Verteiltes Rechnen und Parallelprogrammierung:  
*Introduction to Multi-Threading in Java*

Based on the book (chapter 29):  
„Introduction to Java Programming“ (Comprehensive Version) by Y. Daniel Liang

Based on slides by: Dr. Shyamal Mitra
Today – Our Goals

- Study the use of threads in Java
- See an example of threads used in animations
- Key-words:
  Threads, Shared Resources, Synchronization, Locks, Lock Graphs, Critical Sections, Mutual Exclusion, Deadlocks, Starvation, Race Conditions, Dining Philosopher Problem, Types of Parallelization, Shared Memory, Distributed Memory, OpenMP, Message Passing (MPI), Amdahl’s Law, Granularity, Locality, Load Balancing, Data Dependency, Communication Patterns
• Recall use of **ArrayList** and **LinkedList** classes
  – examples of collection classes
• A **Collections** class also exists
  – one of its methods is `sort()`
  – given any List of objects that can be compared, it will arrange them in ascending order
• Our example creates an **ArrayList** of 1,000,000 items
  – contrasts two different sort routines
• First approach: 
  Build an ArrayList of \( n \) random Integer values 
  – measure time required for the sort

• Second approach: 
  Build two ArrayList objects of \( n/2 \) random Integers
  a) measure time to sort first list using Sorter object running as separate thread
  b) sort second list normally
  c) merge the two into a single sorted list of length \( n \)
• On a computer with single CPU, first method should be slightly faster

• On a computer with multiple CPUs, second method slightly faster
  – thread performing step a) runs on one CPU
  – step b) performed on different CPU
• On uniprocessor (Pentium II)
  – single-threaded sort took less time
  – double threaded routine must share processor
• On multiprocessor (Sun)
  – two-threaded sort faster
  – multiple CPUs allow two threads to run simultaneously
  – give 30 percent faster results
Today - Our Goals

• Appreciate the (increasing) importance of parallel programming

• Understand fundamental concepts:
  – Parallelism, threads, multi-threading, concurrency, locks, etc.

• See some basics of this done in Java
• An area of rapid change!
  – 1990s: parallel computers were $$$$  
  – Now: 4 core machines are commodity
• Variations between languages
• Evolving frameworks, models, etc.
  – E.g. Java’s got Fork/Join in Java 1.7  
  – MAP/REDUCE
• Assume a computer has one CPU
• Can only execute one statement at a time
  – Thus one program at a time
• Process:
  an operating-system level “unit of execution”
• Multi-processing
  – Computer appears to do more than one program (or background process) at a time
Today - Tasks and Threads

• **Thread**: “a thread of execution”
  • “Smaller”, “lighter” than a **process**
  • smallest unit of processing that can be scheduled by an operating system
  • Has its own run-time call stack, copies of the CPU’s registers, its own program counter, etc.
  • Process has its own memory address space, but threads share one address space

• A single program can be multi-threaded
  • Time-slicing done just like in multiprocessing
  • Repeat: the threads share the same memory
Today - Task

- A **task** is an abstraction of a series of steps
  - Might be done in a separate thread

- In Java, there are a number of classes / interfaces that basically correspond to this
  - Example (details soon): Runnable
    - work done by method run()
Today - Intro

- Thread: single sequential flow of control within a program
- Single-threaded program can handle one task at any time.
- Multitasking allows single processor to run several concurrent threads.
- Most modern operating systems support multitasking.
Threads - Advantages of Multithreading

• Reactive systems – constantly monitoring
• More responsive to user input – GUI application can interrupt a time-consuming task
• Server can handle multiple clients simultaneously
• Can take advantage of parallel processing
Different processes do not share memory space.

A thread can execute concurrently with other threads within a single process.

All threads managed by the JVM share memory space and can communicate with each other.
Threads - Concept

Multiple threads on multiple CPUs

- Thread 1
- Thread 2
- Thread 3

Multiple threads sharing a single CPU

- Thread 1
- Thread 2
- Thread 3
Threads - Concept
Thank you for your attention!

Questions?
Assignment 1

• Submission only through SVN
• Use some SVN client to access your personal SVN folder located at:
  https://svn.imp.fu-berlin.de/sopra_ppdc/PPDC2015/[ZEDAT NAME]
• Create sub-folder “assignment1”
• Commit your files into the created sub-folder
• For more information visit course web-site:
  http://biopatterngroup.de/index.php/teaching/
The Sleeping Barber Problem
The Sleeping Barber

time 9 00
barber open
barber sleep
time 9 08
customer look
customer wake
time 9 09
barber serve
time 9 16
barber done
barber sleep
time 9 19
customer look
customer wake
time 9 20
barber serve
time 9 21