IMPLEMENTAREA CONCURENTEI IN LIMBAJE DE PROGRAMARE 2024-2025

Curs:

Ioana Leustean

ioana.leustean@unibuc.ro

Laborator:

Bogdan Macovei

bogdan.macovei@unibuc.ro

- Resurse: moodle, MSTeams

- Sistem de notare:
  - **Proiect**
  - **Examen in sesiune**
    - Probleme asemanatoare celor din curs si laborator
    - Se poate impune un limbaj pentru rezolvarea unei probleme

- 1 punct din oficiu

- Puncte suplimentare pentru activitatea din timpul semestrului (maxim 2 puncte)

"Concurrency is an interesting word because it means different things to different people in our field. In addition to “concurrency,” you may have heard the words, “asynchronous,” “parallel,” or “threaded” bandied about. Some people take these words to mean the same thing, and other people very specifically delineate between each of those words."

Katherine Cox-Buday, Concurrency in Go, O'Reilly, 2017

"In the computer world, when we talk about **concurrency,** we refer to a series of independent and unrelated tasks that run simultaneously on a computer. This simultaneity can be real if the computer has more than one processor or a multi-core processor, or it can be apparent if the computer has only one core processor.

Another concept related to concurrency is **parallelism**. There are different definitions and relations with the concurrency concept. Some authors talk about concurrency when you execute your application with multiple threads in a single-core processor. With this, you can see when your program execution is apparent. They talk about parallelism when you execute your application with multiple threads in a multi-core processor or in a computer with more than one processor […]"

Gonzalez, Javier Fernandez. Java 9 Concurrency Cookbook - Second Edition Packt Publishing. 2017

.

Dining Philosophers Problem

"In ancient times, a wealthy philanthropist endowed a College to accommodate five eminent philosophers. Each philosopher had a room in which he could engage in his professional activity of thinking; there was also a common dining room, furnished with a circular table, surrounded by five chairs, each labelled by the name of the philosopher who was to sit in it. The names of the philosophers were PHIL0, PHIL1, PHIL2, PHIL3, PHIL4, and they were disposed in this order anticlockwise around the table. To the left of each philosopher there was laid a golden fork, and in the center stood a large bowl of spaghetti, which was constantly replenished.

A philosopher was expected to spend most of his time thinking; but when he felt hungry, he went to the dining room, sat down in his own chair, picked up his own fork on his left, and plunged it into the spaghetti. But such is the tangled nature of spaghetti that a second fork is required to carry it to the mouth. The philosopher therefore had also to pick up the fork on his right. When we was finished he would put down both his forks, get up from his chair, and continue thinking. Of course, a fork can be used by only one philosopher at a time. If the other philosopher wants it, he just has to wait until the fork is available again."

C.A.R. Hoare, Communicating Sequential Processes, 2004
(formulate initial de E. Dijkstra

Limbaje concurente

- Scopul programarii concurente este de a descrie programe cu multe interactiuni:
  intre utilizatorii aceleiasi interfete, intre subsistemele aceluiasi sistem, intre sisteme diferite.

- Un limbaj de programare concurent este un limbaj care ofera constructii sintactice explicite prin care se pot descrie aceste interactiuni.

- Un program concurrent este un program scris intr-un limbaj de programare concurrent.

- Comportamentul unui program concurent este independent de sistemul de operare.

Metode de implementare a concurentei

- Thread-uri si memorie partajata
- Software Transactional Memory
- Modelul cu Actori

In acest curs vom analiza toate aceste metode !

Limbajele studiate la curs: Java, Haskell, Go, Erlang

➢Thread-uri

Un **thread** este o secventa de instructiuni in interiorul unei aplicatii/unui proces.

O aplicatie poate contine mai multe thread-uri.

Threadurile pot fi executate in parallel, dar in interiorul fiecarui thread operatiile sunt secventiale.

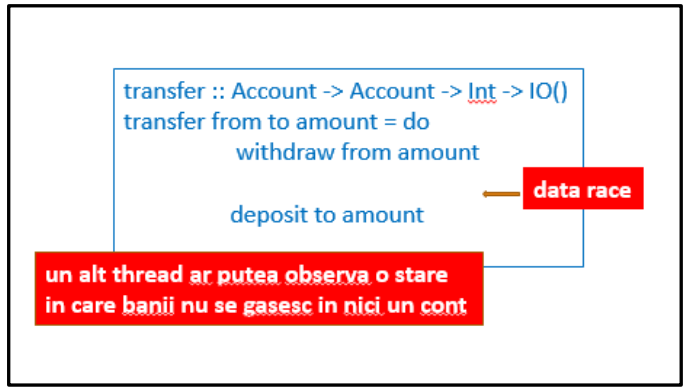
Thread-urile isi impart resursele si comunica intre ele.

Comunicarea pote fi sincrona sau asincrona.

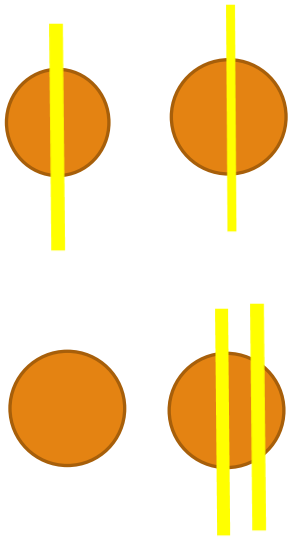
Probleme:
Deadlock
Date Race
Starvation
Livelock

Metode pentru controlul accesului la resurse:
sincronizare
Semafoare
Monitore

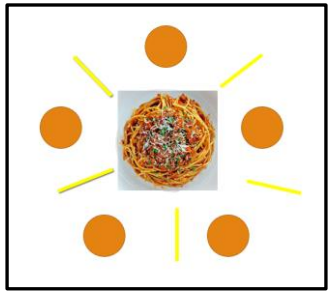
Data Race = apare cand mai multe threaduri acceseaza concurrent aceeasi locatie de memorie fara mecanisme de sincronizare adecvate; codul poate functiona corect in unele situatii dar eronat in altele.



Deadlock = apare cand doua sau mai multe threaduri se asteapta unul pe celalalt pentru a elibera accesul la resurse



Dining Philosophers

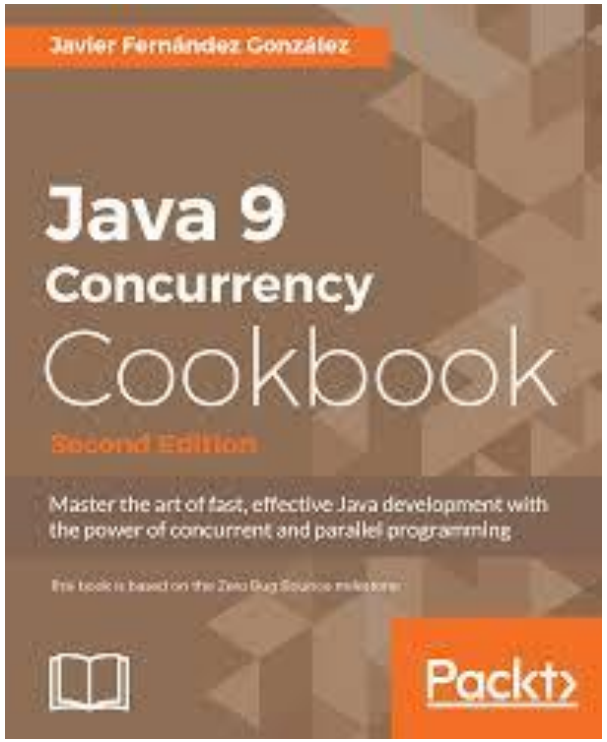


Starvation = apare cand unui thread i se refuza continuu accesul la resurse

Livelock = apare cand doua sau mai multe threaduri actioneaza, blocandu-se unul pe celalalt

Thread-uri si memorie partajata

"When you work with a computer, you can do several things at once. You can listen to music while you edit a document in a word processor and read your e-mails. This can be done because your operating system allows the concurrency of tasks. Concurrent programming is about the elements and mechanisms a platform offers to have multiple tasks or programs running at once and communicating with each other, to exchange data or to synchronize with each other. **Java is a concurrent platform**, and it offers a lot of classes to execute concurrent tasks inside a Java program. With each version, Java increases the functionalities offered to programmers to facilitate the development of concurrent programs."



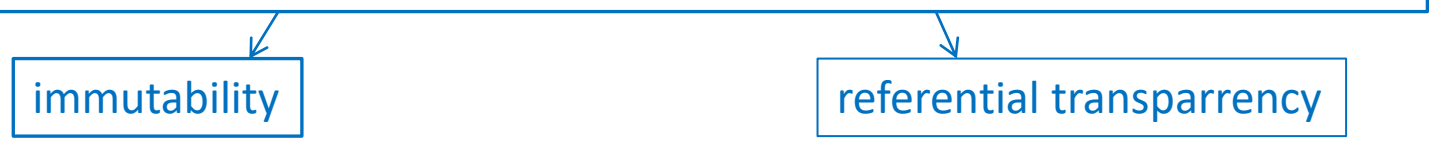
Gonzalez, Javier Fernandez. Java 9 Concurrency Cookbook - Second Edition, Packt Publishing. 2017

Atribuire (assignment)
x:= 2 +3

Legatura (binding)
x= 2+3

Tratarea concurentei este ingreunata de
- sincronizarea accesului resurse
- efectele laterale
- nedeterminism

In programarea functionala:
- datele sunt imuabile
- o expresie poate fi inlocuita cu valoarea ei
- functiile nu au efecte laterale
- valoarea calculata de o functie depinde numai de valorile argumentelor

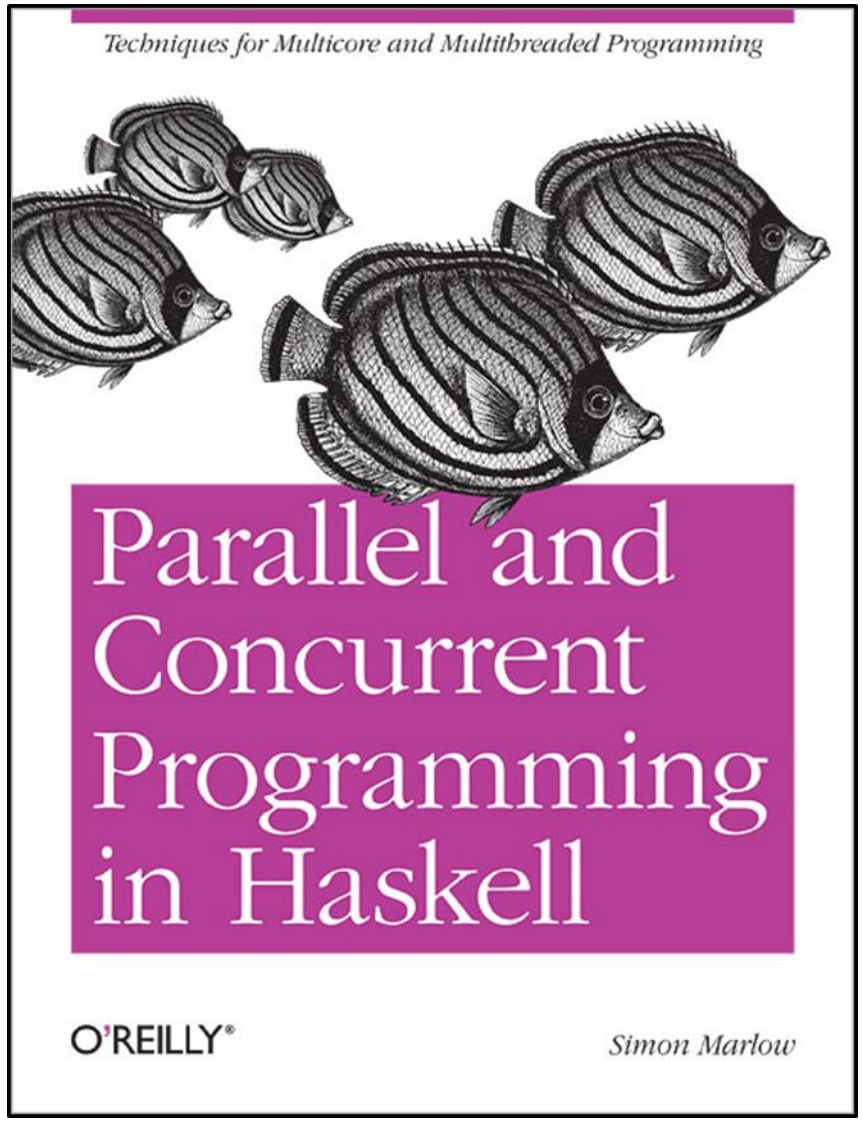


STM (Software Transactional Memory)

"Haskell does not take a stance on which concurrent programming model is best: **actors, shared memory, and transactions** are all supported, for example."

"Haskell provides all of these concurrent programming models and more - but this flexibility is a double-edged sword.

The advantage is that you can choose from a wide range of tools and pick the one best suited to the task at hand, but the disadvantage is that it can be hard to decide which tool is best for the job."
S. Marlow



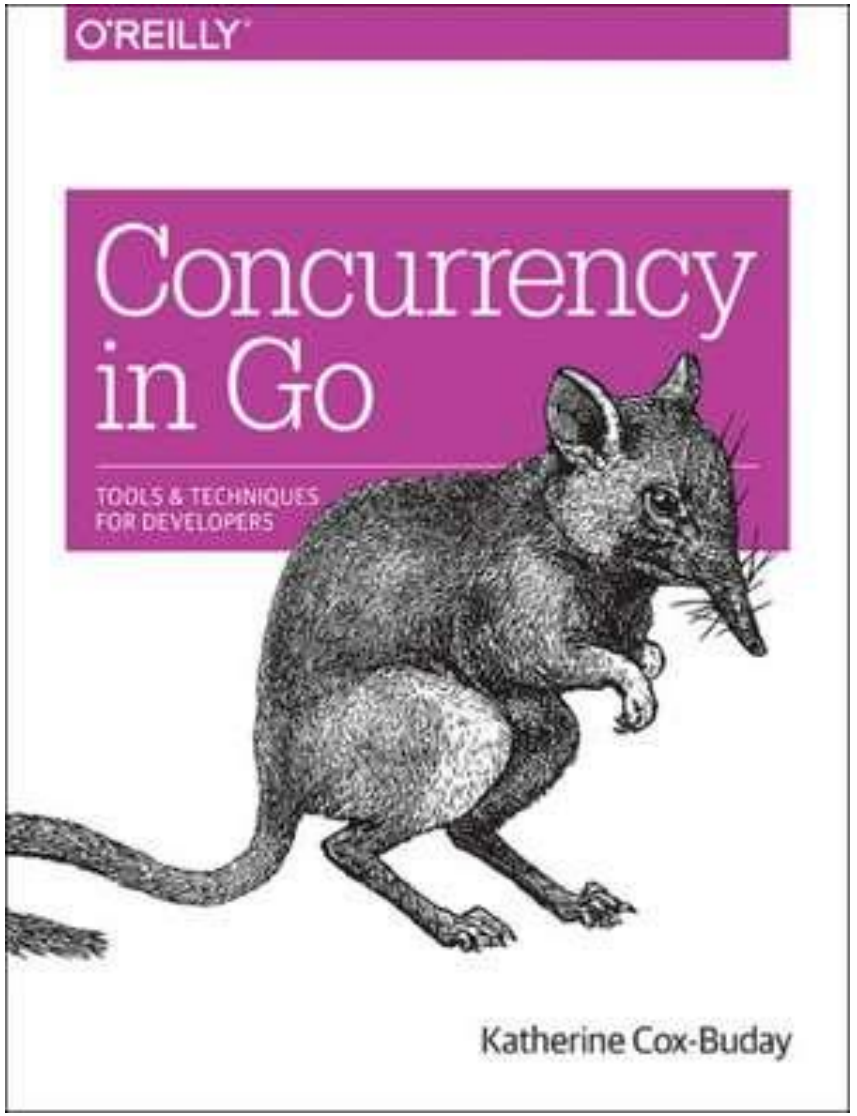
Goroutines and channels

"Before Go was first revealed to the public, this was where the chain of abstraction ended for most of the popular programming languages. If you wanted to write concurrent code, you would model your program in terms of threads and synchronize the access to the memory between them. If you had a lot of things you had to model concurrently and your machine couldn't handle that many threads, you created a *thread pool* and multiplexed your operations onto the thread pool.

Go has added another link in that chain: the *goroutine*. In addition, Go has borrowed several concepts from the work of famed computer scientist Tony Hoare, and introduced new primitives for us to use, namely *channels*.

…

Threads are still there, of course, but we find that we rarely have to think about our problem space in terms of OS threads. Instead, we model things in goroutines and channels, and occasionally shared memory."



Modelul cu Actori

"**Erlang was designed from the bottom up to program concurrent, distributed, fault-tolerant, scalable, soft, real-time systems**. […]
If your problem is concurrent, if you are building a multiuser system, or if you are building a system that evolves with time, then using Erlang might save you a lot of work, since Erlang was explicitly designed for building such systems. […]
Processes interact by one method, and one method only, by exchanging messages. Processes share no data with other processes. This is the reason why we can easily distribute Erlang programs over multicores or networks. "

Joe Armstrong, Programming Erlang, Second Edition 2013