

Blatt 5

Betriebssysteme

Luis Staudt

Aufgabe a) - PipedInputStream und PipedOutputStream

Programm-version	Läuft das Programm fehlerfrei zu Ende, wird eine Exception geworfen oder verklemmt das Programm?	Wenn eine Exception geworfen wird, welche und mit welchem Text?	Wie ist das Verhalten des Programms jeweils zu erklären?
I)	Wird eine Exception geworfen	IOException: "Write end dead"	Thread B liest alle Bytes, aber ohne close() beendet Thread A, wodurch die Pipe ungültig wird
II)	Läuft fehlerfrei zu Ende	Keine	Thread B liest bis -1 (EOF nach close()), sauberes Beenden
III)	Läuft fehlerfrei zu Ende	Keine	Pipe-Buffer (1024 Bytes) ist groß genug
IV)	Programm verklemmt	Keine	Thread A blockiert bei write(), da Pipe-Buffer voll ist
V)	Wird eine Exception geworfen	IOException: "Write end dead"	Thread B liest alle 2048 Bytes, Exception erst nach Beenden von Thread A
VI)	Wird eine Exception geworfen	IOException: "Read end dead"	Thread B liest nur 1 Byte, Thread A kann Rest nicht schreiben

Aufgabe b) - pop() und InterruptedException

Antwort: Die Schleife wird nie verlassen, weil:

1. pop() wirft InterruptedException, wenn der Thread interrupted wird
2. Der leere catch-Block fängt die Exception, tut aber nichts
3. InterruptedException löscht automatisch das Interrupt-Flag
4. Daher bleibt isInterruptedException() = false
5. Die Schleifenbedingung !isInterruptedException() bleibt wahr ⇒ Endlosschleife

Lösung: Im catch-Block entweder:

- `Thread.currentThread().interrupt()` aufrufen, um das Flag wieder zu setzen
- oder die Schleife mit `break` verlassen

Aufgabe c) - Verarbeitungskette

Implementierungshinweise:

- Die Queue-Klasse verwendet `synchronized`-Methoden mit `wait/notify` für Thread-Synchronisation
- Die zufällige Reihenfolge in der `OutputQueue` entsteht durch:
 - Unterschiedliche Sleep-Zeiten der Einspeiser (0-2 Sekunden)
 - Nicht-deterministische Thread-Scheduling des Betriebssystems
- Nach dem Interrupt werden die `InputQueues` vollständig geleert
- Das Programm beendet sich korrekt durch `join()` auf alle Threads