Technische Universität Berlin **Neuronale Informationsverarbeitung**

C/C++ Übungsblatt 9 (Block 2)

Prof. Dr. Klaus Obermayer und Mitarbeiter

Polymorphie, Abstrakte Klassen, IO, Vektoren

Erinnerung: Bitte denken Sie an den abschließenden Block-Test.

Verfügbar ab:	**18.01.2021**
Abgabe bis:	**25.01.2021**

Aufgabe 1: Datum

5 Punkte

Die Darstellung des Datums ist weltweit nicht einheitlich. Während es zwar einen internationalen Standard (ISO 8601) gibt, der das Format JJJJ.MM.TT vorschlägt, wird (nicht nur) in Deutschland das Format TT.MM.JJJJ sowie (hauptsächlich) in den Vereinigten Staaten das Format MM.TT.JJJJ verwendet.

Schreiben Sie eine abstrakte Klasse `Datum`, welche die privaten `int`-Attribute `tag`, `monat` und `jahr` enthält. Die Klasse soll einen parametrisierten Konstruktor definieren, der die drei Attribute initialisiert und eine Warnung auf die Konsole ausgibt, wenn Tag oder Monat nicht positiv sind oder der Tag größer als 31 bzw. der Monat größer als 12 ist[1]. Implementieren Sie in der Klasse weiterhin die drei öffentliche Methoden wie folgt:

- `getTT()` – gibt einen String bestehend aus genau zwei Zeichen entsprechend dem Attribut `tag` (z.B. “04”, falls `tag` die Zahl 4 enthält) zurück
- `getMM()` – gibt einen String bestehend aus genau zwei Zeichen entsprechend dem Attribut `monat` (z.B. “09”, falls `monat` die Zahl 9 enthält) zurück
- `getJahr()` – gibt das Attribut `jahr` als String zurück (falls `jahr` < 1000, muss die Jahreszahl nicht vierstellig sein)

Außerdem soll die Klasse `Datum` eine parameterlose, abstrakte (d.h. rein virtuelle) Methode `formatiere` mit dem Rückgabetyp `string` deklarieren.

Schreiben Sie drei von `Datum` abgeleitete Klassen `DatumISO`, `DatumDE` und `DatumUSA`, in denen Sie die abstrakte Methode `formatiere` gemäß den oben vorgestellten Datums-Formaten implementieren. Verwenden Sie den Konstruktor der Superklasse `Datum`, um die Attribute `tag`, `monat` und `jahr` zu initialisieren.

Testen Sie Ihre Implementierungen in einem Testprogramm `test_datum`, in dessen `main`-Funktion Sie für die beiden in der Tabelle angegebenen Daten Objekte der Typen `DatumDE`, `DatumUSA` und `DatumISO` (d.h. insgesamt 6) erzeugen und die Rückgabe der jeweiligen `formatiere`-Methode auf die Konsole ausgeben.

Achten sie darauf, dass nach dem Ausführen ihres Programmes der verwendete Speicher wieder vollständig freigegeben wird.

Tag	Monat	Jahr
1	12	2021
4	20	2000

[1] Es muss nicht berücksichtigt werden ob es sich um einen Monat mit weniger als 31 Tagen handelt.

Die Ausgabe Ihres Testprogramms könnte z.B. so aussehen:

```
Erzeuge Objekte und gebe korrektes Datum in den drei Formaten aus:
Deutsche Formatierung: 01.12.2021
Amerikanische Formatierung: 12.01.2021
ISO-Formatierung: 2021.12.01
Erzeuge Objekte und gebe falsches Datum in den drei Formaten aus:
Warnung: Ungueltiger Monat 20.
Deutsche Formatierung: 04.20.2000
Warnung: Ungueltiger Monat 20.
Amerikanische Formatierung: 20.04.2000
Warnung: Ungueltiger Monat 20.
ISO-Formatierung: 2000.20.04
```

Aufgabe 2: Vector — 5 Punkte

In dieser Aufgabe sollen ein Vektor benutzt werden, um beliebig viele einzugebene Studenten zu speichern. Zusätzlich sollen diese ausgegeben werden können.

Die Klasse `Student` steht auf ISIS zum Download zur Verfügung.

Erstellen Sie eine Datei `Studentenverwaltung.cpp` mit einer `main`-Funktion. Erfüllen Sie in dieser folgende Aufgaben:

1. Erzeugen Sie einen Vektor `std::vector<Student>`.
2. Erstellen Sie eine Endlos-**`while`**-Schleife. Fragen Sie den Nutzer per **`char`**-Eingabe, ob er
   - e das Programm beenden möchte
   - a einen Student hinzufügen möchte
   - s einen Studenten über die Matrikelnummer suchen möchte
   - d einen Student löschen möchte
   - l alle Studenten auflisten möchte
   - Gibt der Nutzer etwas anderes ein, weisen Sie ihn erneut auf seine Auswahlmöglichkeit hin.
3. Nehmen Sie nun basierend auf der Nutzereingabe eine Fallunterscheidung vor.
   - Fall e: Beenden Sie die Endlosschleife oder direkt das Programm.
   - Fall a: Führen Sie den Benutzer durch die Eingabe aller nötigen Daten für einen Student, fügen Sie dann einen neuen Student mit entsprechen Daten in den Vektor ein. Informieren Sie den Nutzer über den Erfolg.
   - Fall s: Durchsuchen Sie den Vektor nach einem Studenten mit dieser Matrikelnummer und geben Sie ggf. die Rückgabe der `std::string asString`-Methode auf der Konsole aus. Existiert der Student nicht, geben Sie eine entsprechende Meldung aus.
   - Fall d: Durchsuchen Sie den Vektor nach einem Studenten mit dieser Matrikelnummer und entfernen Sie ihn, falls er existiert. Geben Sie eine entsprechende Meldung aus und nutzen sie dabei ggf. die `std::string asString`-Methode.
   - Fall l: Geben Sie alle Studenten unter Nutzung der `std::string asString`-Methoden aus.
4. Nachdem der Befehl abgearbeitet wurde, beginnt die Endlosschleife wieder von vorne und bittet den Nutzer um den nächsten Befehl.

Hinweis 1: Denken Sie daran, dass das Einlesen von Werten mittels `cin >> ...` den Zeilenumbruch nicht aus dem Stream entfernt. Für folgende `cin`-Aufrufe ist dies zunächst irrelevant, aber ein `getline()`-Aufruf liest nur bis zum ersten gefundenen Zeilenumbruch (und entfernt diesen). In den Tutorien und Hausaufgabenvorgaben wurden Wege vorgestellt, die verbleibenden Zeilenumbrüche (und andere Zeichen) aus dem Stream zu entfernen.

Hinweis 2: Sie können sich für die Arbeit mit Vektoren an folgendem Code-Snippet orientieren:

```
1  std::vector<int> myvector;
2  for (unsigned int i=0; i<=5; ++i)
3      myvector.push_back(i);            // pusht die Zahlen 0-5 auf den Vektor
4  std::cout << myvector.at(1) << endl;  // printet 1
5  std::cout << myvector.size() << endl; // printet 6
6  myvector.erase (myvector.begin()+2);  // löscht den momentan 3. Eintrag
7  std::cout << myvector.size() << endl; // printet 5
8  myvector.erase (myvector.begin()+2);  // löscht den momentan 3. Eintrag
9  std::cout << myvector.size() << endl; // printet 4
10 for (unsigned int i=0; i<myvector.size(); ++i)
11     std::cout << ' ' << myvector[i];  // printet  0 1 4 5
```

Hinweis 3: Versetzen Sie sich beim Testen in die Lage des Nutzers. Ist zu jedem Zeitpunkt klar, was der Nutzer für Optionen hat bzw. was er tun muss, um das Programm wie vorgesehen nutzen zu können? Sehen die Konsoleneingaben und -ausgaben gut lesbar formatiert aus?