Studienleistung

Die Studienleistung für das Fach „Grundlagen der praktischen Informatik“ im zweiten Semester besteht aus der Umwandlung der Studienleistung des ersten Semesters (Verwaltungsprogramm in C) in eine JAVA-Anwendung. Dabei ist es nicht erforderlich, nochmals eine vollständige Dokumentation mit Ablaufplan etc. zu erstellen. Es wird gefordert, den ausgearbeiteten, kompilierbaren und lauffähigen JAVA-Quelltext per E-Mail an den Übungsleiter zu schicken (peter.misch@fh-heidelberg.de) mit Angabe des Namens und der Matrikelnummer.

Die Abgabe der Studienleistung ist Voraussetzung
für die Wertung der Klausurergebnisse.

Lernziel der Studienleistung ist es, den Unterschied (und auch die Gemeinsamkeiten) zwischen einer prozeduralen und einer objektorientierten Programmierlösung zu verstehen und die Sprachmittel effizient einsetzen zu können. Programmdetails der C-Anwendung, für die es keine entsprechenden Funktionalitäten in Java gibt, sollen mit eigenen kreativen Mitteln gelöst werden (z.B. Bildschirmlöschen).

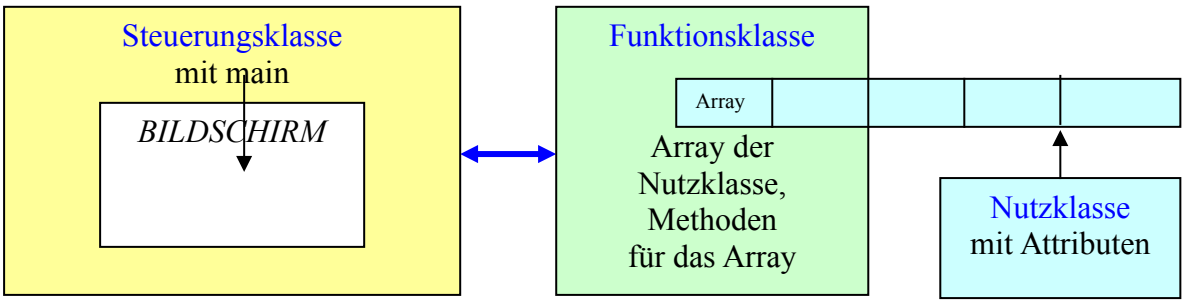
Die Ausarbeitung der Studienleistung basiert prinzipiell auf einem angepassten 3-Schichtenmodell, das die Aufgabenbereiche aller beteiligten Klassen definiert. Durch die Einhaltung und korrekte Umsetzung dieses Architekturmodells werden die didaktischen Voraussetzungen geschaffen, um in fortgeschrittenen Semestern die Wirkungsweise von Client-Server-Programmen zu verstehen. Verteilte Anwendungen sind die Basis aller modernen eBusiness-Programme.

Ein Schichtenmodell unterscheidet die Aufgabenbereiche der beteiligten Klassen nach ihrer grundlegenden Funktionalität. Hierbei definiert die Ansichtsklasse das graphische Frontend, in dem der Nutzer die Datenein- und -ausgabe erledigt. Davon getrennt ist die Steuerungsklasse, die Anwendereingaben umsetzt und an die Funktionsklasse weitergibt. Die Funktionsklasse beinhaltet die eigentliche Aufgabenerfüllung der Anwendung. Sie arbeitet mit einem Datencontainer (Array), das aus Objekten der Nutzklasse besteht.

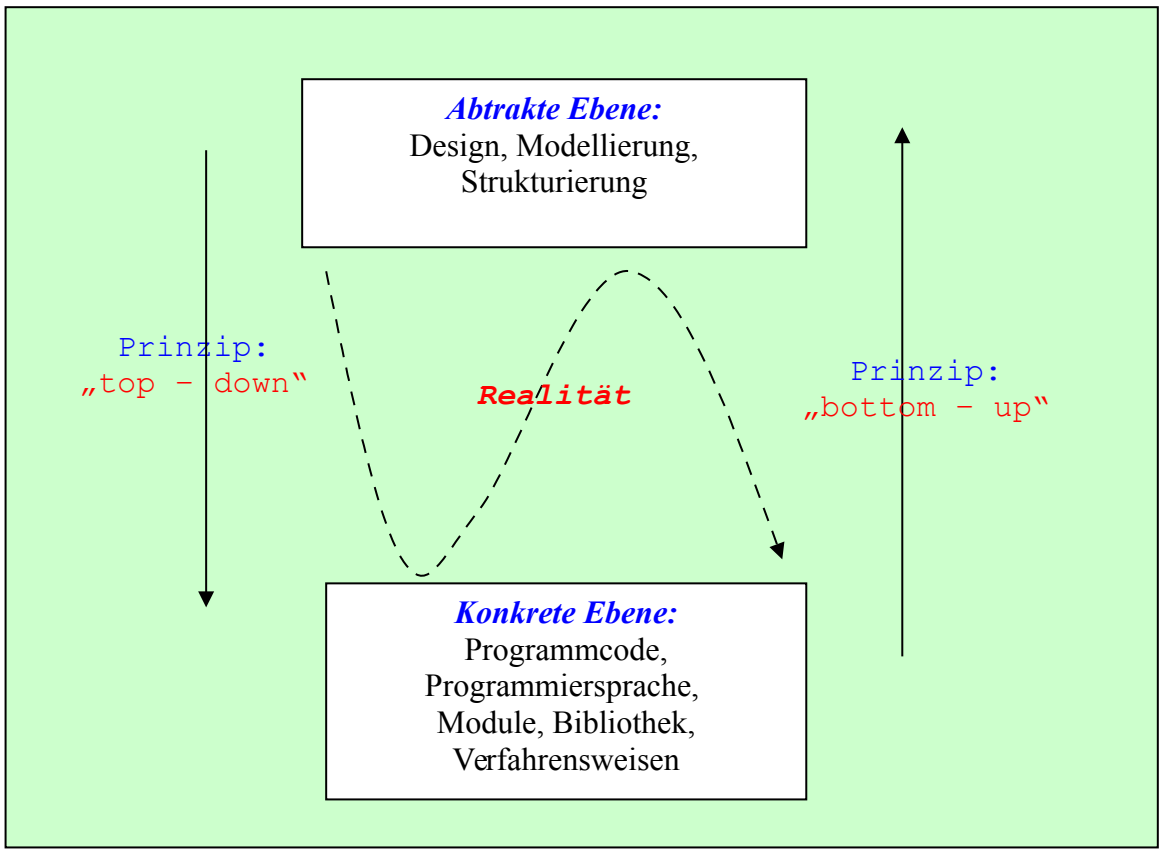
Bei der Ausarbeitung müssen folgende Regeln eingehalten werden:

1. *Es gibt keinen direkten Zugriff auf Attribute der Nutzklasse.*
2. *Der Datenaustausch und die Kommunikation zwischen den Klassen erfolgt prinzipiell nur über Methodenaufrufe.*
3. *Bildschirmein- und Ausgaben erfolgen nur im Frontend, nicht in der Funktions- oder Nutzklasse.*

In einfachen Konsolenprogrammen lässt sich die Bildschirmein- und Ausgabe noch nicht so eindeutig von der Steuerungsklasse trennen, wie das in einer GUI-Anwendung möglich ist. Es wird deshalb eine Anwendung implementiert, die aus folgenden Klassengruppen besteht.

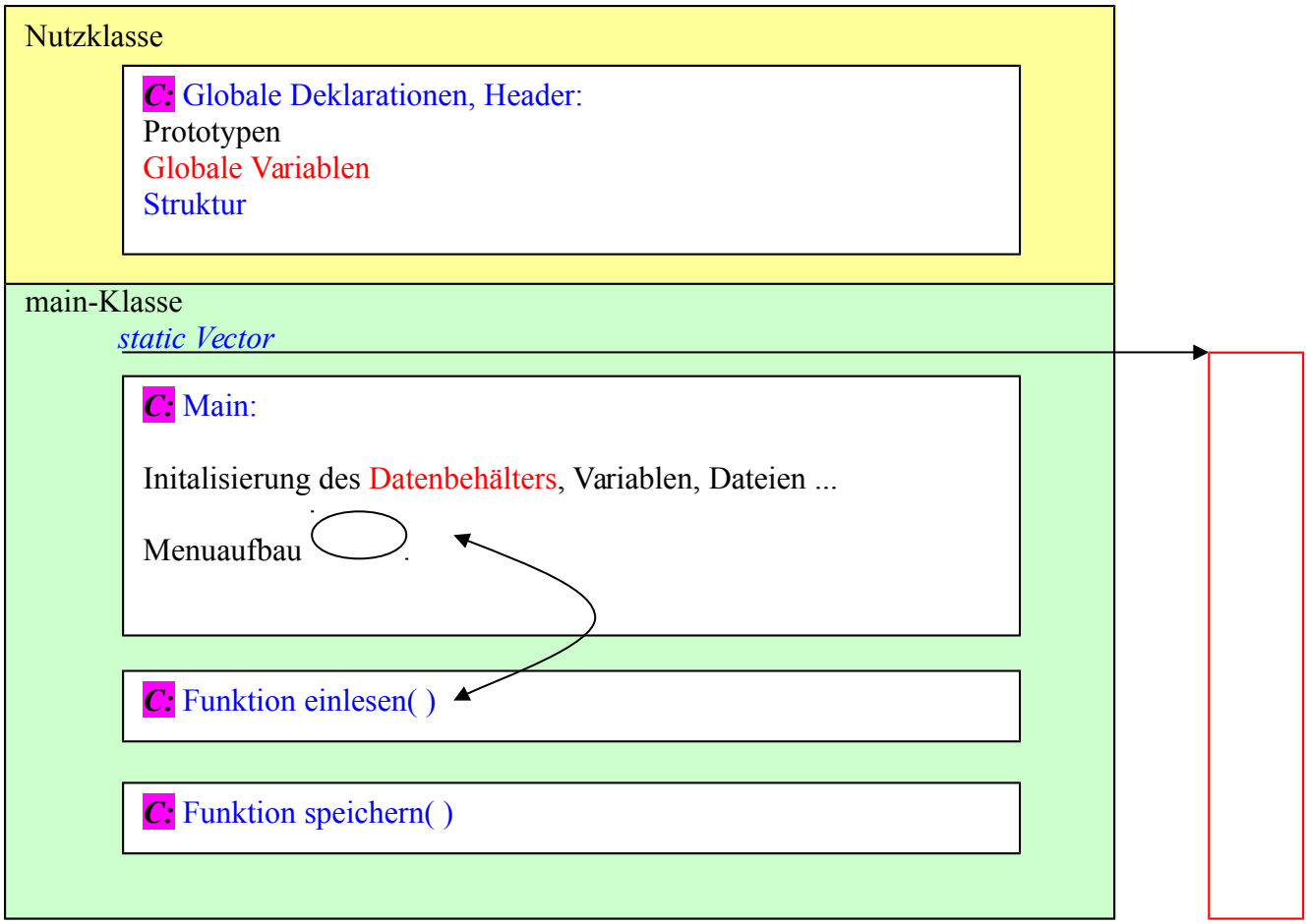


Im folgenden Abschnitt werden konkrete Hinweise gegeben, wie ein existierendes C-Programm in eine objektorientierte JAVA-Anwendung umgewandelt werden kann. Dazu müssen zuerst die möglichen Aspekte (Blickwinkel) auf eine Anwendung dargestellt werden. Ein wichtiges Hilfsprinzip beim objektorientierten Programmentwurf ist bekanntlich die Unterscheidung zwischen der *Abstrakten Modellierungsebene* und der *Konkreten Implementierungsebene*. Generell gilt, dass der erste Programmentwurf möglichst immer auf der Abstrakten Ebene erfolgen sollte. Zunächst sollte das Klassenmodell, das die generelle Programmstrukturierung und die daran beteiligten Klassen darstellt, erstellt werden. Allerdings zeigt die Praxis, dass man sich ständig zwischen diesen Ebenen hin- und herbewegt, um nötige konkrete Änderungen vorzunehmen oder das abstrakte Modell zu korrigieren.



Es ist ratsam, sich bei der Umwandlung der C-Studienleistung schrittweise an den endgültigen objektorientierten Entwurf anzunähern. Im folgenden Abschnitt werden zwei Zwischenlösungen vorgeschlagen, die dabei helfen können, den Überblick nicht zu verlieren. Es kann sehr hilfreich sein, immer wieder lauffähige Zwischenversionen zu erstellen, die ausgetestet werden können und als Vorbereitung auf den nächsten Schritt dienen.

LÖSUNG 1 (nur teilweise objektorientiert)

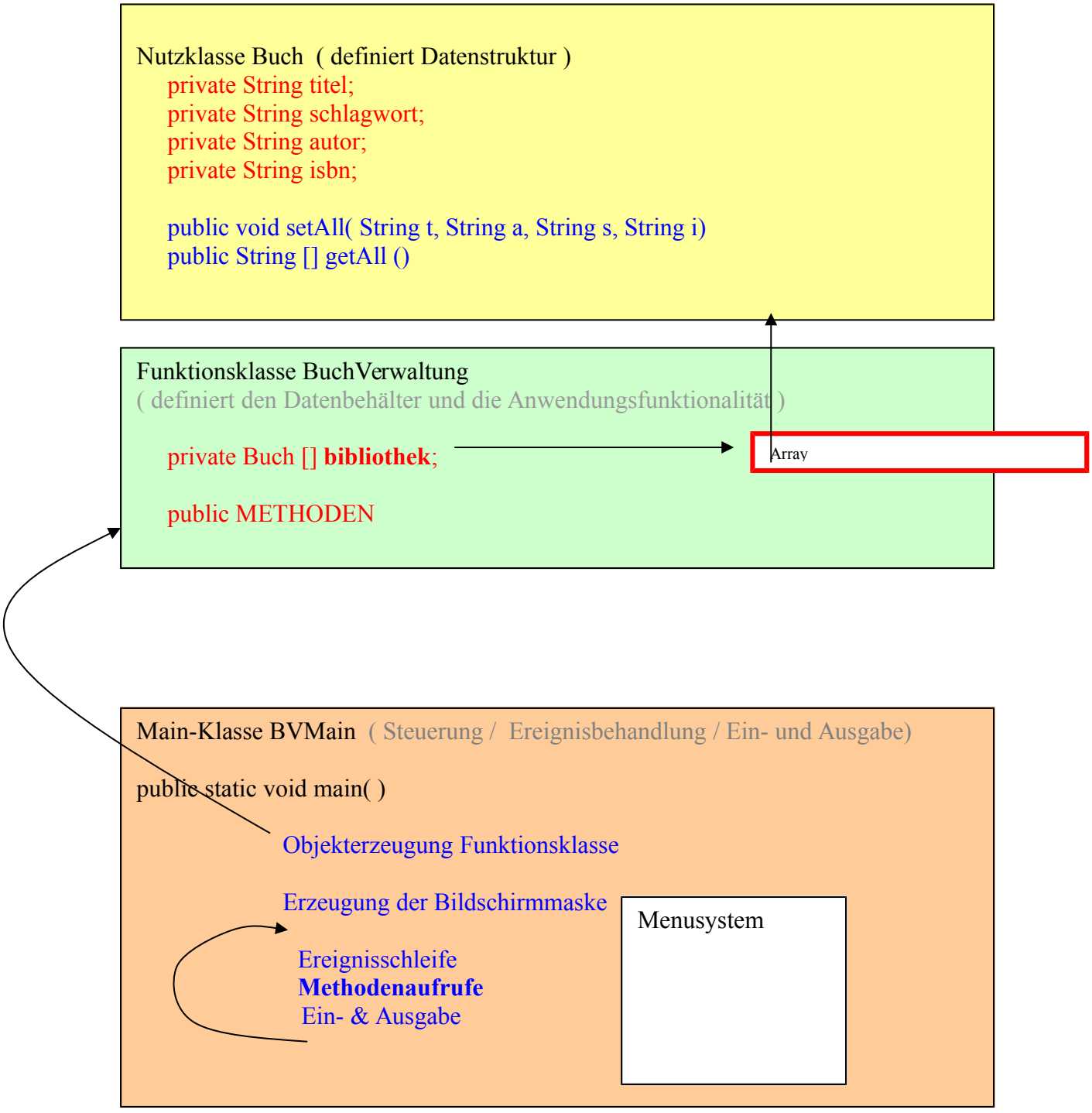


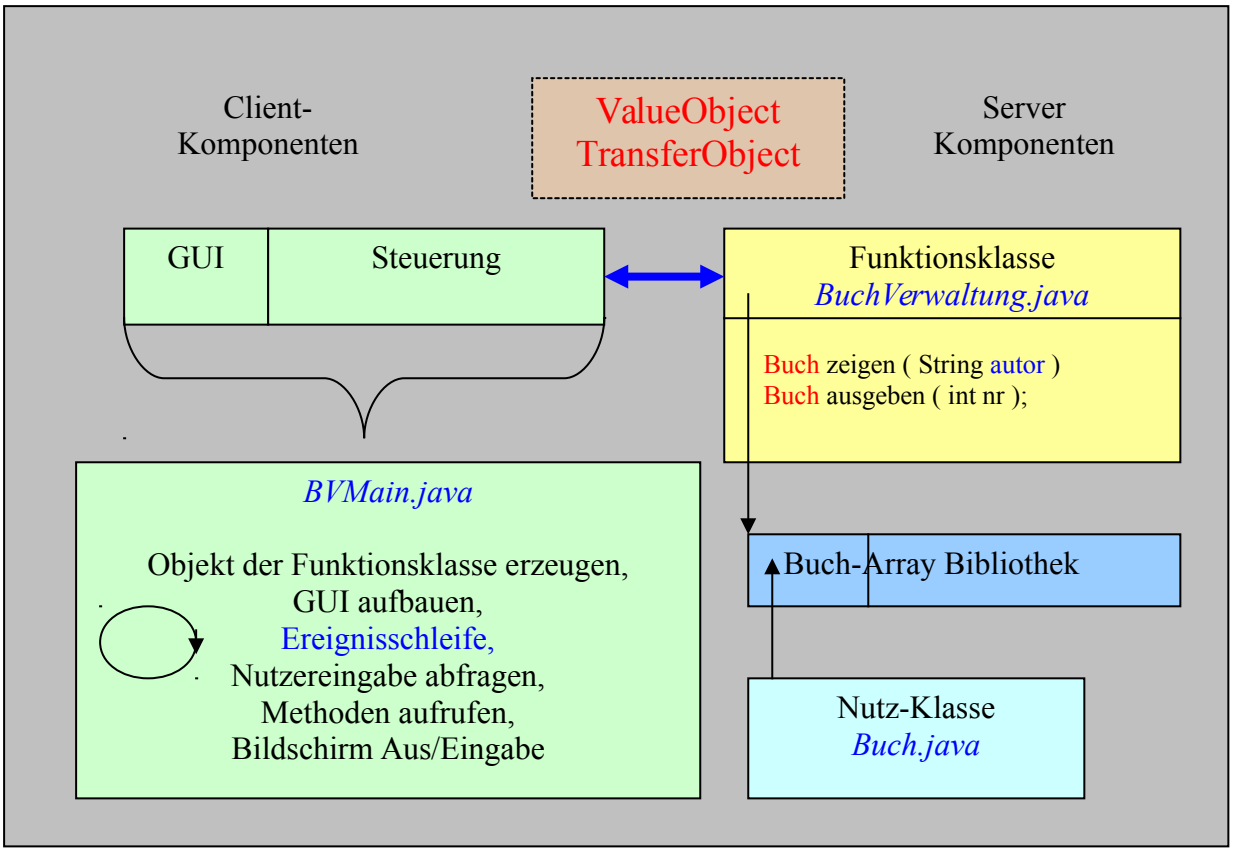
Vorgehensweise:

1. C-Struktur in eine Java-Klasse (Nutzklasse) umwandeln.
2. Main in eine Java-Klasse setzen (main-Hüllklasse), die nicht zur Objekterzeugung dient.
3. Datenbehälter (das Array aus Objekten der Nutzklasse) - muss static sein.
4. Alle C-Funktionen werden zu static-Methoden in der Hüllklasse.

Diese Lösung setzt die Prinzipien der Objektorientierung noch nicht durchgängig um. So müssen bspw. alle Methoden als STATIC gekennzeichnet werden, weil es gar kein Objekt dieser Klasse gibt. Als erster Schritt zur Umsetzung eines C-Programms in eine echte Java-Anwendung ist diese Vorlage aber gut zu gebrauchen.

Lösung 2 (objektorientiert)





Lösung 3 -> **Bibliothek**