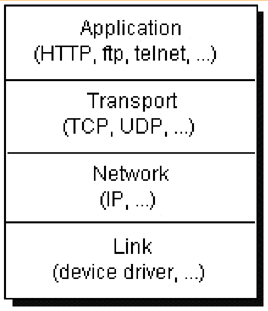
NETWERKPROGRAMMATIE JAVA

Veerle Ongenae

Overzicht

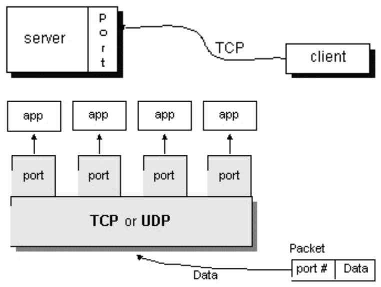
- Herhaling

TCP - UDP



https://docs.oracle.com/javase/tutorial/networking/overview/networking.html

Poorten



https://docs.oracle.com/javase/tutorial/networking/overview/networking.html

Netwerkprogrammatie Java

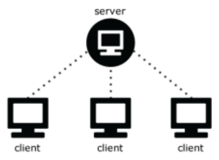
- Package `java.net`
- TCP
  - URL
  - URLConnection
  - Socket
  - ServerSocket
- UDP
  - DatagramPacket
  - DatagramSocket
  - MulticastSocket

Overzicht

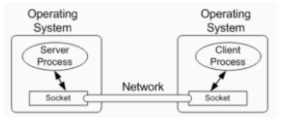
- Herhaling
- Client-Serverapplicatie

Client-server

7



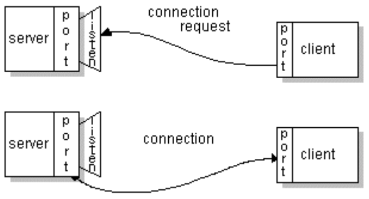
http://gamedevelopment.tutsplus.com/tutorials/building-a-peer-to-peer-multiplayer-networked-game--gamedev-10074



http://www.cs.bgu.ac.il/~spl121/Sockets

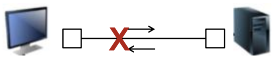
Socket

8



https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

Client



9

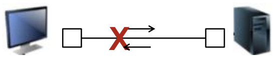
```
String hostName = "localhost";
int portNumber = 4444;

try (Socket echoSocket = new Socket(hostName, portNumber);
     PrintWriter out = new PrintWriter(echoSocket.getOutputStream(), true);
     BufferedReader in = new BufferedReader(
       new InputStreamReader(echoSocket.getInputStream()));
     BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in))) {

  String userInput = leesTekst(stdIn);
  while (!userInput.equals("exit")) {
    out.println(userInput); // naar server
    System.out.println("echo: " + in.readLine()); // van server
    userInput = leesTekst(stdIn);
  }
} catch (IOException ex) {
    Logger.getLogger(EchoClient.class.getName()).log(Level.SEVERE, null, ex);
}
```

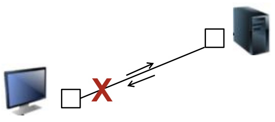
Server

10



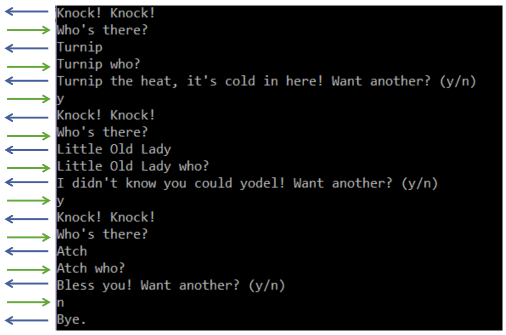
Server

11



KnockKnockServer

12



```
Knock! Knock!
Who's there?
Turnip
Turnip who?
Turnip the heat, it's cold in here! Want another? (y/n)
y
Knock! Knock!
Who's there?
Little Old Lady
Little Old Lady who?
I didn't know you could yodel! Want another? (y/n)
y
Knock! Knock!
Who's there?
Atch
Atch who?
Bless you! Want another? (y/n)
n
Bye.
```

KnockKnockServer

13

```
try {
  ServerSocket serverSocket = new ServerSocket(4444);
  while (true) {
    try (Socket clientSocket = serverSocket.accept())
      try (PrintWriter out
           = new PrintWriter(clientSocket.getOutputStream(), true);
          BufferedReader in = new BufferedReader(
           new InputStreamReader(clientSocket.getInputStream()))){
        String inputLine, outputLine;
        while ((inputLine = in.readLine()) != null) {
          outputLine = kkp.processInput(inputLine);
          out.println(outputLine);
          if (outputLine.equals("Bye.")) {break;}
        }
      } …
```

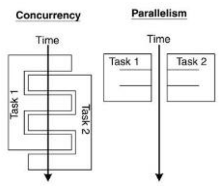
Overzicht

14

- Herhaling
- Client-Serverapplicatie
- Concurrency

Concurrency

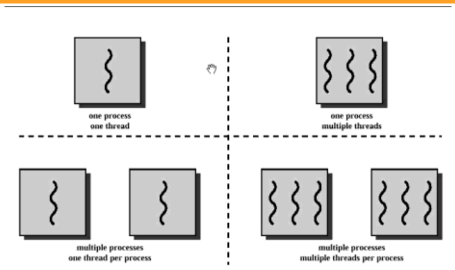
15



http://sourcecodemania.com/thread-programming-in-java/

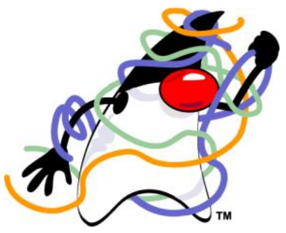
Processen - threads

16



Thread - Executor

17



http://crunchify.com/java-simple-thread-example/

Overzicht

18

- Herhaling
- Client-Serverapplicatie
- Concurrency
  - Threads

Thread – optie 1

19

```
public class HelloRunnable implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }

}
```

Thread – optie 2

20

```
public class HelloThread extends Thread {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new HelloThread()).start();
    }

}
```

Thread – optie 3

21

```
public static void main(String args[]) {
    Runnable task = () -> {
            System.out.println("Hello from a thread!");
         };
    new Thread(task).start();
}
```

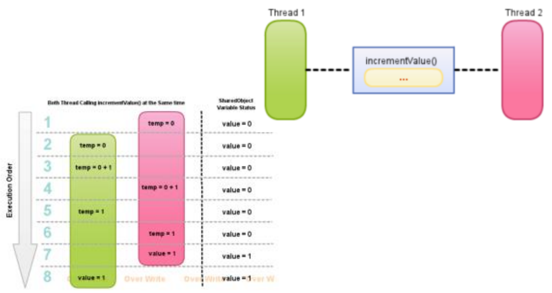
Overzicht

22

- Herhaling
- URL's in Java
- Client-Serverapplicatie
- Concurrency
  - Threads
  - Synchronisatie

Data delen

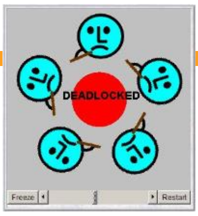
23



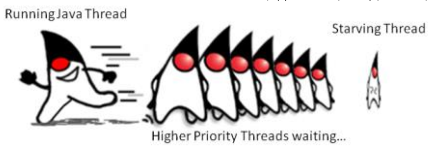
http://blog.jbaysolutions.com/2011/10/06/java-threading-and-concurrency-introduction/

Synchronisatie

24



http://crunchify.com/java-simple-thread-example/



http://avaldes.com/java-thread-starvation-livelock-with-examples/

High level concurrency

25

- Lock-objecten
- Executors
- Concurrent collections
- Atomic variables
- ThreadLocalRandom

Overzicht

26

- Herhaling
- URL's in Java
- Client-Serverapplicatie
- Concurrency
  - Threads
  - Synchronisatie
  - Executor

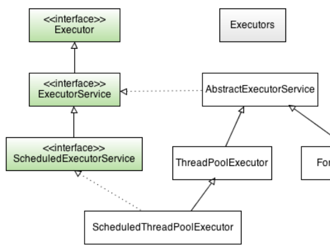
Executor

27

- Scheiding
  - Beheren threads
  - Aanmaken threads

Executor interfaces

28



```
Executor e = …
Runnable r = …
e.execute(r);
```

java.util.concurrent

https://myshadesofgray.wordpress.com/2014/04/13/java-executor-framework/

ExecutorService - aanmaken

29

```
ExecutorService execServ1 = Executors.newSingleThreadExecutor();

ExecutorService execServ2 = Executors.newFixedThreadPool(10);

ExecutorService execServ3 = Executors.newScheduledThreadPool(10);
```

ExecutorService - gebruik

30

```
executorService.execute(new Runnable() {
    @Override
    public void run() {
        System.out.println("Asynchronous task");
    }
});
```

```
executorService.submit(() -> {
    System.out.println("Asynchronous task");
});
```

```
Future<String> future = executorService.submit(new Callable(){
    @Override
    public String call() throws Exception {
        System.out.println("Asynchronous Callable");
        return "Callable Result";
    }
});
try {
    System.out.println("future.get() = " + future.get());
} catch (InterruptedException | ExecutionException ex) { … }
```

ExecutorService – reeks opdrachten

31

```
ExecutorService executorService = Executors.newSingleThreadExecutor();

Set<Callable<String>> callables = new HashSet<>();
callables.add((Callable<String>) () -> "Task 1");
callables.add((Callable<String>) () -> "Task 2");
callables.add((Callable<String>) () -> "Task 3");

try {
  List<Future<String>> futures = executorService.invokeAll(callables);

  for(Future<String> future : futures)
    System.out.println("future.get = " + future.get());

} catch (InterruptedException | ExecutionException ex) { … }

executorService.shutdown();
```

ExecutorService – reeks opdrachten

32

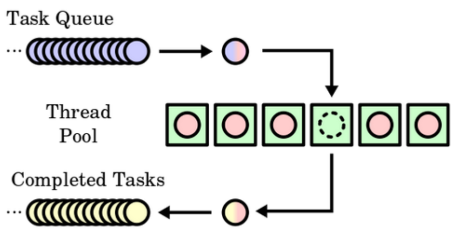
```
ExecutorService executorService = Executors.newSingleThreadExecutor();

Set<Callable<String>> callables = new HashSet<>();
for (int i = 1; i <= 100; i++) {
  final int j = i;
  callables.add((Callable<String>) () -> {
    String opdracht = "Task " + j;
    System.out.println(opdracht);
    return opdracht;
  });}

try {
  String result = executorService.invokeAny(callables);
  System.out.println("result = " + result);
} catch (InterruptedException | ExecutionException ex) {
  Logger.getLogger(…);
}
```

Thread Pools

33



https://en.wikipedia.org/wiki/Thread_pool_pattern

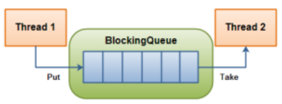
Overzicht

34

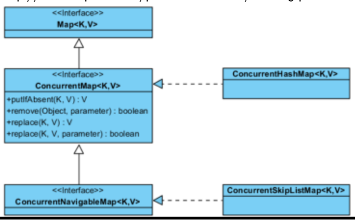
- Herhaling
- URL's in Java
- Client-Serverapplicatie
- Concurrency
  - Threads
  - Synchronisatie
  - Executor
  - Concurrent Collections

Concurrent Collections

35



http://tutorials.jenkov.com/java-util-concurrent/blockingqueue.html

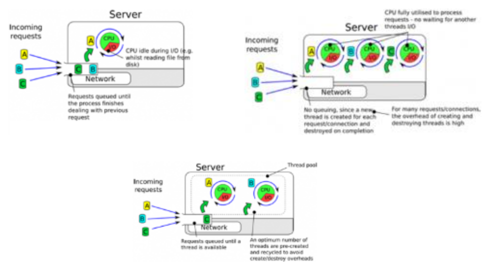


Overzicht

36

- Herhaling
- Client-Serverapplicatie
- Concurrency
  - Threads
  - Synchronisatie
  - Executor
  - Concurrent Collections
- Multithreaded Server

Singlethreaded Server


Server
Incoming requests
A
B
C
CPU
I/O
CPU idle during I/O (e.g. whilst reading file from disk)
Network
Requests queued until the process finishes dealing with previous request
Server
CPU fully utilised to process requests - no waiting for another threads I/O
Incoming requests
Network
No queuing, since a new thread is created for each request/connection and destroyed on completion
For many requests/connections, the overhead of creating and destroying threads is high
Server
Thread pool
Incoming requests
Network
Requests queued until a thread is available
An optimum number of threads are pre-created and recycled to avoid create/destroy overheads

https://myshadesofgray.wordpress.com/2014/04/13/java-executor-framework/

KnockKnockServer

```
int portNumber = 9999;
boolean listening = true;
try (ServerSocket serverSocket = new ServerSocket(portNumber)) {
  while (listening) {
    new KnockKnockThread(serverSocket.accept()).start();
  }
} catch (IOException e) {
  Logger.getLogger(
    KnockKnockMultiServer.class.getName()).log(Level.SEVERE,
       null, e);
  System.err.println("Could not listen on port " + portNumber);
  throw new RuntimeException(e);
}
```

KnockKnockServer - Thread

```
public class KnockKnockThread extends Thread {
  private Socket socket = null;
  public KnockKnockThread(Socket socket) {
    super("KKMultiServerThread");
    this.socket = socket;
  }
  @Override
  public void run() {
    try (
      PrintWriter out = new PrintWriter(socket.getOutputStream(), true)
      BufferedReader in = new BufferedReader(
        new InputStreamReader(socket.getInputStream()));)
      {…}
      socket.close();
```

KnockKnockServer - Executor

```
int portNumber = 9999;
boolean listening = true;
try (ServerSocket serverSocket = new ServerSocket(portNumber)) {
  ExecutorService execServ = Executors.newFixedThreadPool(10);
  while (listening) {
    try {
      Socket clientSocket = serverSocket.accept();
      execServ.submit(new KnockKnockRunnable(clientSocket));
    } catch (IOException ex) {… }
  }
} catch (IOException e) {
  Logger.getLogger(
    KnockKnockPool.class.getName()).log(Level.SEVERE, null, e);
  System.err.println("Could not listen on port " + portNumber);
  throw new RuntimeException(e);
}
```