FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Konzepte moderner Programmiersprachen (KMPS)

(Wahlfach, Modulnummer 55685)

Wintersemester 2022/2023

Prof. Dr. Andreas Claßen

Prof. Dr. Heinrich Faßbender

Fachbereich 5 Elektrotechnik und Informationstechnik
FH Aachen

KMPS Prüfungen 2022/2023: Schriftliche Klausur

Die KMPS-Prüfung in der kommenden Prüfungsperiode ist als schriftliche Klausur an der FH geplant.

Geplanter Prüfungstermin: Donnerstag 2. Februar 2023, 8:30-10:30 Uhr.

Dauer der Klausur: 120 Minuten.
Die Klausur ist in deutscher Sprache.

Zwei Teile ("Faßbender" & "Claßen" Teil) mit gleichem Umfang (ungefähre Aufgabenzahl) & Gewichtung.
Als eine gemeinsame Klausur gehandhabt (keine "Mindestquote pro Teil", "max. 60 Min Zeit pro Teil" o.ä.).

In den Prüfungsperioden Juli/September 2023 findet die KMPS-Prüfung dann als mündliche Einzelprüfung (30 Minuten Dauer) statt.

Generelle Erläuterungen zur Klausur, Claßen Teil

Ziel: Verständnis der Konzepte

- Daher keine Programmieraufgaben, aber Konzepte anhand von (möglichst korrekten) Programmbeispielen erklären oder aufzeigen können.

Keine Einzeiler-Antworten bei Verständnisfragen "Erläutern Sie …".

Leserlich schreiben!

Klausurinfos KMPS, "Claßen Teil"

Beschreiben Sie, was man unter *Concurrency* und was man unter *Parallelismus* versteht. Grenzen Sie die beiden gegeneinander ab.

Beschreiben Sie das Scheduling nebenläufiger Programmteile in Go und in Erlang und grenzen Sie diese beiden Scheduling Konzepte gegeneinander ab.

Was versteht man unter *kooperativem Scheduling*? Nennen Sie dabei auch einen Vorteil und einen Nachteil dieses Scheduling Ansatzes. Beschreiben Sie das kooperative Scheduling in der Programmiersprache Go.

Klausurinfos KMPS, "Claßen Teil"

Wie wird der Wechsel der Kontrolle zwischen den nebenläufigen Goroutinen eines Go Programms veranlasst? Muss der Programmierer des Programms dies immer selbst programmieren?

Beschreiben Sie die Konzepte der Programmiersprachen Go und Erlang betreffend geteilte Daten zwischen nebenläufigen Programmteilen.

Nennen Sie den wesentlichen Unterschied zwischen Goroutinen und den Threads des zugrundeliegenden Betriebssystems, der dazu führt, dass Goroutinen nicht 1:1 auf Threads abgebildet werden.

Klausurinfos KMPS, "Claßen Teil"

Was versteht man in Go unter einem Tight Loop Szenario? Geben Sie ein Beispiel (Go Codeausschnitt) dafür an. Wie kann der Programmierer dieses Problem lösen?

Beschreiben Sie, wofür Channels in Go genutzt werden. Welche Arten von Channels gibt es? Für welche Kommunikationsszenarien werden diese genutzt? Mittels welcher Operationen können die Goroutinen solche Channels nutzen?

Wie verhalten sich die grundlegenden *send* und *receive* Operationen bei Unbuffered Channels?

Wie kann man Unbuffered Channels nutzen, was den Kontrollfluss von Goroutinen betrifft?

Beschreiben Sie, welche Varianten des `select` Befehls es in Go gibt. Es ist nicht die exakte Syntax dieser Befehlsvarianten gefragt, sondern eine Beschreibung des Konzepts (der Semantik) der Varianten.

Klausurinfos KMPS, "Claßen Teil"

Nennen Sie den wesentlichen Unterschied zwischen Erlang Prozessen und den Threads des Betriebssystems, der dazu führt, dass Erlang Prozesse nicht 1:1 auf Threads abgebildet werden.

Nennen Sie den Namen des Nebenläufigkeits-Konzepts von Erlang und beschreiben Sie es.

Begründen Sie, warum sich Erlang Prozesse über mehrere im Netzwerk verbundene Rechner verteilen lassen, ohne dass dies im Erlang Programm besonders berücksichtigt werden muss.

Klausurinfos KMPS, "Claßen Teil"

In Erlang möchte eine Funktion, die als Client agiert, einer anderen Funktion, die als Server agiert, einen Request (eine Anfrage, als Nachricht) schicken. Nennen Sie zwei Möglichkeiten, wie der Client den Server als "Nachrichtenziel" finden bzw. identifizieren kann, so dass die Nachricht dann beim richtigen Empfänger ankommt.

In einem Erlang Programm agiere eine nebenläufige Funktion als Client und eine andere nebenläufige Funktion als Server. Der Client schickt mehrere Anfrage-Nachrichten an den Server und erhält später mehrere Antwort-Nachrichten. Wie kann der Erlang Programmierer sicherstellen, dass der Client Code die empfangenen Nachrichten der richtigen entsprechenden (vorher geschickten) Anfrage zuordnen kann?

Klausurinfos KMPS, "Claßen Teil"

Beschreiben Sie die Best Practice Empfehlung, wie in Erlang Code mit Fehlersituationen umgegangen werden soll.

Bei der Erlang Programmierung gilt die Empfehlung: *"Let it crash"*. Beschreiben Sie, wie im Erlang Code trotz crashender Prozesse die Fehlersituation eingegrenzt und strukturiert behandelt werden kann, so dass es nicht zu einem Crash des gesamten Applikationscodes kommen muss.

Klausurinfos KMPS, "Claßen Teil"

Was versteht man im Erlang OTP Framework unter einem Supervisor Prozess und einem Worker Prozess, wenn es um die Behandlung von Fehlersituationen im Erlang Code geht? Welche Rolle spielen die beiden im Kontext der in Erlang programmierten Applikation?

Klausurinfos KMPS, "Claßen Teil"

Wofür benutzt das Erlang OTP Framework die Restart Strategien? Beschreiben Sie zwei verschiedene dieser Strategien, inklusive Beschreibung eines von Ihnen erdachten Anwendungsfalls, in dem die jeweilige Strategie angemessen ist.

Bei Erlang Applikationen, die auf das Framework OTP aufgebaut sind, besteht die Möglichkeit, bei Teilen der Applikation zur Laufzeit Code auszutauschen. Beschreiben Sie, welche Eigenschaften Erlang als Sprache besitzt, auf die diese Möglichkeiten dann aufbauen können.

Klausurinfos KMPS, "Claßen Teil"

JavaScript Interpreter bieten im Prinzip nur eine "Ausführungseinheit" für den JavaScript Code und nur eine Event Loop. Trotzdem ist es möglich, nebenläufige Programmteile zu programmieren.

1. Beschreiben Sie einen Mechanismus, durch den bestimmt wird, ob und wann die nebenläufigen Programmteile zur Ausführung kommen.
2. Zu welchen negativen Konsequenzen können die Einschränkungen "eine Ausführungseinheit" und "eine Event Loop" trotz des Vorhandenseins von Nebenläufigkeit führen, wenn man ausschließlich die Ausführung von JavaScript Code betrachtet? Denken Sie sich ein Beispiel aus, welches diesen Effekt illustriert, und beschreiben Sie es high-level.

Klausurinfos KMPS, "Claßen Teil"

Was versteht man unter dem JavaScript Callback Programmierstil? Geben Sie ein JavaScript Codebeispiel mit von ihnen erdachten JavaScript Funktionen an, welches das Callback Prinzip illustriert, und erläutern Sie ihr Beispiel.

Wie kann man bei der JavaScript Programmierung mittels einfacher Callbacks die Fehlerbehandlung programmieren? Geben Sie ein JavaScript Codebeispiel mit einer von ihnen erdachten JavaScript Callback Funktion an, welches illustriert, wie die Fehlerinformationen einer asynchronen Operation an ein Callback übergeben und von diesem verarbeitet werden könnten. Das Beispiel soll auch die Registrierung der Callback Funktion beinhalten.

Klausurinfos KMPS, "Claßen Teil"

Was versteht man bei der JavaScript Programmierung unter dem Problem der "Callback Hell" (oder "Pyramid of Doom")? Geben Sie ein JavaScript Codebeispiel mit von ihnen erdachten JavaScript Funktionen an, welches das "Callback Hell" Problem illustriert, und erläutern Sie ihr Beispiel.

Erläutern Sie anhand eines von Ihnen erdachten JavaScript Pseudo-Code-Beispiels das Konzept des *"Reactive Programming"*. Erläutern Sie insbesondere, auf welcher Art von Datenstrukturen dieser Ansatz basiert und welche Art von Operationen auf diesen Daten benutzt werden. Beschreiben Sie, welchen Vorteil dieser Programmieransatz bietet gegenüber der klassischen event-basierten Programmierung in JavaScript.

Klausurinfos KMPS, "Claßen Teil"

Was versteht man bei der JavaScript Programmierung unter dem Konzept der Promises? Was ist eine Promise? Wie wird sie im JavaScript Code programmiert? Geben Sie ein von Ihnen erdachtes JavaScript Pseudo-Code-Beispiel an, in dem eine von Ihnen erdachte JavaScript Funktion eine Promise erzeugt und zurückgibt. Erläutern Sie den Zweck der Promise mittels ihres Beispiels. Erläutern Sie auch, aus welchen Codekonstrukten sich die Promise zusammensetzt und wie diese Konstrukte in ihrem Beispiel genutzt werden, damit die Promise ihren Zweck erfüllen kann.

Klausurinfos KMPS, "Claßen Teil"

Was passiert bei einer JavaScript Promise, wenn im Executor der Promise die Funktion `resolve()` einmal mit dem Parameterwert `11` aufgerufen wird (`resolve(11)`) und anschließend mit dem Parameterwert `22` aufgerufen wird (`resolve(22)`)? Erläutern Sie insbesondere, zu welchem Ergebniswert und Status der Promise dies führt.
Was passiert bei einer JavaScript Promise, wenn im Executor der Promise zuerst `resolve(33)` aufgerufen wird und anschließend `reject(-1)` (d.h. Fehlercode `-1`)? Erläutern Sie insbesondere, zu welchem Ergebniswert und Status der Promise dies führt.

Klausurinfos KMPS, "Claßen Teil"

Es ist im JavaScript Code auf keine der folgenden alternativen Arten möglich, die Beispiel-Promise zu resolven:

```
let my_promise = new Promise( … );
resolve(my_promise);       // funktioniert nicht
my_promise.resolve(42); // funktioniert auch nicht
```

Erläutern Sie, wie bzw. von wo aus die Promise denn dann resolved werden kann und warum das Promise Konzept nicht erlaubt, die Promise auf die obigen zwei Arten zu resolven. Sie können in ihren Erläuterungen auch gerne den oben weggelassenen Teil des Codes (die drei Pünktchen) beispielhaft konkretisieren, wenn dies für ihre Erläuterungen notwendig sein sollte.

Klausurinfos KMPS, "Claßen Teil"

Wozu dient die `.then()` Methode bei JavaScript Promises? Erläutern Sie dies anhand eines von ihnen erdachten illustrativen JavaScript Codebeispiels. Erläutern Sie insbesondere auch, ob die Methode Parameter hat, wenn ja wieviele und wie diese genutzt werden, ob die Methode einen Rückgabewert hat und wenn ja, welche Funktion dieser erfüllt und welche Aufgabe die `.then()` Methode im Konzept der Nebenläufigkeit mittels Promises erfüllt.

Klausurinfos KMPS, "Claßen Teil"

Wozu dient die `.catch()` Methode bei JavaScript Promises? Erläutern Sie dies anhand eines von ihnen erdachten illustrativen JavaScript Codebeispiels. Erläutern Sie insbesondere auch, ob die Methode Parameter hat, wenn ja wieviele und wie diese genutzt werden, ob die Methode einen Rückgabewert hat und wenn ja, welche Funktion dieser erfüllt und welche Aufgabe die `.catch()` Methode im Konzept der Nebenläufigkeit mittels Promises erfüllt.
Erläutern Sie ferner die Relation zwischen der `.catch()` Methode von Promises und dem catch in `try`/`catch` Blöcken. Wie ist die Relation zwischen diesen beiden Konstrukten im Kontext der Programmierung mit Promises? Sollten beide benutzt werden?

Klausurinfos KMPS, "Claßen Teil"

Was versteht man unter der *Promise Absorption*? Erläutern Sie, wie dieser Mechanismus funktioniert und welchen Zweck er für die Programmierung mittels Promises erfüllt.
Was versteht man unter *Promise Chaining*? Erläutern Sie auch hier, wie dieser Mechanismus funktioniert und welchen Zweck er für die Programmierung mittels Promises erfüllt.

Klausurinfos KMPS, "Claßen Teil"

Gegeben eine JavaScript Promise, die schon bei ihrer Definition in den "resolved" Zustand versetzt wird. Auf dieser Promise werde mittels der `.then()` Methode ein Callback registriert.

```
let promise1 = Promise.resolve(42);
promise1.then( result => { console.log("Resultat: " + result); } );
console.log("Skript Ende!");
```

Wann wird der Callback ausgeführt werden? Geben Sie die relevanten Mechanismen des JavaScript Interpreters bzw. der JavaScript Plattform an, die zu diesem Verhalten führen.

Klausurinfos KMPS, "Claßen Teil"

Erläutern Sie das Konzept der Promise.all() Operation: Wozu wird diese Operation genutzt? Was sind die Parameter und was ist der Rückgabewert dieser Operation? Wie werden Fehlerszenarien durch diese Operation behandelt?

Klausurinfos KMPS, "Claßen Teil"

Was versteht man unter der "Promisification" einer JavaScript Library? Muss dazu die Library immer komplett neu geschrieben werden? Oder gibt es ein alternatives, effizienteres Vorgehen, dass sich bei manchen Libraries umsetzen ließ? Beschreiben Sie mittels JavaScript Pseudo-Code, wie dieses alternative, effiziente Verfahren funktioniert.

Klausurinfos KMPS, "Claßen Teil"

Beschreiben Sie anhand eines von Ihnen erdachten illustrativen JavaScript Codebeispiels, was das JavaScript async / await Konstrukt bei einer asynchronen, vom Programmierer definierten Funktion genutzt wird. Wofür wird der await Befehl genutzt, d.h. welche Funktion hat er in der technischen Realisierung der Nebenläufigkeit in JavaScript. Erläutern Sie dies insbesondere an ihrem Codebeispiel. Welche Rolle spielt das async Schlüsselwort für den Code der asynchronen Funktion bzw. für ihre Abarbeitung?

Klausurinfos KMPS, "Claßen Teil"

Beschreiben Sie für das JavaScript async / await Konstrukt, was die async Funktion für einen Rückgabewert besitzt. Erläutern Sie, welche Rolle der Rückgabewert der Funktion und der Rumpf der Funktion im Kontext der Nebenläufigkeit in JavaScript haben.

Klausurinfos KMPS, "Claßen Teil"

Erläutern Sie den Vorteil für den Programmierer der JavaScript Programmierung mittels async / await gegenüber einer Programmierung ausschließlich mit Promises (ohne Nutzung von async / await).

Klausurinfos KMPS, "Claßen Teil"

Gegeben eine JavaScript Promise, die innerhalb einer async Funktion definiert wird und schon bei ihrer Definition in den "resolved" Zustand versetzt wird. Auf diese Promise werde mittels des await Befehls gewartet.

```
async function fun_async() {
    let promise = Promise.resolve(42);
    let result = await promise;
    console.log(result);
    return result;
}
let promise2 = fun_async();
// ...
```

Wann wird der Code hinter dem await ausgeführt werden? Geben Sie die relevanten Mechanismen des JavaScript Interpreters bzw. der JavaScript Plattform an, die zu diesem Verhalten führen.

Klausurinfos KMPS, "Claßen Teil"

Erläutern Sie den Vorteil der JavaScript Programmierung mittels des async / await Konstrukts gegenüber einer Programmierung ausschließlich mit Promises, wenn es mehrere asynchrone Operationen gibt, die über Daten voneinander abhängen, d.h. jede Operation benötigt das Ergebnis der Vorgängeroperation.
Erläutern Sie dies anhand eines von ihnen erdachten, illustrativen JavaScript Codebeispiels.

Klausurinfos KMPS, "Claßen Teil"

Erläutern Sie das Konzept der Fehlerbehandlung beim JavaScript async / await Konstrukt. Wie ist die Relation zur Fehlerbehandlung bei der Programmierung (ausschließlich) mittels Promises und wie ist die Relation zur Fehlerbehandlung bei sequentiellen (nicht-nebenläufigen) JavaScript Code?
Erläutern Sie dies anhand eines von ihnen erdachten, illustrativen JavaScript Codebeispiels.

Klausurinfos KMPS, "Claßen Teil"

Erläutern Sie das Konzept der objekt-basierten / prototypischen Vererbung in JavaScript. Welche Operation ist die zentrale Operation bei dieser Art der Objekt-Orientierung, um das Konzept der Vererbung zu realisieren (es existieren mehrere Standardnamen für diese Operation, geben sie mindestens einen dieser Namen an)? Schreiben Sie JavaScript Code, um folgendes (ausschließlich) mittels dieser Art der Objekt-Orientierung zu realisieren:
Alle Shape2D haben eine x und eine y Koordinate und unterstützen die parameterlose Methode mirror(). Der Rumpf der Methode kann leer gelassen werden. Rectangle erbt von Shape2D und bietet zusätzlich die parameterlose Methode area(). Auch hier kann der Methodenrumpf leergelassen werden. Square erbt von rectangle und überschreibt die Methode area(). my_square1 ist ein Square, bei dem sie dann x auf den Wert 10 und y auf den Wert 20 setzen (über direkte zuweisungen an die Attribute).

Klausurinfos KMPS, "Claßen Teil"

Was versteht man unter prototypischer Vererbung (auch objekt-basierte Vererbung genannt) in JavaScript? Nennen und beschreiben Sie drei charakteristische Aspekte, die wesentlich in diesem Konzept sind. Dies können z.B. Aspekte sein, wie dieses Konzept „funktioniert“ oder z.B. zentrale Konstrukte der Programmiersprache, die in diesem Konzept relevant sind.

Klausurinfos KMPS, "Claßen Teil"

Was versteht man unter pseudo-klassischer Vererbung in JavaScript? Nennen und beschreiben Sie auch hier drei charakteristische Aspekte, die wesentlich in diesem Konzept sind.

Klausurinfos KMPS, "Claßen Teil"

Gegeben sei folgender JavaScript Code:

```
function Rectangle(length, width) {
this.length = length;
this.width = width;
}
Rectangle.prototype.getArea = function() {
return this.length * this.width;
};
Rectangle.prototype.toString = function() {
return "[Rectangle " + this.length + "x" + this.width + "]";
};
var rect = new Rectangle(5, 10);
function Square(size) {
Rectangle.call(this, size, size);
}
Square.prototype = new Rectangle();
Square.prototype.constructor = Square;
Square.prototype.toString = function() {
return "[Square " + this.length + "x" + this.width + "]";
};
var square = new Square(6);
```

Zeichnen Sie das Diagramm der Objekte und ihrer Attribut-Beziehungen zueinander, dass sich bei der JavaScript-internen Umsetzung dieses Programmcodes ergibt. Orientieren Sie sich an der in der Vorlesung verwendeten Diagramm-Notation.

Klausurinfos KMPS, "Claßen Teil"

Erläutern Sie, was man bei JavaScript unter dem Begriff des "Polyfilling" versteht.

(Den Code für das Polyfilling von Object.create() werde ich nicht erfragen. Auch nicht, wie man auf die Existenz testet und dann beim "fehlen" das Polyfilling macht. Aber als Beispiel für Polyfilling sollte man Object.create angeben können.)

Nicht alle Themen in Präsentation abgedeckt

Diese Präsentation umfasst nicht alle Themen des Semesters aus meinem Teil. Somit gibt es auch nicht zu allen Teilen einen „Beispiel-Fragenkatalog".

Die anderen Themen sind aber genauso prüfungsrelevant: Alle meine Themen aus dem Semester sind komplett prüfungsrelevant!

Die Idee der Prüfungsinfos ist ja, Ihnen konkret ein Verständnis zu geben, wonach ich in schriftlichen bzw. mündlichen KMPS Prüfungen frage, d.h. wie sie sich für die Prüfungen vorbereiten müssen / sollten.
Dies wird beispielhaft anhand ausgewählter Themen des Semesters konkret gemacht.
Dieselbe „Idee der Art zu fragen" gilt dann auch für alle meine anderen Themen des Semesters!