Autor: Prof. Dr. Bernhard Humm, FB Informatik, Hochschule Darmstadt
Datum: 13. Juli 2010

Klausur Programmieren / Algorithmen und Datenstrukturen 1

1 Spielregeln zur Klausur

Allgemeines

- Die Bearbeitungszeit beträgt 3 Zeitstunden.
- Bitte halten Sie Ihren Studenten- und Personalausweis bereit
- Erfüllen Sie die Aufgabenstellung genau: machen Sie nicht weniger, aber auch nicht mehr!
- Arbeiten Sie zügig in der Reihenfolge der Aufgaben
- Implementieren Sie in kleinen Schritten und sorgen Sie dafür, dass Sie jederzeit eine lauffähige Anwendung haben
- Speichern Sie regelmäßig

Als Ergebnis der Klausur sind abzugeben

- … alle zum System gehörigen Dateien durch Kopieren auf den Server (genauere Informationen zu Beginn der Klausur).

Zugelassene Hilfsmittel

- Ausgedruckte Vorlesungsunterlagen und Bücher
- Persönliche Notizen auf Papier
- U. Breymann – C++ Einführung und professionelle Programmierung (PDF auf Klausurrechner)

- Online Hilfe in NetBeans

Verboten ist

- die Benutzung eigener Datenträger (Diskette, USB-Stick etc.); insbesondere dürfen keine Programme oder sonstige Daten auf Datenträger mitgebracht werden.
- die Benutzung eines anderen Rechners (Taschenrechner, PDA, Notebook, ...) als des zur Verfügung gestellten
- jegliche Kommunikation (mündlich, schriftlich, elektronisch, per Handy, übers Netz, wie auch immer...) mit anderen Personen, ausgenommen die Aufsichtführenden.
- Login unter einem anderen Account als dem für die Klausur angegebenen.
- Zugriff auf andere Verzeichnisse eines Servers, als die ausdrücklich zugelassenen.

Beurteilungskriterien (in Reihenfolge der Wichtigkeit)

1. Funktion des Systems: das Programm soll fehler- und warnungsfrei kompilieren und alle Tests erfolgreich durchlaufen
2. Qualität des Programmcodes: Übersichtlichkeit, Einhaltung von Programmierrichtlinien
3. Qualität der Dokumentation: knapp, aber verständlich und vollständig

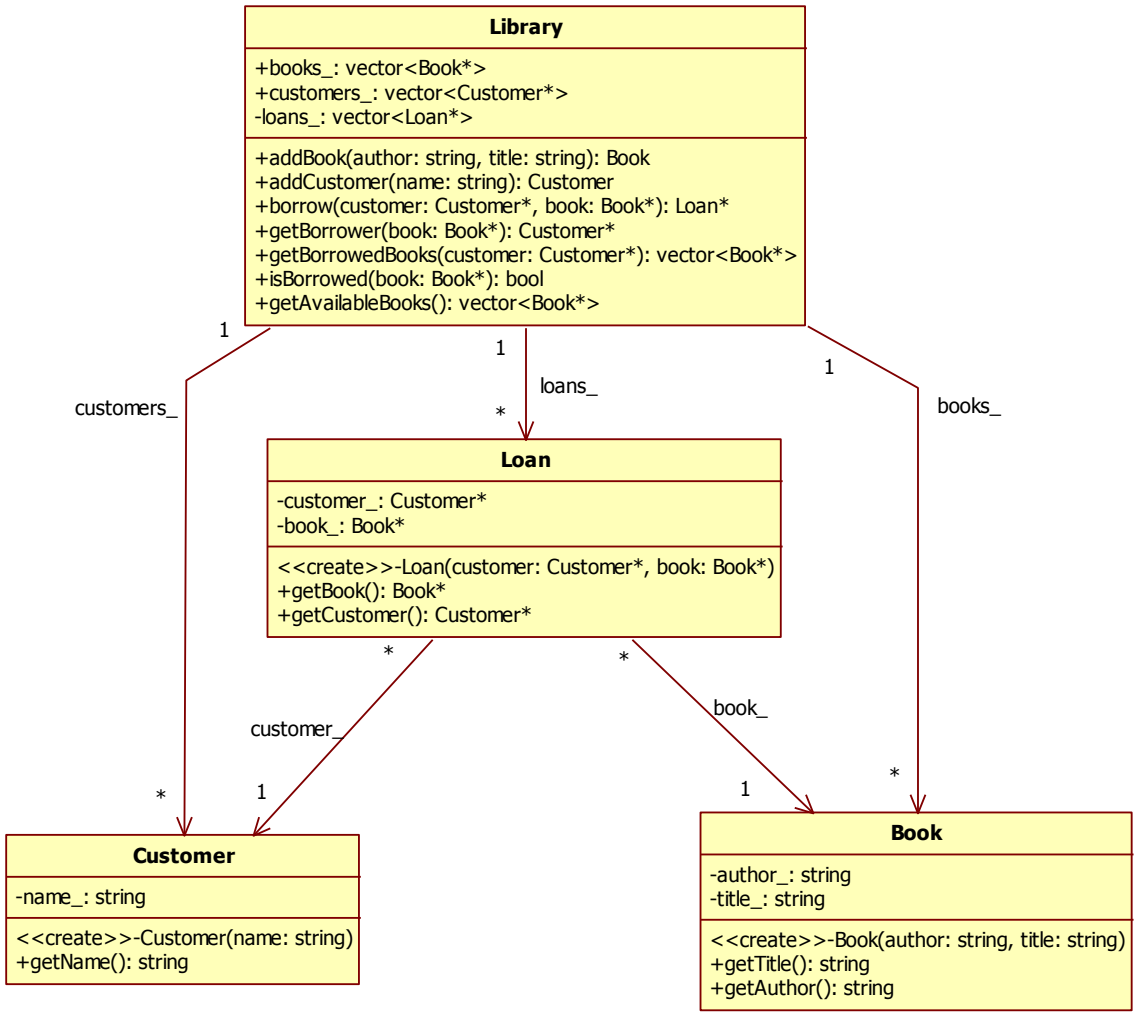
2 Vorbereitung

Legen Sie ein neues NetBeans Projekt benannt nach Ihrem Nachnamen an.

Tipp: File → New Project ... → C/C++ → C/C++ Application

3 Aufgabe: Bibliotheksanwendung

In dieser Klausur werden Sie prototypisch eine Bibliotheksanwendung entwickeln, z.B. für das Entleihen von Büchern aus der Hochschulbibliothek. Das folgende UML Diagramm gibt eine Übersicht über die gesamte zu erstellende Anwendung.



Entwickeln Sie die Anwendung Schritt für Schritt in der angegebenen Reihenfolge. Insgesamt gibt es 100 Punkte zu erreichen.

4 Klasse Customer (10 Punkte)

1. Erstellen Sie eine neue Klasse `Customer` (Kunde).
   Tipp: File → New File… → C++ → C++ Class

2. Erstellen Sie eine private Instanzvariable `name_` vom Typ `string`.
   Tipp:
   ```
   #include <string>
   using namespace std;
   ```

3. Entwickeln Sie die Funktion `string getName()` mit der üblichen Getter-Semantik.

4. Entwickeln Sie einen Konstruktor `Customer(string name)`, in dem die Instanzvariable `name_` initialisiert wird.

5. Stellen Sie sicher, dass Ihr Code fehler- und warnungsfrei kompiliert.

5 Klasse Book (10 Punkte)

1. Erstellen Sie eine neue Klasse `Book` (Buch).
   Tipp: File → New File… → C++ → C++ Class

2. Erstellen Sie die privaten Instanzvariablen `author_` und `title_` vom Typ `string`.
   Tipp:
   ```
   #include <string>
   using namespace std;
   ```

3. Entwickeln Sie die Funktionen `string getAuthor()` und `string getTitle()` mit der üblichen Getter-Semantik.

4. Entwickeln Sie einen Konstruktor `Book(string author, string title)`, in dem die Instanzvariablen `author_` und `title_` initialisiert werden.

5. Stellen Sie sicher, dass Ihr Code fehler- und warnungsfrei kompiliert.

6 Klasse Loan (10 Punkte)

1. Erstellen Sie eine neue Klasse `Loan` (Ausleihe).
   Tipp: File → New File… → C++ → C++ Class

2. Erstellen Sie die privaten Instanzvariablen `customer_` als Zeiger auf ein `Customer` Objekt und `book_` als Zeiger auf ein `Book` Objekt.

3. Entwickeln Sie die Funktionen `Book* getBook()` und `Customer* getCustomer()` mit der üblichen Getter-Semantik.

4. Entwickeln Sie einen Konstruktor `Loan(Customer* customer, Book* book)`, in dem die Instanzvariablen `customer_` und `book_` initialisiert werden.

5. Stellen Sie sicher, dass Ihr Code fehler- und warnungsfrei kompiliert.

7 Klasse Library (40 Punkte)

1. Erstellen Sie eine neue Klasse `Library`.
   Tipp: File → New File… → C++ → C++ Class

2. (2,5 Punkte) Erstellen Sie die privaten Instanzvariablen `books_`, `customers_` und `loans_` als Vektoren von Zeigern auf `Book`, `Customer`, bzw. `Loan` Objekte.
   Tipp:
   ```
   #include <vector>
   using namespace std;
   ```

3. (2,5 Punkte) Entwickeln Sie die Funktion `Book* addBook(string author, string title)` (Buch hinzufügen). In ihr wird ein neues `Book` Objekt angelegt und `author` und `title` gesetzt. Das `Book` Objekt wird dem Vektor `books_` hinzugefügt und zurückgegeben.
   Tipp: verwenden Sie `vector::push_back`

4. (2,5 Punkte) Entwickeln Sie die Funktion `Customer* addCustomer(string name)` (Kunde hinzufügen). In ihr wird ein neues `Customer` Objekt angelegt und `name` gesetzt. Das `Customer` Objekt wird dem Vektor `customers_` hinzugefügt und zurückgegeben.
   Tipp: verwenden Sie `vector::push_back`

5. (2,5 Punkte) Entwickeln Sie die Funktion `Loan* borrow(Customer* customer, Book* book)` (ausleihen). In ihr wird ein neues `Loan` Objekt angelegt, welches auf die als Parameter übergebenen `Customer` und `Book` Objekte verweist. Das `Loan` Objekt wird dem Vektor `loans_` hinzugefügt und zurückgegeben.
   Tipp: verwenden Sie `vector::push_back`

6. (10 Punkte) Entwickeln Sie die Funktion
   `Customer* getBorrower(Book* book)`
   (Ausleihender). Sie liefert den ersten `Customer`, der `book` ausgeliehen hat. Falls es keine Ausleihe für `book` gibt, wird `NULL` zurückgeliefert.

7. (10 Punkte) Entwickeln Sie die Funktion
   `vector<Book*> getBorrowedBooks(Customer* customer)`
   (entliehene Bücher). Sie liefert für einen `customer` alle `Book` Objekte, die er geliehen hat. Falls `customer` aktuelle keine Ausleihen hat, wird ein leerer Vektor zurückgeliefert.

8. (10 Punkte für 8. und 9. zusammen) Entwickeln Sie die Funktion
   `bool isBorrowed(Book* book)`
   (ist entliehen). Sie testet, ob `book` derzeit entliehen ist.
   Tipp: verwenden Sie `getBorrower`

9. Entwickeln Sie die Funktion
   `vector<Book*> getAvailableBooks()`
   (verfügbare Bücher). Sie liefert alle Bücher zurück, die derzeit nicht entliehen sind. Gibt es keine verfügbaren Bücher, so wird ein leerer Vektor zurückgegeben.

10. Stellen Sie sicher, dass Ihr Code fehler- und warnungsfrei kompiliert.

8 Tests (20 Punkte)

1. Entwickeln Sie in der Datei `main.cpp` die Prozedur
   `void test(bool testResult)`
   Hat `testResult` den Wert `true`, so wird eine neue Zeile mit dem Text `"Success"` auf der Konsole ausgegeben, andernfalls `"Failure"`.
   Tipp:
   ```
   #include <iostream>
   using namespace std;
   ```

2. Erstellen Sie im Hauptprogramm die folgenden Testobjekte und speichern Sie diese in entsprechenden Variablen:

   a. Ein `Library` Objekt

   b. `Book` Objekte via `addBook` mit Autor / Titel `"Breymann" / "C++"`, `"Gosling" / "Java"` und `"Goldberg" / "Smalltalk"`.

   c. `Customer` Objekte via `addCustomer` mit Namen `"Huber"` und `"Müller"`

3. Führen Sie die folgenden Ausleihen mittels `borrow` durch und speichern Sie die `Loan` Objekte in entsprechenden Variablen:

    a. `"Huber"` leiht `"Breymann" / "C++"`

    b. `"Huber"` leiht `"Gosling" / "Java"`

4. Entwickeln Sie die nachfolgenden Tests unter Verwendung Ihrer Prozedur `test`:

5. Der Autor des Buchs der ersten Ausleihe ist `"Breymann"`
   Tipp: verwenden Sie zum Vergleich die String-Methode `compare` – der Rückgabewert `0` zeigt String-Gleichheit an

6. Der Titel des Buchs der ersten Ausleihe ist `"C++"`

7. Der Kundenname der ersten Ausleihe ist `"Huber"`

8. `getBorrower` liefert für das erste Buch (Autor / Titel: `"Breymann" / "C++")` den ersten Kunden (Name: `"Huber"`)
   Tipp: vergleichen Sie nicht die Strings, sondern die Zeiger auf das `Customer` Objekt.

9. `getBorrower` liefert für das zweite Buch (Autor / Titel: `Gosling" / "Java")` auch den ersten Kunden (Name: `"Huber"`)

10. `getBorrower` liefert für das dritte Buch (Autor / Titel: `"Goldberg" / "Smalltalk") NULL` (derzeit nicht verliehen).

11. `getBorrowedBooks` liefert für den ersten Kunden (Name: `"Huber"`) einen Ergebnisvektor der Länge 2 (Huber hat zwei Ausleihen)
    Tipp: verwenden Sie die Vector-Methode `size`

12. `getBorrowedBooks` liefert für den zweiten Kunden (Name: `"Müller"`) einen Ergebnisvektor der Länge 0 (Müller hat derzeit keine Ausleihe).

13. `isBorrowed` liefert für das erste Buch (Autor / Titel: `"Breymann" / "C++") true`

14. `isBorrowed` liefert für das zweite Buch Buch (Autor / Titel: `Gosling" / "Java") true`

15. `isBorrowed` liefert für das dritte Buch Buch (Autor / Titel: `"Goldberg" / "Smalltalk") false`

16. `getAvailableBooks` liefert einen Ergebnisvektor der Länge 1 (derzeit ein verfügbares Buch)

17. Stellen Sie sicher, dass Ihr Code fehler- und warnungsfrei kompiliert.

18. Führen Sie alle Tests aus. Das Ergebnis auf der Konsole sollte wie folgt aussehen:

```
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Success
Press [Enter] to close the terminal…
```

9 Dokumentation (10 Punkte)

Dokumentieren Sie die Außensicht Ihrer Anwendung, also alle öffentlichen Operationen in den `*.h`-Dateien in deutscher oder englischer Sprache.

Beispiel- Dokumentation für `Library::borrow`:

```
/*
 * Performs a new loan
 *
 * Parameters:
 * - customer: reference to Customer object representing
 *        the borrowing customer
 * - book: reference to the Book object representing
 *        the borrowed book
 *
 * Returns: reference to new Loan object
 */
Loan* borrow(Customer* customer, Book* book);
```

Viel Erfolg!