Creating and Managing Threads

David Flynn
SOFTWARE ENGINEER, FLYNN IT LTD

Introduction

Topics we'll cover:

- How to start threads
- Monitor when they finish or die
- How to make them sleep
- How to interrupt them
- Misc. topics

Demo

'Hello World!' Demo

Prints a greeting to multiple countries

Nothing special yet – just single-threaded

Creating Threads

To start a thread:

- Create an instance of java.lang.Thread
- call start()

New thread starts at run() method

Standard threads persist until exiting run()

Demo

Same demo, but multithreaded this time

Shows how to create threads by extending the Thread class

Demo

Investigation of run() vs start()

Same program, but calling run() directly

Demo

Investigation of run() vs start()

Same demo but with the thread ID printed

To Create Threads Call start()

Must call start() and not run() to create a thread

This is tested on Java exams

Easy mistake to make

Problems Inheriting from Thread

Should prefer composition over inheritance

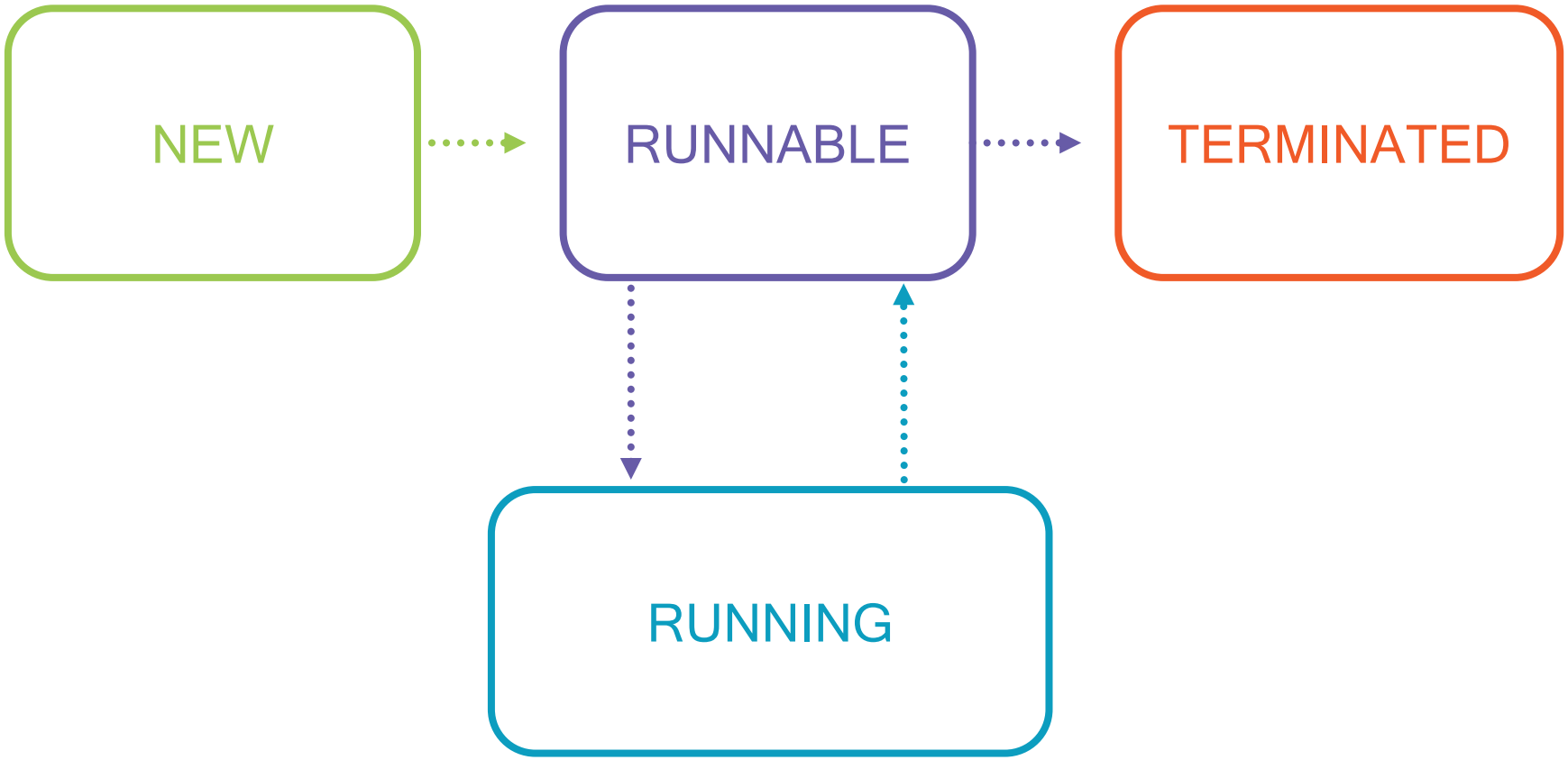
Inheritance hierarchies get unwieldy

Can only inherit from one class in Java

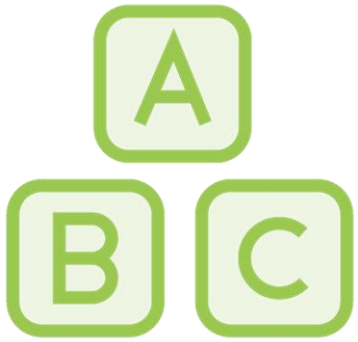
Demo

Creating Multithreaded Runnable

The Thread State Engine



start() Can Be Called Once



Calling start() again throws an IllegalThreadStateException

Can use getState() to obtain the thread's state

Daemon Threads

Exists as long as some non-daemon threads exist

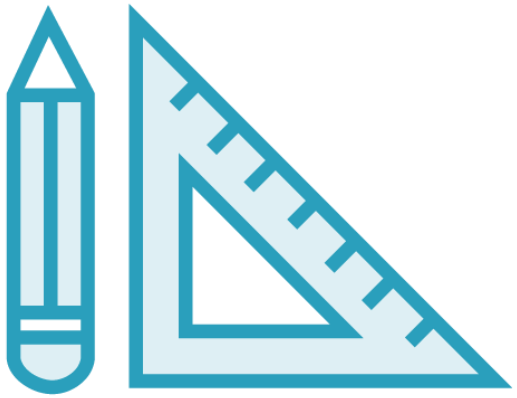
May evaporate if no non-daemon threads left

- No clean up, no finalizers called

Used for support threads

- e.g. garbage collector

Creating Daemon Threads



To designate, call setDaemon(true) before start

Should never handle resources which need to be:

- Closed properly
- Cleaned up

Don't mark threads daemon just to make shutdown easier

Runnable's Interface

marked
@FunctionalInterface

Can use with lambda
expressions safely

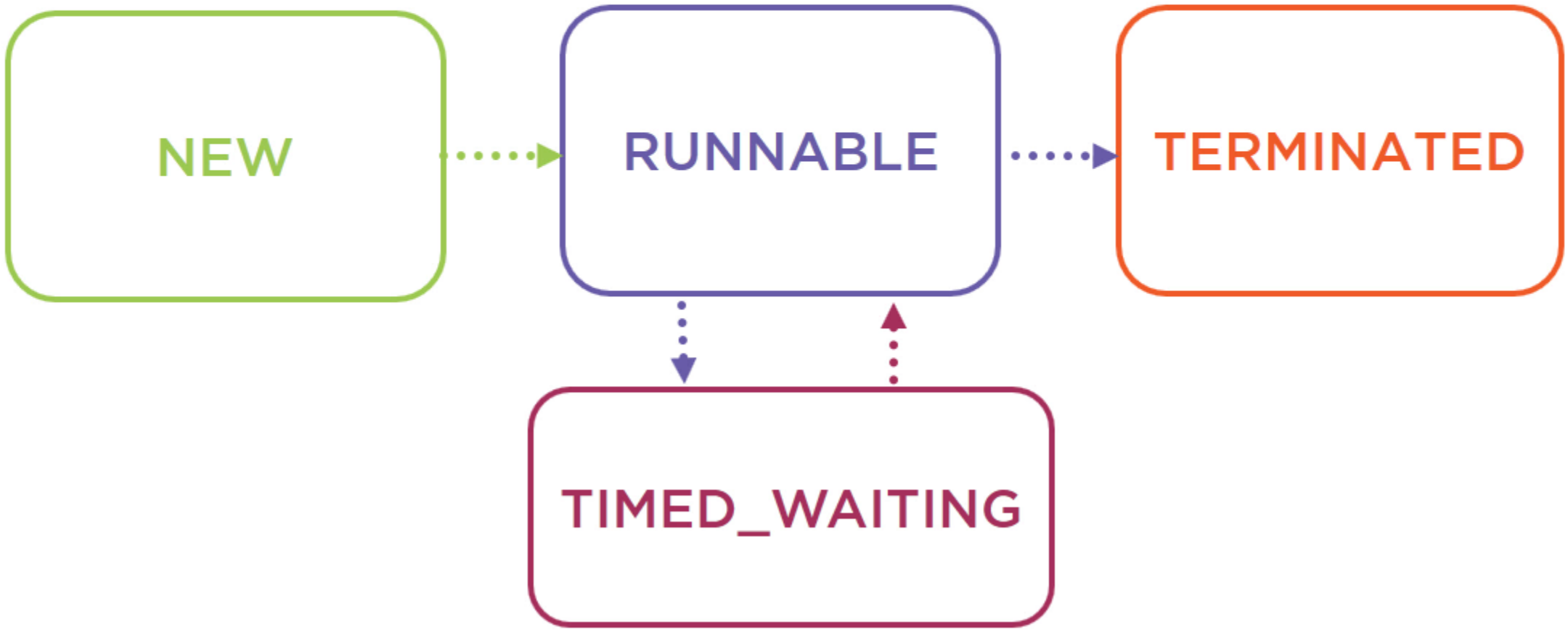
Sleeping

Can use loop to wait for period to expire

- But doesn't block
- Takes CPU resource away from other threads and processes

Better is calling Thread.sleep(long millis)

The Thread State Machine



Demo

Multithreaded HelloWorld with a sleep(0) in it

Sleep Is Overloaded

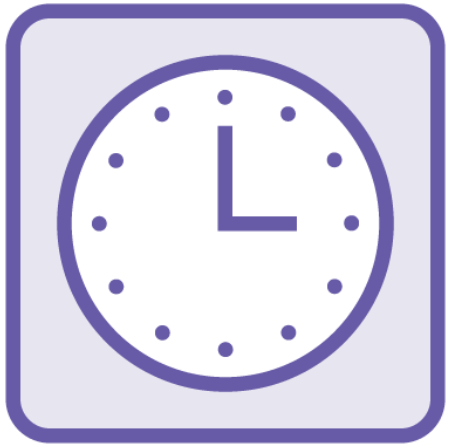
void sleep(long millis)

void sleep(long millis, int nanos)

In Java SE the nanoseconds are rounded

- Need to look at the code to see this!

TimeUnit



Class for working with time units

Convenience void sleep(long timeout) method

- Calls Thread's sleep with the appropriate values

Demo

How to use TimeUnit's sleep

sleep(0) and yield()

Platform dependent

- sleep(0) may or may not sleep
- yield() may or may not yield

Avoid unless target platform requires it

void setPriority(int newPriority)

Use to change thread priorities to give important threads more CPU

But...

- Platform dependent
- Can cause thread starvation

Avoid unless target platform requires it and thoroughly tested

jstack & jconsole

Useful tools to diagnose problems

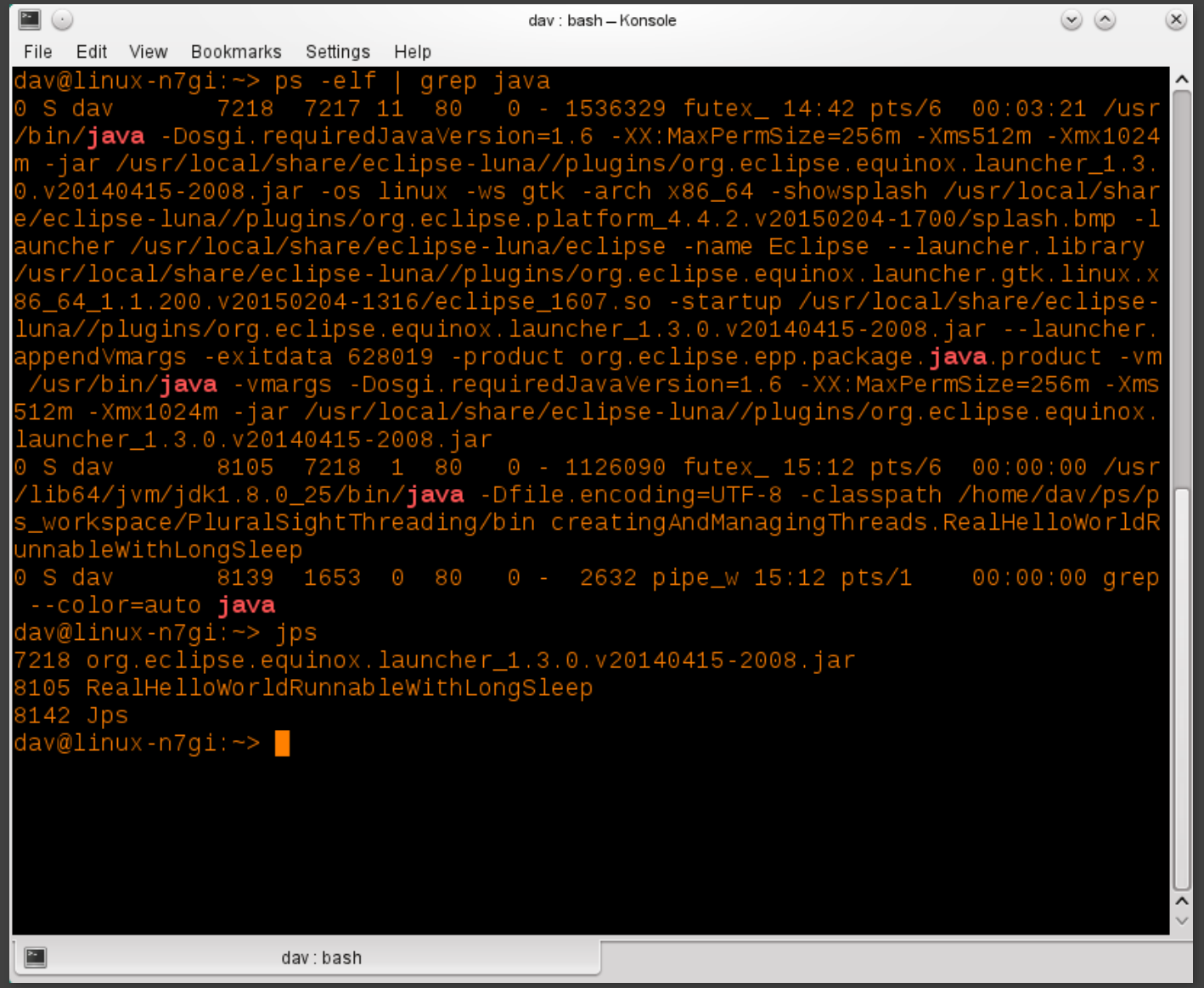
Or for the nosey!

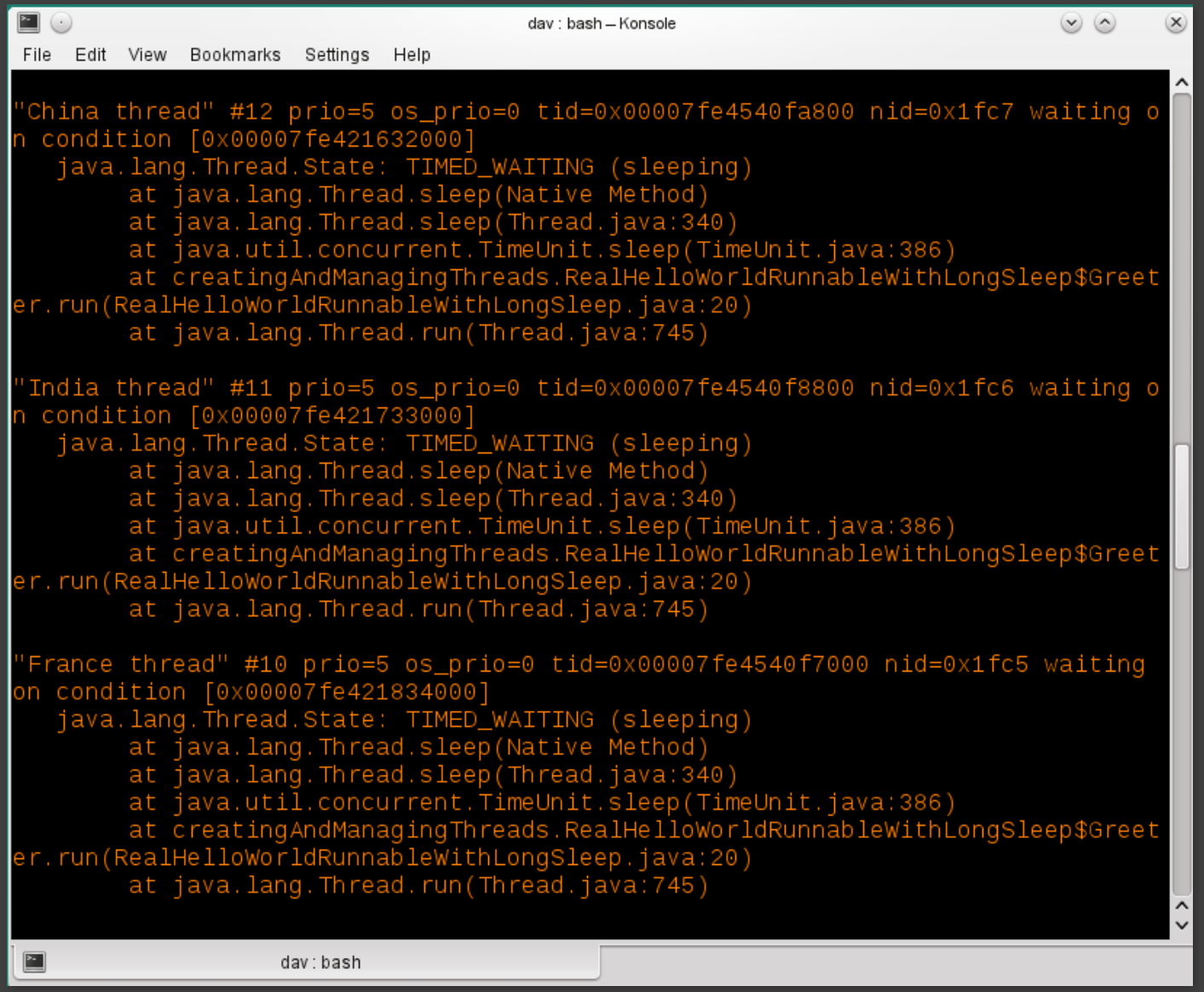
Getting the Java Process ID

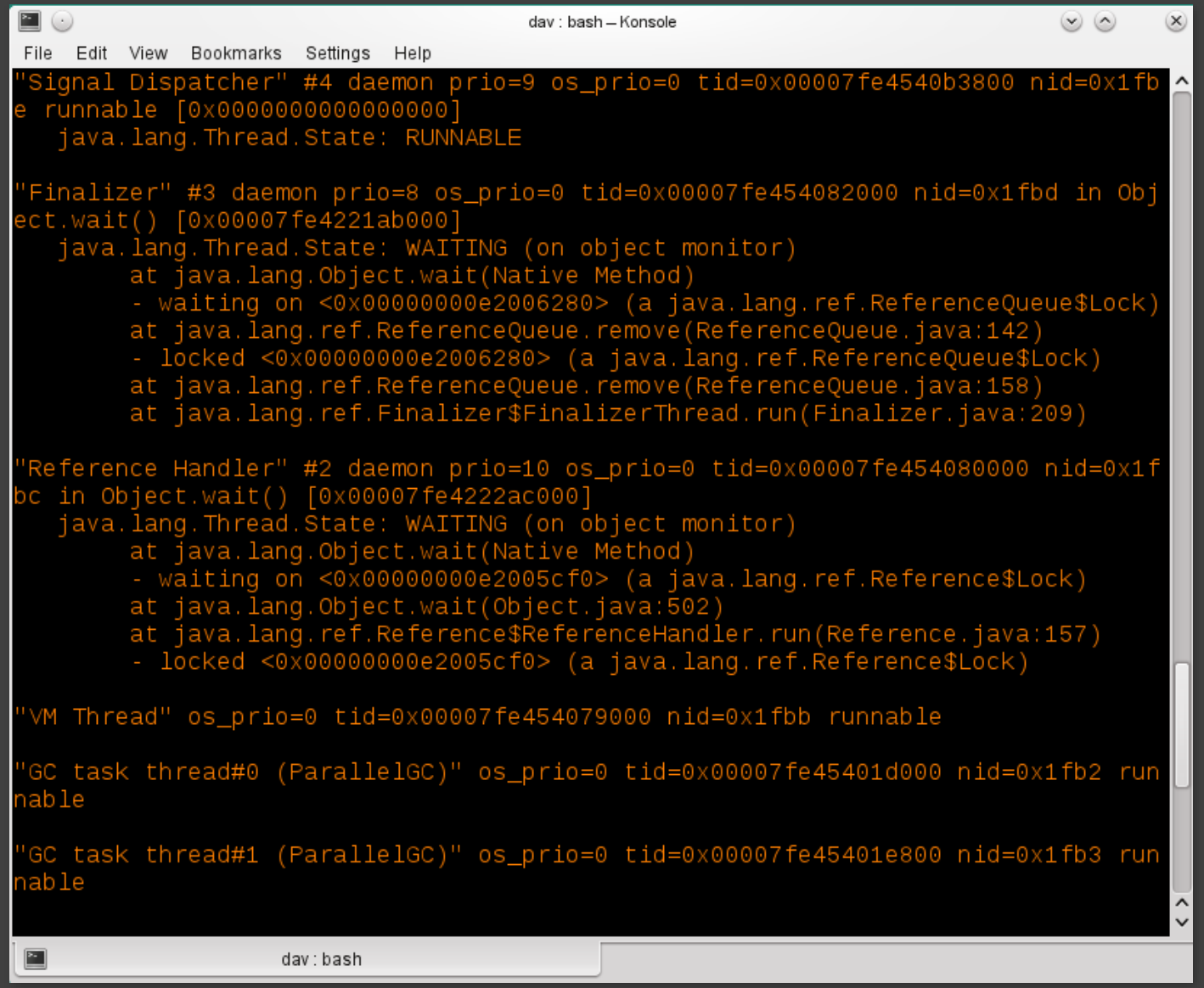
Linux: ps

Windows: Task manager or tasklist command

Platform independent: jps


dav : bash — Konsole
File Edit View Bookmarks Settings Help
dav@linux-n7gi:~> ps -elf | grep java
0 S dav 7218 7217 11 80 0 - 1536329 futex_ 14:42 pts/6 00:03:21 /usr/bin/java -Dosgi.requiredJavaVersion=1.6 -XX:MaxPermSize=256m -Xms512m -Xmx1024m -jar /usr/local/share/eclipse-luna//plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar -os linux -ws gtk -arch x86_64 -showsplash /usr/local/share/eclipse-luna//plugins/org.eclipse.platform_4.4.2.v20150204-1700/splash.bmp -launcher /usr/local/share/eclipse-luna/eclipse -name Eclipse --launcher.library /usr/local/share/eclipse-luna//plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.200.v20150204-1316/eclipse_1607.so -startup /usr/local/share/eclipse-luna//plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar --launcher.appendVmargs -exitdata 628019 -product org.eclipse.epp.package.java.product -vm /usr/bin/java -vmargs -Dosgi.requiredJavaVersion=1.6 -XX:MaxPermSize=256m -Xms512m -Xmx1024m -jar /usr/local/share/eclipse-luna//plugins/org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
0 S dav 8105 7218 1 80 0 - 1126090 futex_ 15:12 pts/6 00:00:00 /usr/lib64/jvm/jdk1.8.0_25/bin/java -Dfile.encoding=UTF-8 -classpath /home/dav/ps/ps_workspace/PluralSightThreading/bin creatingAndManagingThreads.RealHelloWorldRunnableWithLongSleep
0 S dav 8139 1653 0 80 0 - 2632 pipe_w 15:12 pts/1 00:00:00 grep --color=auto java
dav@linux-n7gi:~> jps
7218 org.eclipse.equinox.launcher_1.3.0.v20140415-2008.jar
8105 RealHelloWorldRunnableWithLongSleep
8142 Jps
dav@linux-n7gi:~>
dav : bash


dav : bash — Konsole
File Edit View Bookmarks Settings Help
"China thread" #12 prio=5 os_prio=0 tid=0x00007fe4540fa800 nid=0x1fc7 waiting o
n condition [0x00007fe421632000]
java.lang.Thread.State: TIMED_WAITING (sleeping)
at java.lang.Thread.sleep(Native Method)
at java.lang.Thread.sleep(Thread.java:340)
at java.util.concurrent.TimeUnit.sleep(TimeUnit.java:386)
at creatingAndManagingThreads.RealHelloWorldRunnableWithLongSleep$Greet
er.run(RealHelloWorldRunnableWithLongSleep.java:20)
at java.lang.Thread.run(Thread.java:745)
"India thread" #11 prio=5 os_prio=0 tid=0x00007fe4540f8800 nid=0x1fc6 waiting o
n condition [0x00007fe421733000]
java.lang.Thread.State: TIMED_WAITING (sleeping)
at java.lang.Thread.sleep(Native Method)
at java.lang.Thread.sleep(Thread.java:340)
at java.util.concurrent.TimeUnit.sleep(TimeUnit.java:386)
at creatingAndManagingThreads.RealHelloWorldRunnableWithLongSleep$Greet
er.run(RealHelloWorldRunnableWithLongSleep.java:20)
at java.lang.Thread.run(Thread.java:745)
"France thread" #10 prio=5 os_prio=0 tid=0x00007fe4540f7000 nid=0x1fc5 waiting
on condition [0x00007fe421834000]
java.lang.Thread.State: TIMED_WAITING (sleeping)
at java.lang.Thread.sleep(Native Method)
at java.lang.Thread.sleep(Thread.java:340)
at java.util.concurrent.TimeUnit.sleep(TimeUnit.java:386)
at creatingAndManagingThreads.RealHelloWorldRunnableWithLongSleep$Greet
er.run(RealHelloWorldRunnableWithLongSleep.java:20)
at java.lang.Thread.run(Thread.java:745)
dav : bash


dav : bash – Konsole
File Edit View Bookmarks Settings Help
"Signal Dispatcher" #4 daemon prio=9 os_prio=0 tid=0x00007fe4540b3800 nid=0x1fb
e runnable [0x0000000000000000]
java.lang.Thread.State: RUNNABLE
"Finalizer" #3 daemon prio=8 os_prio=0 tid=0x00007fe454082000 nid=0x1fbd in Obj
ect.wait() [0x00007fe4221ab000]
java.lang.Thread.State: WAITING (on object monitor)
at java.lang.Object.wait(Native Method)
- waiting on <0x00000000e2006280> (a java.lang.ref.ReferenceQueue$Lock)
at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:142)
- locked <0x00000000e2006280> (a java.lang.ref.ReferenceQueue$Lock)
at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:158)
at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:209)
"Reference Handler" #2 daemon prio=10 os_prio=0 tid=0x00007fe454080000 nid=0x1f
bc in Object.wait() [0x00007fe4222ac000]
java.lang.Thread.State: WAITING (on object monitor)
at java.lang.Object.wait(Native Method)
- waiting on <0x00000000e2005cf0> (a java.lang.ref.Reference$Lock)
at java.lang.Object.wait(Object.java:502)
at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:157)
- locked <0x00000000e2005cf0> (a java.lang.ref.Reference$Lock)
"VM Thread" os_prio=0 tid=0x00007fe454079000 nid=0x1fbb runnable
"GC task thread#0 (ParallelGC)" os_prio=0 tid=0x00007fe45401d000 nid=0x1fb2 run
nable
"GC task thread#1 (ParallelGC)" os_prio=0 tid=0x00007fe45401e800 nid=0x1fb3 run
nable
dav : bash

Demo

Java monitoring and management console

'jconsole'

InterruptedException

Thrown by sleep

When interrupt() is called on the thread

Checked exception, must be caught somewhere

Ignoring InterruptedException

Empty catch block
- Should add a comment

Only safe if interrupts will not happen
- E.g. small tools and demos

More complex programs will use it to shutdown properly

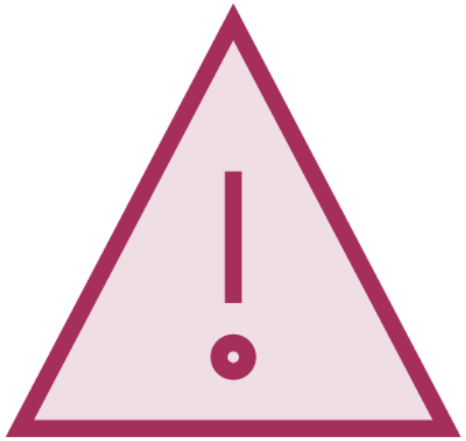
Libraries: Unsure What to Do?

Throw the InterruptedException

- Has to be caught before run exits
- Code that does know what to do will deal with it

Must never handle unless responsible for interruption policy

Checking for Interrupts



interrupted() - checks for and clears the interrupt

isInterrupted() - checks for but doesn't clear the interrupt

Demo

'Sleepy Bartender' demo

Demonstrates handling InterruptedException

Although it really should be used in exceptional circumstances or to quit and cancel

Need to Catch Unchecked Exceptions

Thread could set a shared flag which we can check

It'd have to also catch unchecked exceptions

- Otherwise the flag won't be set

Clumsy approach

isAlive()

Threads are alive after NEW until TERMINATED

Call isAlive() to check if is alive

Demo

'Sleepy bartender 2'

Shows use of join and correct use of interrupt

Join Is Overloaded

```
void join() // untimed version

void join(long millis)

void join(long millis, int nanos)
```

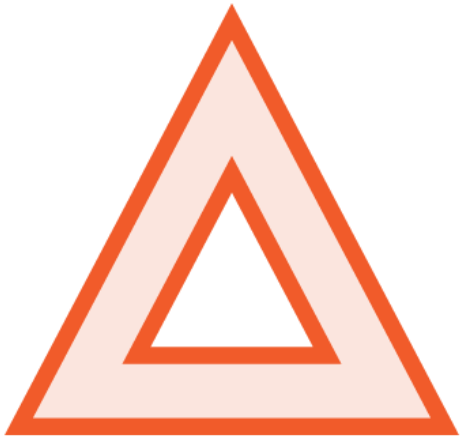
Join

Must check isAlive() after timed join

Join with 0 ms calls untimed version

timedJoin(thread, 0) in TimeUnit doesn't do anything

Catching Exceptions Thrown from run()



Could catch before exiting and save

Or

Set uncaughtExceptionHandler

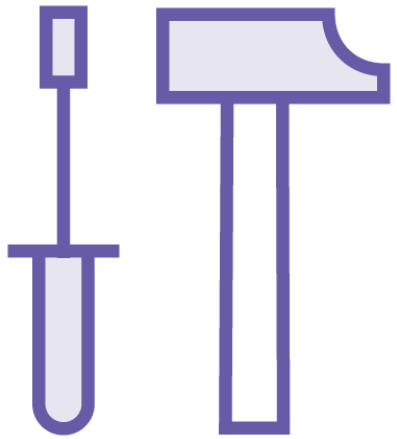
Must wrap checked exceptions as these cannot be thrown from run

How the JVM Handles Uncaught Exceptions from Threads

dispatchUncaughtException
called by JVM

Uses uncaughtExceptionHandler
to determine
what to call

Making a Custom Exception Handler



void setUncaughtExceptionHandler
(UncaughtExceptionHandler eh)

Pass an instance of a class which implements UncaughtExceptionHandler

Implement uncaughtException method

Default ThreadGroup Handler

Checks if defaultUncaughtExceptionHandler has been set

- Can set by setDefaultUncaughtExceptionHandler on Thread

Otherwise prints stack trace

Example Custom Handlers

Send details to logger

Pass exception report on

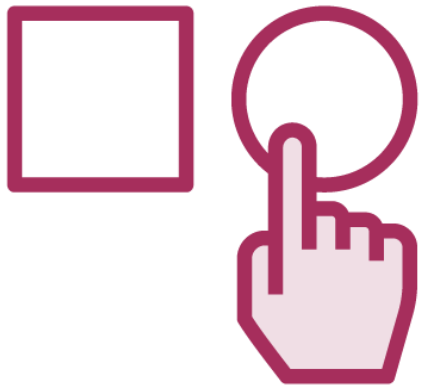
Recreate thread and retry

Demo

'Goodbye World'

Shows how to use a custom uncaughtExceptionHandler

ThreadGroup (Virtually Deprecated)



For handling groups of threads:

- Setting priority
- Interrupting
- Logical grouping due to function
- Security settings - never worked well

ThreadLocal

Want to avoid global variables

- Cause additional coupling
- Modifications can cause problems

But useful:

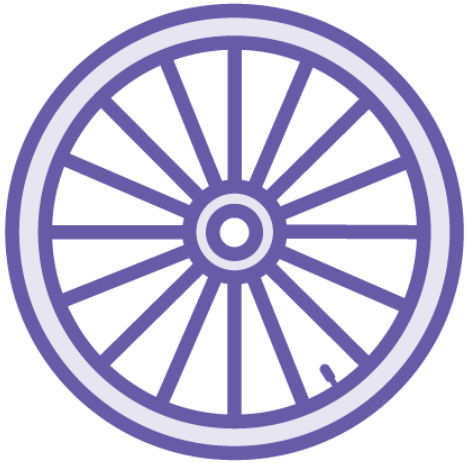
- To not have to pass values for formatters and loggers
- For threads to have their own copy accessed via same global reference

Demo

'Conflicting SimpleDateFormat'

Shows problem of sharing a SimpleDateFormat in a global variable

Using ThreadLocal



Rolling our own isn't trivial

- It might seem it's just a HashMap, but getting tricky getting performance and correctness right

ThreadLocal is the official implementation

- It's tempting to want to reinvent the wheel, but never do so unless you really have to.

ThreadLocal with Default Values

Extend ThreadLocal

- Override initialValue method

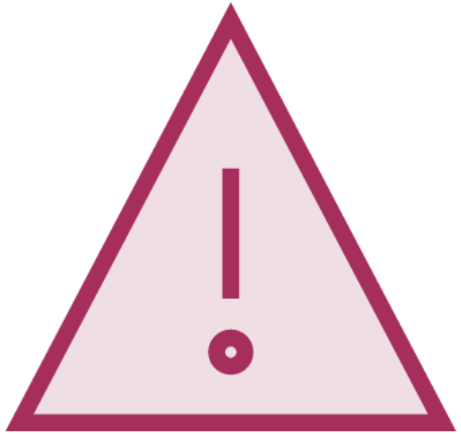
If we call get without set, it stores initialValue

- Thus can change without affecting other's defaults

Demo

Showing how to use ThreadLocal with initial values

Beware of Leaks



Data persists until thread dies or no instances of ThreadLocal left

Danger with long-lived threads

- If task finishes/dies without clearing out Threadlocal objects
- Lose knowledge that thread stored an object
- Effectively leaked memory
- Need to make sure we call remove on ThreadLocals

Stop, Suspend, Resume

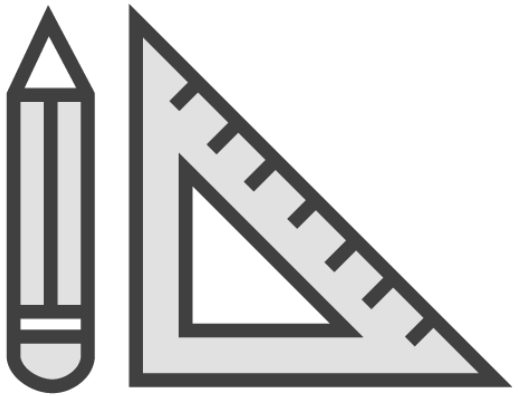
Might seem useful, but...

- Dangerous to use
- Should we release or hang on to mutexes and resources?

Therefore deprecated

- Use other mechanisms, such as interrupts

Creating Threads



Create by:

- Inheriting from Thread

Or

- Passing Runnable to Thread's constructor (preferred)

Naming threads is good practice – jstack / jconsole

Call start (not run) to start

Sleep and Join

- Calling sleep to make threads sleep
- Wait for threads to die (isAlive / join)
- Using TimeUnit – makes working with timeout periods easier
- Handle InterruptedException and how to interrupt threads

Exception Handlng

Wrapping checked exceptions and rethrow

Handling uncaught exceptions – register a UncaughtExceptionHandler

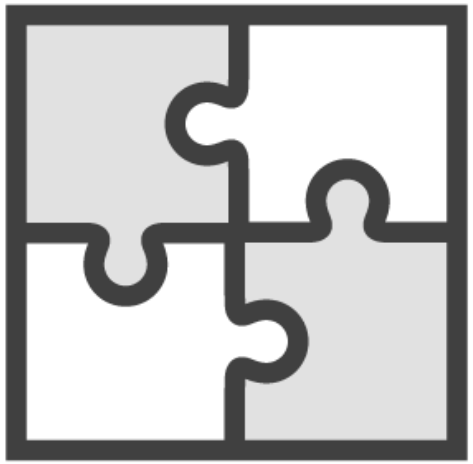
ThreadLocal

Using ThreadLocal to get a local instance of a variable

- From a global ThreadLocal variable
- Avoids dangers of modification

Be careful to avoid leaks

Deprecated Topics



- ThreadGroups
- Changing thread priority
- sleep(0)
- yield()
- stop(), suspend(), resume()