Examination
Computer Programming (CP)

Prof. Dr.-Ing. Christian Heller <christian.heller@ba-leipzig.de>

Personal Data	
First and Last Name	Ihre Daten werden von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten.
Matriculation Number	
Subject and Year	CS 2016-2
Login	Das Login "klaus..." muss unbedingt schriftlich auf der Anwesenheitsliste festgehalten werden, da sonst keine Zuordnung des Logins zu Ihrem Namen und damit keine Korrektur der Klausur möglich ist!

Examination Data	
Date	
Duration [min]	108 von insgesamt 180
Maximum Points [Point]	60
Permitted Study Aids	Dokumentation im lokalen Netzwerkverzeichnis (Intranet); NICHT gestattet sind Kommunikationsmöglichkeiten (Internet) oder Anmeldung via SSH auf dem Rechner "fileserv", wo Ihr Homeverzeichnis liegt, oder eine Anmeldung mit Ihrem Klausur-Login nach Ende der Prüfung. Dies kann leicht geprüft werden (last cs12*, Server-Log-Dateien). Bitte unterlassen Sie also Täuschungsversuche in Ihrem eigenen Interesse.
Remarks	Hinterlegen Sie alle Programme und Antworten in elektronischer Form! Es wird kein Papier angenommen. Möchten Sie Lösungen erläutern, so nutzen Sie Quelltext-Kommentare oder legen eine Text-Datei an. Speichern Sie sämtliche Daten im HOME-Verzeichnis des Nutzers, d.h. unter Windows auf Laufwerk H:\ (NICHT auf C:\ oder "Eigene Dateien")! Idealerweise legen Sie dort ein Unterverzeichnis namens "klausur" an. Lesen Sie die Aufgaben komplett durch, bevor Sie sie lösen! Die Reihenfolge der Lösung ist Ihnen überlassen. Probieren Sie immer, eine Aufgabe zu lösen, da auch auf richtige Teile nicht vollständiger Lösungen Punkte vergeben werden! Falls vom Prinzip her richtig, so werden auch alternative Lösungen akzeptiert. Sie dürfen beliebig viele Bildschirmausgaben von Werten in den Quelltext einbauen, um ein Programm besser nachvollziehen zu können. Diese Aufgabenstellung in Papierform können Sie nach dem Ende der Klausur behalten.

Evaluation											
Task	1	2	3	4	5	6	7	8	9	10	Summe
Points	10	10	10	20	10	0	0	0	0	0	60

Ihre Programmierkenntnisse werden anhand kleiner, voneinander unabhängiger Java-Programme geprüft.

Task 1: Structured Programming [10]

Ziel dieser Aufgabe ist die Ausgabe einer Matrix von Produkten zweier Faktoren.

a) Erstellen Sie eine Startklasse namens *Launcher* mit statischer *main*-Methode! [2]

b) Iterieren Sie mittels Zählschleife von 1 bis 10! [2]

c) Schachteln Sie innerhalb der ersten eine zweite Zählschleife mit den gleichen Werten! [2]

d) Lassen Sie das Produkt der beiden Schleifenlaufvariablen auf der Konsole ausgeben! [2]

e) Formatieren Sie die Ausgabe so, dass jede neue Zahl durch eine Tabulatorweite eingerückt wird! Fügen Sie außerdem am Ende einer Folge von zehn Zahlen einen Zeilenumbruch ein! [2]

Task 2: String [10+2+2]

Ziel dieser Aufgabe ist die Verschlüsselung eines Textes, wobei Leerzeichen ignoriert werden.

Alles passiert in der *main*-Methode. Gegeben seien dafür zwei lokale Variablen:

```
String alphabet = "abcdefghijklmnopqrstuvwxyz";
String text = "Heute ist schönes Wetter.";
```

a) Wandeln Sie im Text alle Groß- in Kleinbuchstaben um! [2]

b) Iterieren Sie mittels Zählschleife über den gesamten gegebenen Text! [2]

c) Ersetzen Sie jeden einzelnen Buchstaben durch seine Positionsnummer im Alphabet! [2]

d) Speichern Sie die resultierenden Zahlen-Zeichen in einem Objekt der Klasse *StringBuilder*, wobei sie jeweils durch ein Leerzeichen zu trennen sind! Geben Sie dieses am Ende auf der Konsole aus! [2]

e) Ignorieren Sie alle Zeichen, die nicht im gegebenen Alphabet vorkommen! [2]

Hinweis: Herausfiltern mittels *if*-Konstrukt, noch vor der Speicherung im *StringBuilder*-Objekt.

Zusatz: f) Nutzen Sie ein *switch-case*-Konstrukt, um Sonderzeichen durch ihr Pendant ohne Pünktchen zu ersetzen, also 'ä' durch 'a' usw.! [2]

Zusatz: g) Ergänzen Sie das Pendant für Umlaute durch ein 'e' sowie das Pendant für 'ß' durch ein 'z' als zweiten Buchstaben! [2]

Beispiel: Umlaut 'ä' wird ersetzt durch "ae". Das scharfe 'ß' wird ersetzt durch "sz".

Task 3: Procedural Programming [10]

Ziel dieser Aufgabe ist ein Vergleich der Klassen String, StringBuilder und StringBuffer.

a) Erstellen Sie eine Methode namens *testString*, welche ein Argument vom Typ *int* entgegennimmt, aber keinen Rückgabewert liefert! Deklarieren Sie darin eine lokale Variable des Types *String*, welche initial leer ist! [2]

b) Verwenden Sie eine Zählschleife, um der *String*-Variablen den Buchstaben "*x*" anzuhängen! Die Schleifenzyklen werden durch den als Argument übergebenen *int*-Wert vorgegeben. [2]

c) Bestimmen Sie vor und nach Ausführung der Schleife die aktuelle Zeit in Millisekunden und geben Sie deren Differenz am Ende der Methode auf der Konsole aus! [2]

Hinweis: Die Klasse *System* bietet eine geeignete Methode an.

d) Erstellen Sie nach dem Muster der eben erstellten Methode *testString* zwei weitere Methoden, welche jedoch *StringBuilder* und *StringBuffer* verwenden und anzupassen sind! [2]

e) Rufen Sie alle drei Methoden aus der *main*-Methode heraus auf! Nutzen Sie das erste Kommandozeilenargument für eine Fallunterscheidung nach "String", "StringBuilder" oder "StringBuffer"! Konvertieren und übergeben Sie das zweite Kommandozeilenargument an die jeweils aufgerufene Methode! [2]

Task 4: Object Oriented Programming (OOP) [20]

Ziel dieser Aufgabe ist die Berechnung von Positionen eines Auftrages.

a) Erstellen Sie das Klassengerüst gemäß unten stehendem UML-Klassendiagramm!

[Klassen: 2; Attribute: 4; Methoden: 4]

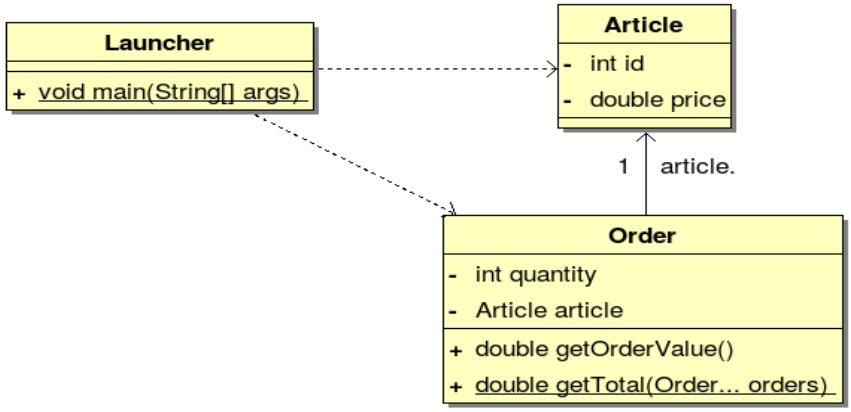
b) Erstellen (oder generieren) Sie beiderlei Zugriffsmethoden für alle Attribute! [2]

c) Geben Sie den Klassen *Article* und *Order* je einen Konstruktor, der die Attribute mit an ihn übergebenen Argumenten initialisiert! [2]

d) Implementieren Sie in der Klasse *Order* eine Methode namens *getOrderValue*, welche das Produkt aus Anzahl und Preis des aktuellen Artikels zurückliefert! [2]

e) Implementieren Sie in derselben Klasse außerdem eine Klassenmethode namens *getTotal*, welche eine unbestimmte Anzahl an *Order*-Objekten als Argument entgegennimmt! Iterieren Sie mittels *for-each*-Schleife durch sämtliche *Order*-Objekte und liefern Sie die Summe ihrer Bestellwerte (*getOrderValue*) als Ergebniswert der Methode zurück! [2]

f) Erstellen Sie in der *main*-Methode der Startklasse *Launcher* zwei *Article*-Objekte und damit dann zwei *Order*-Objekte, wobei beliebige Initialwerte verwendet werden können! Geben Sie schließlich die per *getTotal* ermittelte Endsumme auf der Konsole aus! [2]



Task 5: Exception Handling [10]

Ziel dieser Aufgabe ist die Meldung eines Fehlers bei Auftreten eines Null-Wertes.

a) Erstellen Sie eine Startklasse mit einer Konstanten namens *MAX*, der der Wert *500* zuzuweisen ist! [2]

b) Erstellen Sie eine Klassenmethode namens *getRandom*, welche eine Ganzzahl zurück liefert! Kennzeichnen Sie in der Methodensignatur, dass die Methode eine Standard-Ausnahme werfen kann! [2]

c) Erzeugen Sie in der Methode eine Zufallszahl *z* im Bereich zwischen 0 <= z < 100 und geben Sie sie als Ergebniswert zurück! [2]

d) Lassen Sie die Methode eine Ausnahme werfen, falls die Zahl *z* den arithmetischen Wert Null hat! [2]

e) Rufen Sie die Methode *getRandom* in der *main*-Methode innerhalb einer Zählschleife auf! Lassen Sie die Schleife von Null bis zum Wert der Konstanten *MAX* laufen! Geben Sie die gelieferte Zufallszahl auf der Konsole aus -- oder eine Fehlermeldung, falls eine Ausnahme geworfen wird! [2]

Viel Erfolg!