ergänzen Sie die Funktionen

```
1 void init(void);
2 void init(unsigned char stunde);
```

`init()` soll die Uhrzeit auf 12:00:00 Uhr setzen.
`init(19)` soll die Uhrzeit auf 19:00:00 Uhr setzen.

e) Ergänzen Sie die Funktion

```
1 bool is_ok(void);
```

Sie soll prüfen, ob die Attribute im gültigen Bereich liegen. Falls ja, soll `true` ausgegeben werden, andernfalls `false`[1].

f) Fügen Sie alles so zusammen, dass die Funktionalität aus `tageszeittyp1.cpp` weiter erhalten bleibt!

g) Zeichnen Sie das UML-Diagramm für die veränderte Klasse (mit Zugriffsspezifizierern)!

Aufgabe 3: Neuer Datentyp `udouble`

In C und C++ gibt es leider keinen Datentyp `unsigned double`, der physikalische Größen verkörpern würde, die immer positiv sind (z. B. die Temperatur in Kelvin kann nicht negativ sein). Aber man kann in C++ eine solche Klasse aufbauen. Einziges Attribut ist eine `double`-Komponente `wert`.

a) Schreiben Sie die Funktionen

```
1 void set_wert(double neu_wert);
2 double get_wert(void);
3 bool is_ok(void);
4 void init(void);
```

Achten Sie dabei darauf, dass sichergestellt ist, dass `wert` niemals negativ sein darf.

b) Zeichnen Sie das UML-Diagramm der Klasse!

c) (`udouble2.cpp`)
Fügen Sie die Klasse und ein Hauptprogramm zusammen, in welchem die Klasse geprüft wird!

[1] `bool` ist mit `int` vergleichbar. `false` bedeutet 0, `true` bedeutet 1.

Aufgabe 4: Erweiterung von `bruch`

Der Zugriffsschutz soll auch für die Klasse `bruch` eingeführt werden (Ergebnis: `bruch2.cpp`).

a) Ergänzen Sie die Zugriffsspezifizierer! Alle Attribute sowie die Methoden `ggT()` und `kuerzen()` sollen als `private:` bestimmt werden, der Rest als `public:`.

b) Erstellen Sie eine Methode `init()`, mit der man den Bruch auf einen Startwert setzen kann!

c) Erstellen Sie das veränderte UML-Diagramm!

Aufgabe 5: Erweiterung von `datum`

Auch für die Elemente von Klasse `datum` soll es Zugriffsschutz geben (Ergebnis: `datum2.cpp`).

a) Ergänzen Sie die Zugriffsspezifizierer! Alle Attribute sowie die Methoden `is_schaltjahr()` und `anzahl_tage()` sollen als `private:` bestimmt werden, der Rest als `public:`.

b) Erstellen Sie eine Methode `init()`, mit der man das Datum auf einen gültigen Startwert setzen kann!

c) Erstellen Sie auch hier das veränderte UML-Diagramm!

Aufgabe 6: Erweiterung von `person`

Auch in `person` soll es Zugriffsschutz geben (Ergebnis: `person2.cpp`).

a) Die Attribute sollen `private:` sein, der Rest `public:`.

b) Die zusätzliche Methode `init()` soll Vorname und Name auf `""` und das Geburtsdatum auf den 1.1.2020 setzen.

c) Erstellen Sie eine Methode `bool ist_ok(void)`, mit der überprüft werden kann, ob alle Attribute einen sinnvollen Inhalt haben!