hochschule mannheim

Test 1 (Studienleistung)
PP – Parallele Programmierung
Bachelor Informatik (IB)
Wintersemester 2019/2020 [28.10.2019]

FAKULTÄT FÜR INFORMATIK

Matrikel-Nr.: ..... ☐ CSB / ☐ IB / ☐ IMB / ☐ MEB / ☐ UIB / ☐ IM

Name: .....

Vorname: .....

Unterschrift: .....

Hinweise:

(1) Dieser Test hat 6 Seiten. Sie haben 30 Minuten Zeit zur Beantwortung.

(2) Schreiben Sie Ihren *Namen und Ihre Matrikelnummer zu Beginn* auf das Deckblatt und kreuzen Sie Ihren Studiengang an. Überprüfen Sie, ob der Test vollständig ist.

(3) Schreiben Sie die *Lösung* zu einer Aufgabe *auf das vorgesehene Blatt*. Sollte der Platz nicht reichen, benutzen Sie bitte den Reserveplatz am Ende. Notieren Sie dabei am vorgesehenen Platz, dass es beim Reserveplatz weitergeht, und kennzeichnen Sie auch unbedingt dort, zu welcher Aufgabe die Antwort gehört.

(4) Zerlegen Sie die Heftung *nicht in einzelne Blätter*. Lose Blätter werden nicht gewertet.

(5) Es sind keine Hilfsmittel erlaubt.

(6) Ein Täuschungsversuch führt zur Bewertung mit 0 Punkten. Dazu zählt Kommunikation mit anderen Personen außer Prüfer/in/Aufsichtspersonen und jede Benutzung von unerlaubten Hilfsmitteln wie mobilen Endgeräten (z. B. Smartphones).

(7) Schreiben Sie mit dokumentenechten Stiften (Kugelschreiber). Mit Bleistiften, Füllern o. ä. erstellte Lösungen sind ungültig! Tipp-Ex und rotschreibende Stifte sind ebenfalls verboten. Schreiben Sie bitte leserlich.

Aufgabe	1	2	3	4	Summe
Punkte	6	12	7	5	30
Erreicht					

Viel Erfolg!

Aufgabe 1

Aufgabe 1: von 6 P

Die folgenden Aufgaben beziehen sich auf dieses Java-Programm:

```
public class Main extends Thread {
    public void interact() {
        System.out.println(Thread.currentThread().getName());
    }

    @Override
    public void run() {
        while (true); // Endlosschleife
    }

    public static void main(String[] args) throws InterruptedException {
        var t = new Main();
        t.setName("Thread");
        t.setDaemon(true);
        t.start();
        var actor = new Thread(() -> t.interact(), "Actor");
        actor.start();
    }
}
```

(a) Welche Ausgabe hat das obige Java-Programm? 2 P

(b) Wieviele Threads laufen in dem Programm ab und wie heißen sie? 2 P

(c) Terminiert das Programm? Begründen Sie Ihre Antwort. 2 P

Aufgabe 2

Aufgabe 2: von 12 P

(a) Erläutern Sie kurz, was man unter einer Memory-Barrier versteht und wie man in Java eine Memory-Barrier anfordert. 3 P

(b) Warum terminiert das folgende Programm nicht nach ca. 1000ms? Was muss am Programm bezüglich Nebenläufigkeit geändert werden, damit es nach ca. 1000ms terminiert? 6 P

```
class B extends Thread {
    private boolean sollAnhalten = false;
    private Thread self = Thread.currentThread();
    private synchronized boolean sollAnhalten() {
        return this.sollAnhalten;
    }
    private synchronized void anhalten() {
        this.sollAnhalten = true;
    }
    @Override
    public void run() {
        while (!sollAnhalten()) {
            // arbeiten
            try {
                Thread.sleep(100000);
            } catch (final InterruptedException e) {
            }
        }
        // enden
    }
    public static void main(final String[] args) throws InterruptedException {
        final B t = new B();
        t.start();
        Thread.sleep(1000);
        t.anhalten();
    }
}
```

(c) Erläutern Sie, ob die folgende Java-Klasse threadsafe ist, und begründen Sie dies kurz. 3 P

```
final class Super {
    static class Sub {
    }

    private static Sub object;
    private static int number;

    public static int getInstances() {
        return number;
    }

    public static Sub make() {
        if (object == null) {
            object = new Sub();
            number++;
        }
        return object;
    }
}
```

Aufgabe 3

Aufgabe 3: von 7 P

(a) Welche der folgenden Methoden sind threadsafe? Ergänzen Sie den Programmcode so, dass die Klasse threadsafe wird und Stellen gegenseitigen Ausschlusses so kurz wie möglich sind. 7 P

```
final class Tester {
    private int x = 0;
    private int y = 0;
    public void step1(final int n) {
        this.y += n * n;
        this.x++;
    }
    public int step2(final int n) {
        return n * n;
    }
    public int step3(final int n) {
        this.y += n;
        return n*2;
    }
    public int getX() {
        return this.x;
        }
```

```
        public int getY() {
        return this.y;
    }
}
```

Aufgabe 4 | **Aufgabe 4: von 5 P**

Die folgende Aufgabe bezieht sich auf den begrenzten Ringspeicher: Falls der Speicher leer ist, müssen Threads aufgehalten werden, die die `take()`-Methode aufrufen. Falls der Speicher voll ist, müssen Threads aufgehalten werden, die die `put()`-Methode aufrufen.

Korrigieren Sie etwaige Fehler direkt im folgenden Quelltext.

```
public class BoundedBuffer<T> {
    final Object[] items = new Object[8];
    int putptr, takeptr, count;
    public void put(final T x) throws InterruptedException {
        if (this.count == this.items.length) {
            1111111111();
        }
        this.items[this.putptr] = x;
        if (++this.putptr == this.items.length) {
            this.putptr = 0;
        }
        ++this.count;
        2222222222();
    }
    public T take() throws InterruptedException {
        if (this.count == 0) {
            3333333333();
        }
        final T x = (T) this.items[this.takeptr];
        if (++this.takeptr == this.items.length) {
            this.takeptr = 0;
        }
        --this.count;
        4444444444();
        return x;
    }
}
```

Welche Methoden müssen anstelle der folgenden Platzhalter eingesetzt werden?

`1111111111()` =

`2222222222()` =

`3333333333()` =

`4444444444()` =

Reserveplatz