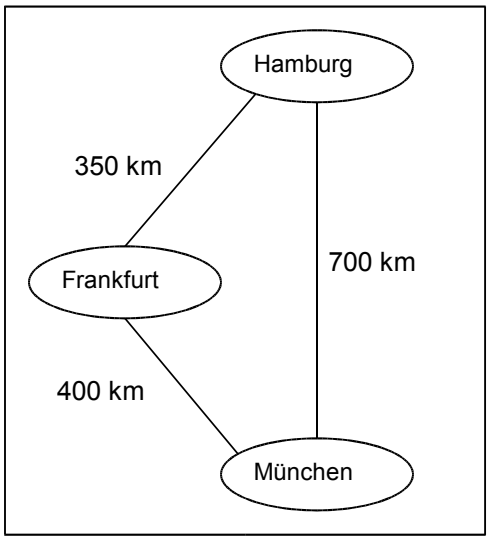
Prüfung Software-Entwicklung 2

Aufgabensteller: Dr. B.Glavina, Dr. S.Hahndel, Dr. R. Hendrych, Dr. H.-M. Windisch
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Aufgabe 1 (Eisenbahn-Streckennetz in C; etwa 27%)

Nebenstehende Abbildung zeigt einen kleinen Ausschnitt aus einem Eisenbahn-Streckennetz.
Es besteht aus Knoten und Kanten.
Ein Knoten stellt einen Ort dar, an dem eine Zugfahrt beginnt oder endet.
Eine Kante zwischen zwei Knoten zeigt an, dass eine direkte Zugverbindung zwischen den beiden Orten besteht und gibt die zurückzulegende Entfernung an.



a) Definieren Sie zwei Verbunde für die Knoten und Kanten des Streckennetzes. Jeder Knoten kann dabei bis zu 5 Kanten referenzieren. Jede Kante enthält neben der Entfernungsinformation Verweise auf die verbundenen Knoten.

b) Nachfolgendes Programm-Fragment erzeugt das in der Abbildung gezeigte Streckennetz mit den unter a) definierten Verbunden.

```
void main()
{
        struct Knoten *hamburg   = erzeugeKnoten("Hamburg");
        struct Knoten *frankfurt = erzeugeKnoten("Frankfurt");
        struct Knoten *muenchen  = erzeugeKnoten("München");

        verbindeKnoten(hamburg,  frankfurt, 350.0);
        verbindeKnoten(muenchen, hamburg,   700.0);
        verbindeKnoten(muenchen, frankfurt, 400.0);
}
```

Geben Sie den vollständigen C-Quellcode für die beiden Funktionen `erzeugeKnoten` und `verbindeKnoten` an.

c) Geben Sie den vollständigen C-Quellcode für eine Funktion `fuegeKnotenEin` an, die einen neuen Knoten zwischen zwei bereits verbundene Knoten einfügt. Stützen Sie sich dabei auf die bereits definierten Funktionen ab.

Die Funktion `fuegeKnotenEin` habe folgende Parameter:

- k1, k2: Zeiger auf verbundene Knoten
- name: Ortsname des neuen Knotens
- entfernung: Entfernung zwischen Knoten k1 und dem neuen Knoten. Die Entfernung zwischen dem neuen Knoten und Knoten k2 ergibt sich aus der Differenz aus der „alten“ Entfernung und der angegebenen.

Beispiel (mit Bezug auf obiges Hauptprogramm): Einfügen des Orts „Nürnberg“ zwischen München und Hamburg.

```
fuegeKnotenEin(muenchen, hamburg, "Nürnberg", 180.0);
```

Aufgabe 2 (Klassen und Vererbung; etwa 12%)

a) Das Sortiment eines Supermarkts soll in einem Programm modelliert werden. Jede Warengruppe hat gewisse Eigenschaften, die im Folgenden beschrieben sind.

- Gartenpflanzen: Bezeichnung, Blütezeit, Anlieferkosten, Preis
- Elektrospielzeug: Bezeichnung, Batterietyp, Preis, Zielgruppenalter
- Backwaren: Preis, Herstelldatum, Bezeichnung, Lagertemperatur,
- Puppen: Zielgruppenalter, Bezeichnung, Kunststoffart, Preis
- Milchwaren: Volumen, Lagertemperatur, Preis, Verfallsdatum, Bezeichnung
- Gartenmöbel: Anlieferkosten, Preis, Bezeichnung, Wetterfestigkeit
- Fleischwaren: Lagertemperatur, Bezeichnung, Gewicht, Fettanteil, Preis

Zeichnen Sie ein Klassendiagramm für alle dieses Klassen, wobei Sie geschickt die Vorteile der Vererbung nutzen, indem Sie zusätzliche (Ober-)Klassen einführen: Produkt, Nahrung, Spielzeug, Gartenprodukt.

b) Wo (bei welcher Klasse bzw. welchen Klassen) würden Sie die Methode "druckePreisschild" hinzufügen, wenn sie für alle Klassen gelten soll?
Was müssten Sie weiter tun, damit auf dem Preisschild der Milchprodukte auch die Verfallsdaten erscheinen?

Aufgabe 3 (Klassen in C++; etwa 27%)

a) Schreiben Sie in C++ die Spezifikation einer Klasse `Konto`, die folgende Bestandteile hat.

- `nummer` zur Speicherung einer ganzzahligen Kontonummer
- `inhaber` zur Speicherung des Namens des Inhabers
- `stand` zur Speicherung des auf dem Konto befindlichen ganzzahligen Guthabens in der Einheit Euro-Cent (Schulden werden als negatives Guthaben gespeichert)
- ein Konstruktor zur Initialisierung eines Kontos mit Inhabernamen und Kontonummer als Parameter; der Kontostand wird anfangs immer auf Null gesetzt
- `druckeBasisinfo()` zur Ausgabe der aktuellen Werte aller Attribute eines `Konto`-Objekts auf dem Bildschirm (inklusive Angabe der Bedeutung der Werte)
- `buche(...)` zur Einzahlung bzw. Auszahlung von als Parameter übergebenen Beträgen auf das Konto (positive Beträge erhöhen `stand`, negative wirken als Auszahlung)

Begründen Sie Ihre Wahl der Zugriffsrechte kurz; wären auch Alternativen sinnvoll?

b) Geben Sie die Implementierung in C++ für die Klasse `Konto` an.

c) Schreiben Sie in C++ eine Spezifikation für die Klasse `GiroKonto`, welche wie `Konto` aufgebaut ist, aber zusätzlich noch die Komponenten `sollzins` (als Kommazahl) und `druckeInfo()` hat. `sollzins` wird über den dritten Parameter des Konstruktors von `GiroKonto` besetzt.
Warum sind Sie hier gut beraten, wenn Sie das Konzept der Vererbung benutzen?

d) Schreiben Sie in C++ die Spezifikation für eine weitere Klasse `FestgeldKonto`, welche wie `Konto` aufgebaut ist, aber zusätzlich noch die Komponenten `habenzins` (als Kommazahl), `druckeInfo()` und `ablaufdatum` (vom Typ `date`) hat. Der Konstruktor bekommt als dritten und vierten Parameter die initiale Besetzung von `habenzins` und `ablaufdatum` mit.

e) Schreiben Sie in C++ die Implementierungen von `GiroKonto` und `FestgeldKonto`. Die Methoden `druckeInfo()` sollen die Basis-Informationen über das Konto und die jeweiligen Zinssätze drucken.
Was müssten Sie hier (zusätzlich) tun, wenn C++ keine Vererbung ermöglichte?

f) Definieren Sie im Hauptprogramm (`main`) zwei Objekte `gk1` und `gk2` von der Klasse `GiroKonto`. Die Konten gehören Arthur Dent, die Nummern dürfen Sie frei wählen und die Zinssätze sollen 10% betragen.
Buchen Sie nun von einen Konto 30 Euro ab, und auf das andere Konto 30 Euro dazu. Drucken Sie die Konteninformation für beide Konten aus.
Was hat das Umbuchen bzw. das Ausdrucken mit Polymorphie zu tun?

Aufgabe 4 (Login-Dialog in Java; etwa 27%)

Gegeben seien folgende Abbildung eines Login-Dialogs sowie das dazugehörige Programm-Fragment.


```
import java.awt.*;
import java.awt.event.*;

public class LoginDialog
    extends Frame
    implements ActionListener
{
    private TextField nameField;
    private TextField passwordField;
    private Button okButton;
    private Button cancelButton;

    static public void main(String args[])
    {
        LoginDialog d = new LoginDialog();
        d.setVisible(true);
    }

    public LoginDialog()
    {
        setTitle("Login");
        setLayout(new GridLayout(3, 2));

(b)

        pack();
    }

    public void actionPerformed(ActionEvent event)
    {
        String cmd = event.getActionCommand();
(c)

    }
}
```

a) Geben Sie ein Klassendiagramm in UML-Notation für das Programm-Fragment an!

b) Ergänzen Sie das Programm-Fragment an der Stelle (b) so, dass der Dialog korrekt aufgebaut wird und die Methode *actionPerformed* zur Behandlung von Ereignissen bzgl. der beiden Buttons verwendet wird.

c) Ergänzen Sie das Programm-Fragment an der Stelle (c) so, dass der Dialog folgendermaßen reagiert:

- Wird der OK-Button betätigt, soll eine Passwortprüfung ausgeführt werden. Ein Passwort ist dabei korrekt, wenn es folgenden Bedingungen genügt:
  - Das Passwort muss eine Länge von mindestens 4 Zeichen besitzen.
  - Es darf den Benutzernamen nicht als Teilstring enthalten.
  - Das Passwort darf kein Leerzeichen enthalten.
- Wird ein Passwort als falsch im Sinne der obigen Festlegung erkannt, so wird eine den ersten Fehler beschreibende Fehlermeldung auf die Standard-Ausgabe ausgegeben und der Dialog wartet auf weitere Aktionen des Benutzers.
- Ist das Passwort korrekt oder wurde der Abbrechen-Button betätigt, so wird das Programm beendet.

Hinweise:

- Verwenden Sie die Methode length() der Klasse String zur Bestimmung der Länge eines Strings sowie die Methode charAt(int i) zum Zugriff auf das i-te Zeichen eines Strings.
- Beenden Sie das Programm mit System.exit(0);
- Verwenden Sie zum Lesen der eingegebenen Werte für die Textfelder Benutzername und Passwort die Methode getText().

Aufgabe 5 (Allgemeine Fragen; etwa 7%)

a) Gegeben ist folgendes Code-Stück:

```
SomeClass *x;
...
x = new SomeClass();
x = NULL;
...
```

Was wird nach gewisser Zeit – wenn das Code-Stück hinreichend oft ausgeführt wurde – passieren?
Mit welchem zusätzlichen Befehl könnte man das vermeiden? Geben Sie sowohl den Befehl an, als auch die Stelle, wo er stehen müsste.
Ist obiger Code C oder C++ oder Java? Warum?

b) Warum gibt es in Java keinen Operator zum Freigeben von dynamischem Speicher?

c) C++ und Java unterstützen Exception-Handling. Erläutern Sie kurz die Vorteile sowie die Gefahr beim Einsatz dieses Konzepts.