Report: Answer the Following Questions  
Deadline: Monday 11-01-2021 @ 11.59 PM -- To: ahmedaboulroos@gmail.com  
**By: Mohammad** **AbouElSherbini**

Q1. What is the Difference between?  
• Process & Thread:

Process is the running instance of a program and can be divided into multiple independent threads each running in parallel   
• System Thread & User-Defined Thread & Daemon Thread

System thread: system defined threads, auto handled by jvm

User-Defined: high priority thread running in the background, prevent jvm from exit

Daemon: independent low priority thread running in the background, doesn’t prevent jvm from exit

• Parallelism & Concurrency

Parallelism: Tasks run at the same time on multicores

Concurrency: Tasks run together but not at the same time

• Synchronous & Asynchronous Programming

Synchronous: run obeying time constraints defined, can cause wait delay

Asynchronous: can run independently of each other

• Callable & Runnable & Supplier interfaces  
Callable: has call which is T (void) throws Exception

Computes a result, or throws an exception if unable to do so.

Runnable: has run which is void (void)

used to create a thread

Supplier: has call which is T (void)

• StringBuffer & StringBuilder

StringBuffer: synchronized, less effecient

StringBuilder: non synchronized, more effecient

• Synchronized Collections & Concurrent Collections

Synchronized collections: uses locking on the whole collections so it’s not efficient

Concurrent collections: uses partitioned locking so it has better performance

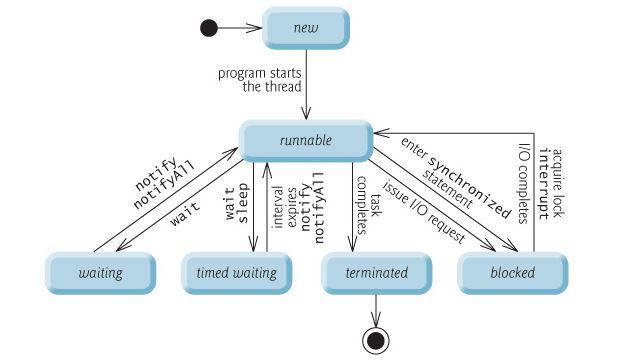
• Inter-Process Communication & Inter-Thread Communication

Inter-process: harder and slower, handled by the os

Inter-thread: faster as they share the same program memory, but synchronization is required  
Q2. What is?  
• Thread Creation Techniques

1. By extending thread class
2. By implementing runnable interface

• Thread States

  
• Context Switch

Time wasted in changing a task from running state to waiting to get another task to the running state  
• Thread Priority

Is what allowed the language to favor a thread over another, allow it to run before it’s peers

Range is 1 to 10, 5 is normal  
• Polling

Waiting for something by continually checking

• Thread Safety

Multiple thread can run safely without producing a bad program state

• Liveness

The program’s ability to execute in a timely manner

• Dead-Lock (How to Avoid, Recover, Detect)

Tasks can’t continue running as the resource they are waiting for is held by some other tasks that’s also waiting for another resource that it can’t get

How to Avoid:

1. by numbering resources in order and not allowing resource acquisition when it would cause a dead lock like never allow lower number allocations
2. force the task to request all resource in some order at the beginning
3. never allow a task to go into sleep while holding a resource

How to recover:

1. by ignoring the issue
2. by freeing resources from tasks either by request or by killing tasks

How to detect: when resource are mutually exclusive or cyclic dependent tasks

• Live-Lock

Tasks are running but not doing any useful job as they are waiting for others  
• Starvation

A task that continuously gets preempted by other tasks leading to it not being able to run  
• Thread contention

When multiple threads try to access the same resource at the same time  
• Thread interference

When multiple threads access the same data, this could put the program in a bad state

• Thread interrupts

Allows the thread to continue running if it wasn’t already running it, sets the interrupted flag, and throws Interrupted exception

• Memory Consistency errors

It’s when the changes made by a thread isn’t visible correctly to other threads

• Synchronization

Is to control access to a resource from multiple threads

• Fork/Join Framework (and when to use it)

Used to break a task into smaller tasks to be processed in multiple processes

Like breaking a recursive process to multiple smaller processes to calculate a function (fork), then waiting for it to finish in the calling thread (join)  
• Race Condition

When multiple threads try to access the same “synchronized” resource at the same time leading to a race between them to determine who gets it first

• Shared Mutability

Sharing the ability to change an objects state with other

• Double Checked Locking of Singleton  
Used to make sure the singleton is really a singleton in multithreaded environments

• Parallel Streams

Stream are used to process collections of objects then transforming them or reducing them to some usable useful form, and parallel streams runs them in parallel making them faster  
Q3. What happens when an Exception occurs in a thread?

It will die if it’s not handled