Distributed Applications – Session 1

Lecturer Mouhsen Ibrahim

Contents

Introduction

- A distributed application is an application that runs on multiple nodes to run a single task or job.
- Distributed Applications should be Highly Available, Fault tolerant and Scalable.
- Highly Available means that the application should be always up and running to serve clients, availability can be measured as percent of time when the application is UP vs when the application is DOWN.
- Fault tolerant means that the failure of one node should not affect the application and it should continue to serve clients.

Introduction

- Scalability: It means that the application can serve a sudden increase in workload without affecting performance and availability.
- Elastic: It means that the application's infrastructure can expand or shrink based on workload needs.
- Distributed applications can be created almost as normal applications the only difference is how we deploy them to achieve the previous characteristics.

Tools

- In this course we will use Java JDK 8 and eclipse for writing and running distributed applications.
- Eclipse can be downloaded from here
- http://www.eclipse.org/downloads/eclipse-packages
- JDK 8 can be downloaded from here
- http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Threads

- A thread is currently the smallest unit of execution, it is part of a process and shares the process's code and global variables.
- Multiple threads can be created in a single process to do different tasks.
- Advantages:

→ Can improve performance and use multiple processing units if available.

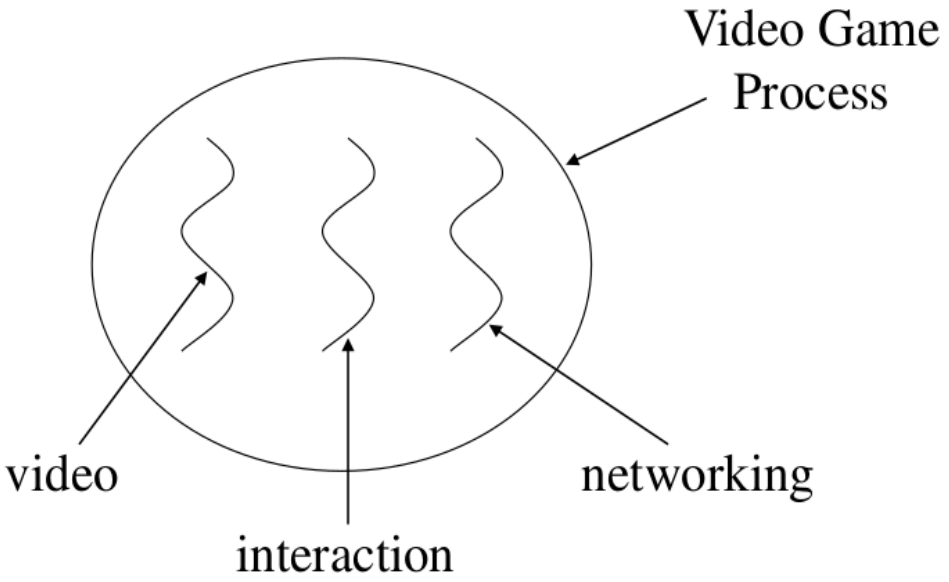
→ Threads can share resources together.

Threads

- Disadvantages

→ Threads can lead to deadlocks.

→ Overhead of switching between threads.



Java Threads

- Threads in Java can be created using two methods
  - Extending the Thread class
    - It must implement the run() method.
    - The thread ends when run() returns.
    - Call start() method to get the thread ready for running.
- Check thread_example1.java for a complete example of a program that uses threads.

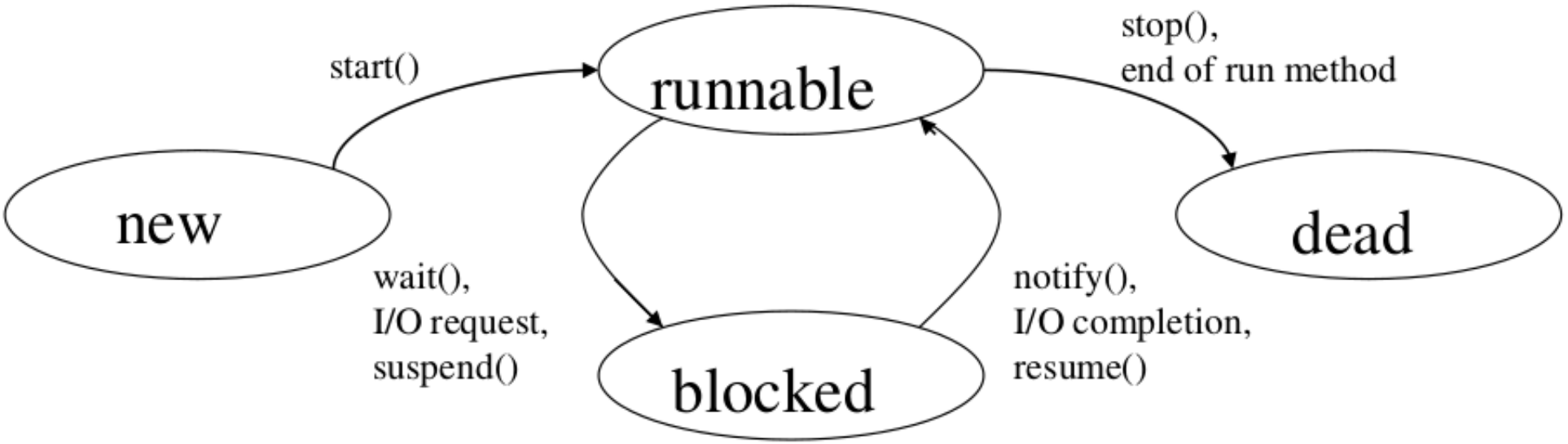
Java Threads

- Implements the Runnable interface.
- Implement the run() method.
- Use the Runnable object as an argument to the Thread class.
- Using Runnable allows you to extend other classes as well, because in Java you can only extend from one class.
- Check thread_example2.java for an example of using this method to create threads.

Controlling Java Threads

- start() method to mark thread as ready for execution.
- join() wait for a thread to finish.
- wait(), notify(), notifyAll() are used for synchronization.
- setPriority() 0 to 10 (MIN_PRIORITY to MAX_PRIORITY) 5 is default NORMAL_PRIORITY.
- yield() causes current thread to stop running to allow other threads to run.
- sleep(msec) stop execution for some time

Java Thread States



Exercise

- Write a single java class that extends the Thread class to print numbers from num1 to num2.
- Use this class in a main program to print numbers from 1 to 10 and numbers from 10 to 20.

Good Luck