Übersicht Nebenläufiger Kontrollkonzepte in verschiedenen Programmiersprachen

Konzept/Sprache	Java	Golang	C#	Erlang	Kotlin	Haskell
Atomics	java.util.concurrent.atomic AtomicCounter	sync.atomic	System.Threading.Interlocked	:atomics atomics_ref()	AtomicReference java.util.concurrent.atomic	Data.Atomics
Mutex	SynchronizedMethodsCounter SynchronizedObjectCounter SynchronizedThisCounter	sync.Mutex	System.Threading: Monitor (Enter, Wait, Pulse, Exit), Mutex, Semaphore, CountdownEvent	Mit Actor Concurancy	Lock.withLock(action) ReentrantReadWriteLock @Synchronized	Control.Concurrent.MVar: takeMVar putMVar MVar Chan bounded_Chan
Software transactional memory	TransactionalCounterMultiverse TransactionalCounterNarayana TransactionalCounterScala	https://github.com/anacrolix/stm	Shielded STMNet Sasa.TM	https://github.com/ian-plosker/stm_erl	Über Java Libraries	TransactionalCounter TChans bounded_TChans
Concurrent Queues, Channels und Messages	java.util.concurrent.BlockingQueue LoomPRNG LoomContinuationPRNG SingleThreadScheduledExecutorDemo	GoroutinesSimple Goroutines MultibleGoroutines MultibleGoroutinesBlocking GoroutinesRandom async	System.Collections.Concurrent.ConcurrentQueue System.Threading.Channels. Channel<T> AsyncLargeFileDownload TasksRandom CoroutinesRandom	!-Operator count/actors prng/actors	kotlinx.coroutines.channels. Channel or https://raw.githubusercontent.com/Kotlin/coroutines-examples/master/examples/channel/channel.kt Coroutines MultibleCoroutines MultibleCoroutinesBlocking CoroutinesRandom CoroutinesRandomYield	Control.Concurrent.Chan or Control.Concurrent.STM.TChan Chan TChans bounded_chan bounded_TChans

🟩	Direkt von der Sprache unterstützt	🟩	Bibliothek in gleicher Sprache	🟦	Zukünftig direkt von der Sprache unterstützt
🟨	Bibliothek in anderer Sprache	⬜	Möglich durch höhere Nebenläufige Programmiermodelle	🟧	Nicht unterstützt

Text: Referenzierter Programmcode

Übersicht Nebenläufiger Programmiermodelle in verschiedenen Programmiersprachen

Konzept/Sprache	Java	Golang	C#	Erlang	Kotlin	Haskell
Thread Pools	FixedThreadPoolDemo CachedThreadPoolDemo ForkJoinPoolDemo SingleThreadExecutorDemo SingleThreadScheduledExecutor	Ja (mit Gorutines oder Library) https://github.com/shettyh/threadpool	System.Threading.ThreadPool	https://github.com/devinus/poolboy https://github.com/g-andrade/taskforce	Gleich wie in Java	Control.ThreadPool Control.Concurrent.Async.Pool
Futures/Tasks	AsyncHandlers BlockingFuture	Ja (über Gorutines und Channels) https://appliedgo.net/futures/	System.Threading.Tasks	https://github.com/gleber/erlfu	Gleich wie in Java	Control.Concurrent.Future
Futures with completion logic/Promises	CompletableFuture CompletableFutureBranchless ListenableFutureDemo	Ja (über Gorutines und Channels oder async)	Ja (Task oder Async/Await)	https://github.com/gleber/erlfu	Gleich wie in Java	Control.Concurrent.Future
Async/Await		async	AsyncLargeFileDownload	https://github.com/redink/task (Native mit Elixir)		Control.Concurrent.Async
Reactive	ReactiveX (RxJava) ReactiveStreams ReactivePRNG	ReactiveX (RxGo)	ReactiveX (Rx.NET)	Functional Reactive Programming	ReactiveX (RxKotlin)	Functional Reactive Programming
Continuation	LoomContinuationPRNG LoomSingleThreadedContinuationPRNG	Ja (https://bbengfort.github.io/2016/12/yielding-functions-for-iteration-golang/)	CoroutinesRandom	Nur in Elixir mit Continuation Passing Style	CoroutinesRandomYield	Continuation Passing Style
Coroutines	Mit Continuation und Fibers manual machbar	Coroutines CoroutinesSimple MultibleCoroutines MultibleCoroutinesWaitgroup CoroutinesRandom	Mit Continuation und Tasks manuell machbar – Eingebaut in der Unity C# Game Engine	Mit Erlang processes	Coroutines MultibleCoroutines MultibleCoroutinesBlocking CoroutinesRandom	Control.Monad.Coroutine
Fibers	LoomFiberCounter LoomPRNG	Siehe Coroutines Beispiele	Nein aber Async/Await	Light-weight process	Mit Project Loom	Threads sind Fibers
Actor Concurrency	AkkaPRNG AkkaHTTP_ConnectionLevel AkkaHTTP_HostLevel	Ja (Native und über Akka.NET kompatibles protoactor-go) https://github.com/AsynkronIT/protoactor-go	Akka.NET	count/actors prng/actors	https://akka.io/	Control.Concurrent.CHP Control.Concurrent.Actor

🟩	Direkt von der Sprache unterstützt	▢	Bibliothek in gleicher Sprache	🟦	Zukünftig direkt von der Sprache unterstützt
🟨	Bibliothek in anderer Sprache	⬜	Möglich durch höhere Nebenläufige Programmiermodelle	🟧	Nicht unterstützt

Text: Referenzierter Programmcode

Übersicht der Begriffe von nebenläufigen Programmiermodellen in verschiedenen Programmiersprachen

Konzept/Sprache	Java	Golang	C#	Erlang	Kotlin	Haskell
Thread Pools	**Thread Pools** https://www.baeldung.com/thread-pool-java-and-guava https://docs.oracle.com/javase/tutorial/essential/concurrency/pools.html https://www.geeksforgeeks.org/thread-pools-java/	**Goroutines** https://gobyexample.com/goroutines https://tour.golang.org/concurrency/1 https://medium.com/technofunnel/understanding-golang-and-goroutines-72ac3c9a014d	**Thread Pools** https://www.dotnetperls.com/threadpool https://docs.microsoft.com/en-us/dotnet/api/system.threading.threadpool?view=net-5.0		**Thread Pools** Siehe Java	**ThreadPool Pool** https://hackage.haskell.org/package/Control-Engine-1.1.0.1/docs/Control-ThreadPool.html http://hackage.haskell.org/package/async-pool-0.9.1/docs/Control-Concurrent-Async-Pool.html
Futures/Tasks	**Future** https://www.baeldung.com/java-future https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Future.html	**Goroutines** https://appliedgo.net/futures/	**Tasks** https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task?view=net-5.0		**Future** https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.native.concurrent/-future/	
Futures with completion logic/Promises	**CompletableFuture** https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/CompletableFuture.html https://stackoverflow.com/questions/14541975/whats-the-difference-between-a-future-and-a-promise	**Goroutines** https://levelup.gitconnected.com/use-go-channels-as-promises-and-async-await-ee62d93078ec	**TaskCompletionSource** https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskcompletionsource-1?view=net-5.0		**CompletableFuture** Siehe Java	
Async/Await		**Asynchronous programming** https://medium.com/@gauravsingharoy/asynchronous-programming-with-go-546b96cd50c1	**Asynchronous programming** https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/			
Reactive				**Functional Reactive Programming** https://www.infoq.com/presentations/frp-erlang-grisp/		**Functional Reactive Programming** http://wiki.haskell.org/Functional_Reactive_Programming
Continuation	**Continuation** https://openjdk.java.net/projects/loom/ https://cr.openjdk.java.net/~rpressler/loom/Loom-Proposal.html	**Continuation-Passing style** Yielding Functions https://medium.com/@meeusdylan/continuation-passing-style-in-go-fa06a0ca00a2 https://bbengfort.github.io/2016/12/yielding-functions-for-iteration-golang/	**yield contextual keyword** https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/yield		**Continuation** suspending functions https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.coroutines/-continuation/ https://kotlinlang.org/docs/composing-suspending-functions.html	**Continuation** https://wiki.haskell.org/Continuation
Coroutines	**Coroutines** https://blog.frankel.ch/project-loom-reactive-coroutines/	**Goroutines** https://tour.golang.org/concurrency/1		**Coroutines** https://en.wikipedia.org/wiki/Coroutine	**Coroutines** https://kotlinlang.org/docs/coroutines-overview.html https://github.com/Kotlin/coroutines-examples	

Fibers	**Fibers** Lightweight threads https://openjdk.java.net/projects/loom/ https://cr.openjdk.java.net/~rpressler/loom/Loom-Proposal.html	**Goroutines** https://yourbasic.org/golang/goroutines-explained/		**Light-weight process** http://erlang.org/doc/efficiency_guide/processes.html https://en.wikipedia.org/wiki/Erlang_(programming_language)#Concurrency_and_distribution_orientation	**Fibers** Lightweight threads Siehe Java	**Threads** https://hackage.haskell.org/package/base-4.15.0.0/docs/Control-Concurrent.html https://hackage.haskell.org/package/base-4.15.0.0/docs/Control-Concurrent.html#v:forkIO
Actor Concurrency		**Actors** https://slcjordan.github.io/posts/actors/		**Actors** https://www.infoworld.com/article/2077999/understanding-actor-concurrency--part-1--actors-in-erlang.html		**Actors** https://hackage.haskell.org/package/hactors-0.0.3.1/docs/Control-Concurrent-Actor.html

Übersicht der Begriffe nebenläufiger Kontrollkonzepte in verschiedenen Programmiersprachen

Konzept/Sprache	Java	Golang	C#	Erlang	Kotlin	Haskell
Atomics	**Atomic Variables** Atomic Operations https://www.baeldung.com/java-atomic-variables	**Atomic Memory Primitives** https://golang.org/pkg/sync/atomic/	**Interlocked** https://www.dotnetperls.com/interlocked	**Atomic Functions** https://erlang.org/doc/man/atomics.html	**Atomic References** https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.native.concurrent/-atomic-reference/ https://kotlinlang.org/docs/mobile/concurrent-mutability.html	**Atomic Memory** https://hackage.haskell.org/package/atomic-primops-0.8.4/docs/Data-Atomics.html
Monitors (Mutual exclusion & Synchronisation)	**Monitors** Locks Guarded Blocks https://stackoverflow.com/questions/3362303/whats-a-monitor-in-java https://winterbe.com/posts/2015/04/30/java8-concurrency-tutorial-synchronized-locks-examples/ https://www.baeldung.com/java-concurrent-locks https://docs.oracle.com/javase/tutorial/essential/concurrency/guardmeth.html	**Monitors** Mutexes https://medium.com/dm03514-tech-blog/golang-monitors-and-mutexes-a-light-survey-84f04f9b7c09	**Monitor** Locks https://docs.microsoft.com/en-us/dotnet/api/system.threading.monitor?view=net-5.0 https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/lock-statement		**Mutual exclusion** Locks https://kotlin.github.io/kotlinx.coroutines/kotlinx-coroutines-core/kotlinx.coroutines.sync/-mutex/index.html https://blog.egorand.me/concurrency-primitives-in-kotlin/	**Synchronising variables** MVar (gesprochen: "em-var") https://hackage.haskell.org/package/base-4.3.1.0/docs/Control-Concurrent-MVar.html
Software transactional memory						**Atomic transaction** https://en.wikipedia.org/wiki/Concurrent_Haskell
Concurrent Queues, Channels und Messages	**Blocking Queue** https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/BlockingQueue.html	**Channels** https://tour.golang.org/concurrency/2	**Channels** Concurrent Queue https://devblogs.microsoft.com/dotnet/an-introduction-to-system-threading-channels/ https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentqueue-1?view=net-5.0	**!-Operator** https://erlang.org/doc/reference_manual/expressions.html	**Channels** https://kotlinlang.org/docs/channels.html	**Chan/TChan** https://hackage.haskell.org/package/base-4.15.0.0/docs/Control-Concurrent-Chan.html https://hackage.haskell.org/package/stm-2.5.0.1/docs/Control-Concurrent-STM-TChan.html