# Prüfung
# *Computer Programming (CP)*

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

| **Student** | |
|---|---|
| Vor- und Nachname | Ihre Daten werden von der Klausuraufsichtsperson schriftlich auf der Anwesenheitsliste festgehalten. |
| Matrikelnummer | |
| Studienrichtung und Jahr | CS 2018-2 |
| Anmeldename | 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! |

| **Prüfung** | |
|---|---|
| Datum | März 2019 |
| Dauer [min] | 120 |
| Hilfsmittel | Dokumentation im lokalen Netzwerk (Intranet) sowie Recherche im Internet.<br>NICHT gestattet:<br>* Kommunikation in jeglicher Form<br>* Anmeldung via SSH auf dem Rechner "fileserv"<br>* Anmeldung mit Klausur-Login nach Ende der Prüfung<br>Dies kann leicht geprüft werden (last cs16*, Server-Log-Dateien).<br>Bitte unterlassen Sie also Täuschungsversuche in Ihrem eigenen Interesse. |
| Bemerkungen | Hinterlegen Sie alle Programme und Antworten in elektronischer Form! Es wird kein Papier angenommen.<br>Möchten Sie Lösungen erläutern, so nutzen Sie Quelltext-Kommentare oder legen eine Text-Datei an.<br>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.<br>Lesen Sie die Aufgaben komplett durch, bevor Sie sie lösen! Die Reihenfolge der Lösung ist Ihnen überlassen.<br>Probieren Sie immer, eine Aufgabe zu lösen, da auch auf richtige Teile nicht vollständiger Lösungen Punkte vergeben werden!<br>Falls vom Prinzip her richtig, so werden auch alternative Lösungen akzeptiert.<br>Sie dürfen beliebig viele Bildschirmausgaben von Werten in den Quelltext einbauen, um ein Programm besser nachvollziehen zu können.<br>Bitte duplizieren Sie Ihre Quelltextdateien (workspace) NICHT, da beim Korrigieren dann beide durchsucht werden müssen, was sinnlosen Aufwand verursacht. Diese Aufgabenstellung in Papierform können Sie nach dem Ende der Klausur behalten. |

| **Bewertung** | | | | | | |
|---|---|---|---|---|---|---|
| **Aufgabe** | 1 | 2 | 3 | 4 | 5 | Summe |
| **Punkte** | 10 | 20 | 20 | 20 | 30 | 100 |

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

## Aufgabe 1: Operator [10]

Zweck: Berechnen der Quersumme einer Zahl.

a) Erstellen Sie eine statische Methode namens „berechneQuersumme“, welche eine Ganzzahl „z“ als Parameter entgegennimmt und eine Ganzzahl als Ergebniswert zurückliefert! [2]

b) Deklarieren Sie eine lokale Variable „summe“ des Typs „int“ und initialisieren Sie sie mit dem Wert 0! Geben Sie ihren Wert am Ende zurück! [2]

c) Fügen Sie eine „while“-Schleife ein, die läuft, solange die übergebene Zahl z nicht arithmetisch Null ist! [2]

d) Addieren Sie in der Schleife die letzte Ziffer der Zahl z zur Summe! [2]

Hinweis: Dies erreicht man durch eine Restbestimmung (Modulo) der Division der Zahl z durch 10.

e) Entfernen Sie nun die letzte Ziffer der Zahl z! [2]

Hinweis: Dies erreicht man durch Ganzzahl-Division der Zahl durch 10.

Ergebnis: Bei Aufruf der Methode wird die korrekte Quersumme auf der Konsole ausgegeben.

**Aufgabe 2: Array [20]**

Zweck: Ausgeben von Ganzzahlen und Wochentagen auf der Konsole.

a) Erstellen Sie ein Integer-Array der Größe 10! [2]

b) Befüllen Sie es mittels klassischer Zählschleife mit den Zahlen von 0 bis 900! [2]

c) Nutzen Sie eine for-each-Schleife, um den Inhalt des Arrays auf der Konsole auszugeben! [2]

d) Stellen Sie sicher, dass alle Zahlen hintereinander in EINER Zeile ausgegeben werden! Bauen Sie abschließend einen Zeilenumbruch ein! [2]

e) Ändern Sie den Schleifenkörper der Zählschleife zum Befüllen nun in der Weise, dass nunmehr die Zahlen von 100 bis 1.000 (statt 0 bis 900) ausgegeben werden! [2]

f) Erstellen Sie ein String-Array, dem die sieben Wochentagskürzel (Mo, Di usw.) als Literale zugewiesen werden! [2]

g) Verwenden Sie eine kopfgesteuerte while-Schleife als Endlosschleife! [2]

h) Spendieren Sie ihr eine Lauf- bzw. Zählvariable! [2]

i) Geben Sie in der Schleife alle Elemente des String-Arrays auf der Konsole aus! [2]

j) Bauen Sie in den Schleifenkörper ein passendes Abbruchkriterium ein! [2]

Ergebnis: Alle Zahlen und Wochentage wurden korrekt ausgegeben.

**Aufgabe 3: Structure [20]**

Zweck: Bestimmen des größten gemeinsamen Teilers zweier positiver ganzer Zahlen p und q mittels des Euklidischen Algorithmus’.

a) Belegen Sie die Variablen p und q jeweils mit einer positiven ganzen Zahl! [2]

b) Prüfen Sie, ob zwei Kommandozeilenargumente übergeben wurden! [2]

c) Konvertieren Sie im positiven Falle die beiden Argumente in Ganzzahlen, welche p und q zuzuweisen sind! [2]

d) Fahren Sie fort mit e), falls p < q ist, sonst mit f)! [2]

e) Vertausche die Belegung von p und q! [2]

f) Fahren Sie fort mit j), falls q == 0, sonst mit g)! [2]

g) Belegen Sie r mit dem Rest der Division p durch q! [2]

h) Belegen Sie p mit dem Wert von q und q mit dem Wert von r! [2]

i) Fahren Sie fort mit f)! Hinweis: Schleife. [2]

j) Geben Sie die Belegung von p als Ergebnis aus! [2]

Ergebnis: Der größte gemeinsame Teiler wurde auf der Konsole ausgegeben.

**Aufgabe 4: Procedure [20]**

Zweck: Transformieren zweier Schleifen in ein rekursives Programm.

Gegeben sei folgendes Programm zur Ausgabe einer Tabelle mit dem kleinen Einmaleins.

```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.print(i * j + "\t");
    }
    System.out.println();
}
```

Hinweis: Es erscheint sinnvoll, zunächst das gegebene Programm zum Laufen zu bringen und hiernach schrittweise umzubauen.

a) Erstellen Sie eine statische Methode für die innere Schleife, welche zwei Ganzzahlen als Parameter i und j entgegennimmt! [2]

b) Fügen Sie ein passendes Laufkriterium ein! [2]

c) Geben Sie das Produkt der beiden Parameter auf der Konsole aus! Inkrementieren Sie den zweiten Parameter j! [2]

d) Rufen Sie die Methode für die innere Schleife rekursiv auf! [2]

e) Fügen Sie anschließend einen Zeilenumbruch ein! [2]

f) Rufen Sie die neu implementierte Methode als Ersatz für die ursprüngliche innere Schleife innerhalb der äußeren Schleife auf! [2]

Hinweis: Der Parameter j ist vor dem Aufruf zu deklarieren und mit 1 zu initialisieren.

g) Lagern Sie schließlich auf ähnliche Weise die äußere Schleife in eine neue statische Methode aus, wobei hier nur der Ganzzahl-Parameter i zu übergeben ist! [2]

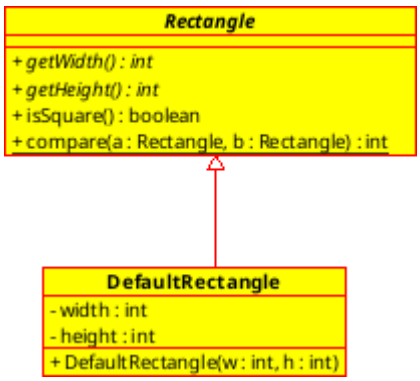
h) Fügen Sie ein passendes Laufkriterium ein! [2]

i) Inkrementieren Sie den Parameter i! Rufen Sie die Methode selbst rekursiv auf! [2]

j) Ersetzen Sie die ursprüngliche äußere Schleife in der „main“-Methode durch einen Aufruf der neu implementierten Methode! [2]

Ergebnis: Eine Tabelle mit zehn Zeilen und je zehn Spalten wurde ausgegeben.

**Aufgabe 5: Object [30]**




Zweck: Implementieren einer kleinen Software-Architektur.

Gegeben sei dazu ein Klassendiagramm in Unified Modeling Language (UML)-Notation.

Hinweise:

- Kursivschreibung bedeutet „abstract“: Rectangle, getWidth(), getHeight()
- Unterstreichung bedeutet „static“: compare()

Implementieren Sie die gegebene Architektur!

a) Klassen [2]

b) Signatur der abstrakten Methoden der abstrakten Klasse „Rectangle“ [2]

c) Signatur der übrigen Methoden der abstrakten Klasse „Rectangle“ [2]

d) Attribute der Klasse „DefaultRectangle“ [2]

e) Signatur des Konstruktors der Klasse „DefaultRectangle“ [2]

f) Implementieren Sie die Methode „isSquare“, worin ein Seitenvergleich durchzuführen ist! [2]

g) Berechnen Sie in der statischen Methode „compare“ den Flächeninhalt beider Rechtecke! [2]

h) Weisen Sie die Flächenwerte zwei neuen lokalen Variablen „x“ und „y“ zu! [2]

i) Geben Sie -1 zurück, wenn x kleiner als y ist; 0 wenn beide gleich sind; +1 sonst! [2]

j) Erzeugen Sie „get“-Methoden für die beiden Attribute der Klasse „DefaultRectangle“! [2]

k) Initialisieren Sie die Attribute im Konstruktor mit den übergebenen Parameterwerten! [2]

l) Berücksichtigen Sie die im Diagramm gezeigten Sichtbarkeiten! [2]

m) Erzeugen Sie in einer Startklasse mit „main“-Methode drei Rechteck-Objekte der Breite und Höhe (2, 3), (3, 2), (3, 3)! [2]

n) Testen Sie, ob es sich um Quadrate handelt! [2]

o) Vergleichen Sie r1 mit r2 und r1 mit r3! [2]

Ergebnis: Die Anwendung kann mit Rechtecken umgehen und funktioniert wie erwartet.

Auf der Konsole sollte bei richtiger Implementierung folgende Ausgabe erscheinen:

```
Is r1 a square? false
Is r2 a square? false
Is r3 a square? true
Comparison of r1 with r2: 0
Comparison of r1 with r2: -1
```

**Viel Erfolg!**