THM TECHNISCHE HOCHSCHULE MITTELHESSEN | CAMPUS GIESSEN | MNI Mathematik, Naturwissenschaften und Informatik

Prof. Dr. Dr. Dominikus Herzberg

Informationen zur Klausur

DIE KLAUSUR LIEGEN LASSEN, BIS SIE LOSLEGEN DÜRFEN!

Prüfungsdauer: 90 Minuten

Wertung: Insgesamt können 110 Punkte erreicht werden, davon 10 Bonuspunkte. Die pro Aufgabe bzw. Teilaufgabe erzielbaren Punkte sind angegeben. *Die erreichten Punkte werden Ihnen hälftig auf die kombinierte Prüfungsleistung angerechnet.*

Hilfsmittel: Es sind keine Hilfsmittel erlaubt. Es darf kein eigenes Papier verwendet werden.

Hinweise:

- Bitte verwenden Sie keinen Rotstift oder Bleistift.
- Ihre Lösungen tragen Sie bitte nur auf dem Lösungsblatt ein.
- Schreiben Sie Ihre Antworten bitte ausschließlich in die auf dem Lösungsblatt vorgesehenen Abschnitte.
- Sie müssen mit den vorhandenen leeren Blättern für Ihre Notizen und Lösungsskizzen auskommen. Sie bekommen kein weiteres Papier.
- Es werden nur leserliche Klausurlösungen bewertet!
- Die Angabe der Lösungen hat sich in Art und Form an den in der Veranstaltung verwendeten Konventionen zu orientieren.
- Sie geben am Ende der Klausur nur das Lösungsblatt ab. Die Aufgabenblätter sind für den Verbleib bei Ihnen bestimmt.
- Die Nutzung von `var` zur Deklaration von Variablen ist nicht erlaubt.

Prof. Dr. Dr. Dominikus Herzberg

Liebe Studierende,

zu Beginn ein paar formale Infos rund um die Klausur:

- Tragen Sie Ihren Namen und Ihre Matrikelnummer auf dem Lösungsblatt ein und unterschreiben Sie das Lösungsblatt.
- Während der Klausur werden keine Fragen zum Verständnis oder zur Klärung einer Klausuraufgabe beantwortet.
- Smartphones, Smartwatches, Kopfhörer und andere Hightech-Geräte dürfen aus verständlichen Gründen nicht verwendet werden.
- Sie verlassen den Platz nur mit unserer Erlaubnis. Melden Sie sich, wenn etwas ist.
- Das gilt auch, wenn Sie vorzeitig die Klausur abgeben wollen: Melden Sie sich. Wir sammeln das Lösungsblatt ein. Verlassen Sie anschließend bitte leise den Saal.
- 20 Minuten vor Ende der Klausur bitte keine vorzeitigen Abgaben mehr. Das macht zu viel Unruhe und erzeugt Stress.
- Legen Sie Ihren Studierendenausweis an den äußeren Rand ihres Tischs.
- In die Lösungsfelder bitte erst dann eine Lösung eintragen, wenn Sie sich sicher sind.

Viel Erfolg,

Dominikus Herzberg

Aufgabenteil zur PiS-Klausur, SoSe 2023

Prüfer: Prof. Dr. Dr. D. Herzberg, THM/MNI

⚠ Bitte nutzen Sie für Ihre Antworten ausschließlich die separat ausgehändigten Lösungsblätter! Sie geben am Ende der Klausur nur die Lösungsblätter ab! Ingesamt können 110 Punkte erreicht werden, davon sind 10 Bonuspunkte.

1 Objektvergleich (32 Punkte)

Gegeben sei eine Klasse `Point`, die einen Punkt mittels der privaten Variablen `x` und `y` vom Typ `long` als diskrete Koordinatenwerte in der Fläche verortet.

`equals` (12): Implementieren Sie die Methode `equals` nach dem in der Veranstaltung erlernten Schema.

`hashCode` (4): Implementieren Sie die Methode `hashCode` in einer Weise, so dass kein besonderes Verständnis zur Berechnung eines HashCodes erforderlich ist.

`toString` (4): Implementieren Sie die Methode `toString`, die Instanzen wie folgt repräsentiert:

```
jshell> new Point(1,2)
$27 ==> (1, 2)
```

`hashCodeMin` (12): Ergänzen Sie eine Methode `hashCodeMin`. Diese Methode legt zunächst eine immutable Liste aus der aktuellen Instanz und den drei durch Koordinatenachsen- und Ursprungsspiegelung erzeugten Punkte an. Für den Punkt `(1, 2)` wären das die Punkte `(-1, 2)`, `(1, -2)` und `(-1, -2)`; man geht also alle Vorzeichenvarianten durch. Aus der Liste wird ein Stream erzeugt, für all die vier Punkte werden die HashCodes berechnet, der Minimalwert herausgesucht (ein `Optional`) und der kleinste Integerwert zurückgegeben. Es wird nicht mehr als ein Semikolon benötigt.

2 Stapelbares (34 Punkte)

Sie sollen einen generischen Stapel als algebraischen Datentypen programmieren. Sie verwenden dafür ein versiegeltes (*sealed*) Interface namens `Stackable` und zwei Datenklassen `EmptyStack` und `StackWithElements`. Das Interface deklariert die folgenden Methoden:

- Die boolsche Methode `isEmpty` stellt fest, ob der Stapel leer ist.
- Die Methode `push` legt ein Element vom Datentyp `T` auf dem Stapel ab.
- Die Methode `top` gibt das oberste Datenelement vom Stapel zurück.
- Die Methode `pop` gibt den Stapel ohne das oberste Datenelement zurück.

`Stackable` (12): Deklarieren Sie das generische Interface `Stackable` und liefern Sie ausschließlich für die Methoden `isEmpty` und `push` default-Implementierungen; dafür ist es nötig, die nachstehenden Unteraufgaben gelesen zu haben. Im Rumpf des Interfaces kommen exakt 4 Semikolons ";“ vor. Vervollständigen Sie die Codezeilen. Achten Sie auf die Generizität und die Parameter und Rückgabetypen der Methoden; die Typen sollen so spezifisch wie möglich sein.

```
sealed interface Stackable
    default boolean isEmpty() {
        return
    }
    default
        return
    }


}
```

`StackWithElements` (6): Deklarieren Sie die generische Datenklasse `StackWithElements` und implementieren Sie zuerst die Methode `pop` und dann `top` jeweils als Einzeiler. Die Datenklasse verweist auf den vorigen Stapel (`previous`) und nimmt ein generisches Datenelement (`element`) auf. Es werden exakt 2 Semikolons benötigt.

```
record StackWithElements


}
```

`EmptyStack` (6): Deklarieren Sie die generische Datenklasse `EmptyStack` und implementieren Sie die Methoden `pop` und `top` jeweils als Einzeiler mit der Ausnahme `UnsupportedOperationException`. Es werden exakt 2 Semikolons benötigt.

```
record EmptyStack


}
```

`of`-Methode (6): Ergänzen Sie das Interface um eine statische, generische Methode namens `of`, mit der sich ein Stapel komfortabel anlegen lässt, wie die folgende Interaktion in der JShell zeigt:

```
jshell> Stackable.<Integer>of(1,2)
$1 ==> StackWithElements[previous=StackWithElements[previous=EmptyStack[], element=1], element=2]
```

Verwenden Sie eine `for`-Schleife, um die übergebenen Elemente auf den Stapel zu legen. Es werden nicht mehr als 3 Semikolons benötigt.

```


    return stack;
}
```

`toString` (4): Die defaultmäßige `toString`-Methode erzeugt eine zu lange Zeichenkette, siehe oben bei der `of`-Methode. Ergänzen Sie die beiden Datenklassen um je eine passende Implementierung der `toString`-Methode entsprechend der folgenden JShell-Interaktion. Die Methoden konzipieren Sie bitte als Einzeiler, sie haben jeweils 1 Semikolon und kommen für benötigte Zugriffe ausschließlichen mit den Interface-Methoden aus.

```
jshell> Stackable.<Integer>of()
    $106 ==> [ ]

jshell> Stackable.<Integer>of(1,2,3)
$107 ==> [ ] <- 1 <- 2 <- 3
```

Implementierung für `StackWithElements`:

```
1   public String toString() {
```

Implementierung für `EmptyStack`:

```
1   public String toString() {
```

3 Zahlensuche (20 Punkte)

Gesucht ist eine positive Ganzzahl mit der folgenden Eigenschaft: Die Zahl ist durch 2, 3 und 5 teilbar. Das gilt ebenso für ihre Quersumme. Welche ist die kleinste Zahl mit dieser Eigenschaft? — So lautet die Aufgabe, wie sie auch in der Übung behandelt wurde. Jedoch soll die Implementierung so umgesetzt werden, dass die Aufgabe grundsätzlich auch für eine beliebige Anzahl anderer Teiler gelöst werden kann. Wir gehen davon aus, dass die Methode für die Quersumme `long quer(long n)` gegeben ist und dass mit der JShell gearbeitet wird.

`isDivisible` (8): Schreiben Sie eine Methode namens `isDivisible`, die eine beliebige Anzahl von möglichen Teilern (`divisors`) als `Integer` entgegennimmt und ein Prädikat zurückliefert, das eine `Long`-Zahl erwartet; das Prädikat testet, ob die Zahl von allen Teilern restlos geteilt werden kann. Die Implementierung der Methode besteht aus einem `return` mit Lambda-Ausdruck, es kommt nur ein Semikolon zum Einsatz. Die Teiler werden gestreamt, und per `allMatch` wird die übergebene Zahl jeweils auf Teilbarkeit geprüft.

```
1
2       return
3   }
```

`isValid` (2): Deklarieren Sie eine Variable namens `isValid`, die mit dem durch `isDivisible(2,3,5)` erzeugten Rückgabewert initialisiert wird. (Der Gebrauch von `var` ist untersagt.)

```
1
```

Berechnung (8): Setzen Sie einen Ausdruck(!) für die JShell auf (d.h. kein Semikolon), der die gesuchte Zahl aus der Aufgabe berechnet. Dazu sind `Long`-Zahlen bis zum maximal möglichen

`Long`-Wert zu streamen und die Lösung (kleinste Zahl) ist unter Bezug auf `isValid` zu ermitteln.

```
1
2
3
```

Ergebnis (2): In dem speziellen Fall mit den Teilern 2, 3 und 5 lautet die Lösung 39990. Wie wird die Lösung in der JShell als Ergebnis der Berechnung angezeigt? Hinweis: Sie wissen im generellen Fall nicht, ob eine Lösung existiert.

```
1  $31 ==>
```

4 Fragen (8 × 3 = 24 Punkte)

Hinweis: Pro Frage kann mehr als eine Antwort korrekt sein. Die Fragen sind so formuliert, dass in der Regel nicht ableitbar ist, ob eine oder mehr Antworten zutreffen.

Frage A: Welche der folgenden Aussagen beschreibt den Unterschied zwischen `List` und `ArrayList` in Java korrekt?

1. `List` und `ArrayList` sind beide konkrete Klassen in Java.
2. `ArrayList` ist eine Schnittstelle, während `List` eine konkrete Implementierung dieser Schnittstelle ist.
3. `List` ist eine Schnittstelle, `ArrayList` ist eine konkrete Implementierung dieser Schnittstelle.
4. `ArrayList` und `List` sind beides Schnittstellen, aber `ArrayList` enthält mehr Methoden als `List`.
5. `List` ist eine abstrakte Klasse und `ArrayList` ist eine konkrete Klasse.

Frage B: Was passiert, wenn Sie versuchen, ein Element zu einer Liste hinzuzufügen, die mit `List.of` erstellt wurde?

1. Das Element wird erfolgreich hinzugefügt.
2. Es wird eine `UnsupportedOperationException` ausgelöst.
3. Es wird eine `NullPointerException` ausgelöst.
4. Es wird eine `IndexOutOfBoundsException` ausgelöst.
5. Die Liste wird zu einer modifizierbaren Liste konvertiert.

Frage C: Welche alternativen Möglichkeiten der nebenläufigen und parallelen Programmierung gibt es in Java, statt direkt mit Threads zu arbeiten?

1. Relative Programmierung
2. Logische Programmierung
3. Ergebnisbasierte Programmierung
4. Strombasierte Programmierung
5. Wertbasierte Programmierung

Frage D: Welche Syntax für einen Lambda-Ausdruck ist gültig?

1. `(int x, int y) -> { return x + y; }`
2. `(String s) -> s.length()`
3. `(int x, int y) -> { x * y; }`
4. `() -> return "Hello World!"`
5. `x -> x * x`

Frage E: Eine Klasse `E` erweitere die Klasse `X`. Was syntaktisch möglich ist, ist semantisch zu überprüfen. Welche Aussage diesbezüglich gültig?

1. Die erweiterte Klasse muss isomorph zur Oberklasse sein
2. Die Klassenhierarchie sollte nicht tiefer sein als sie breit ist
3. Die Klassen müssen das Liskov'sche Substitutionsprinzip erfüllen
4. Klassennamen sind wichtig, da der Name die Syntax bestimmt
5. Die erweiterte Klasse muss den *isa*-Test bestehen

Frage F: Was ist ein `Set` in Java?

1. Eine Datenstruktur, die Elemente in einer geordneten Folge speichert.
2. Eine Datenstruktur, die keine Duplikate zulässt.
3. Eine Schnittstelle zur Verwaltung von Schlüssel-Wert-Paaren.
4. Eine Datenstruktur, die Elemente nach dem Prinzip „First In, First Out“ (FIFO) verwaltet.
5. Eine Art von Kontrollstruktur.

Frage G: Welche der folgenden Aussagen über `Optional` in Java ist richtig?

1. Mit `Optional` können Sie `NullPointerExceptions` vermeiden.
2. `Optional.empty()` gibt ein `Optional` zurück, das einen `null`-Wert enthält.
3. `Optional` ist immer mit einem nicht-`null`-Wert befüllt.
4. `Optional` kann nur mit Strings verwendet werden.
5. `Optional` ist eine Containerklasse für den Fall, dass ein Wert `null` sein könnte.

Frage H: Sie haben das Spiel Mühle (*mills*) ohne User Interface in einer Klasse `Game` umgesetzt und haben dazu ein Interface `IGame` deklariert, das alle notwendigen Operationen anbietet und von der Klasse implementiert wird. Sie wollen nun eine Variable namens `mills` deklarieren und initialisieren und darauf achten, dass ausschließlich Methoden des Interfaces aufgerufen werden können. Welche Anweisung ist dafür geeignet?

1. `Game mills = new Game();`
2. `IGame mills = new IGame();`
3. `Game mills = new IGame();`
4. `IGame mills = new Game();`
5. keine Antwort ist zutreffend

(*Notizblatt*)

(*Notizblatt*)

(*Notizblatt*)

Lösungsblatt zur PiS-Klausur

Sie geben am Ende der Klausur nur das Lösungsblatt ab!

Nachname, Vorname	**Matrikelnummer**	**Note**
Unterschrift 21.07.2023		

1. Objektvergleich

`equals:`

`hashCode:`

`hashCodeMin:`

`toString:`

2. Stapelbares

`Stackable`

```
sealed interface Stackable
    default boolean isEmpty() {
        return
    }
    default
        return
    }


}
```

StackWithElements

```
record StackWithElements



}
```

EmptyStack

```
record EmptyStack



}
```

of-Methode

```



    return stack;
}
```

toString StackWithElements

```
public String toString() {
```

EmptyStack

```
public String toString() {
```

3. Zahlensuche

isDivisible

```

    return
}
```

isValid

```

```

Berechnung

```


```

Ergebnis

```
$31 ==>
```

4. Fragen

	1.	2.	3.	4.	5.	*Punkte*		1.	2.	3.	4.	5.	*Punkte*
A	☐	☐	☐	☐	☐		E	☐	☐	☐	☐	☐	
B	☐	☐	☐	☐	☐		F	☐	☐	☐	☐	☐	
C	☐	☐	☐	☐	☐		G	☐	☐	☐	☐	☐	
D	☐	☐	☐	☐	☐		H	☐	☐	☐	☐	☐	

Sollten Sie eine Lösung revidieren wollen, streichen Sie die betreffende Reihe durch und notieren Sie hier neben der Tabelle Ihre korrigierte Lösung im Stil von Z13: die hypothetische Frage Z, Antwort 1 und 3.