1.What do you mean by multithreading? Why is it important?

Ans: Multithreading is the ability of a program or an operating system to enable more than one user at a time without requiring multiple copies of the program running on the computer.

Multithreading can also handle multiple requests from the same user.

2. What are the benefits of using multithreading?

Ans: Some of the most important benefits of multithreading are:

- Improved throughput. Many concurrent computer operations and I/O requests within a single process.
- Simultaneous and fully symmetric use of multiple processors for computation and I/O
- Superior application responsiveness. If a request can be launched on its own thread, applications do not freeze or show the "hourglass". An entire application will not block, or otherwise wait, pending the completion of another request.
- Improved server responsiveness. Large or complex requests or slow clients don't block other requests for service. The overall throughput of the server is much greater.
- Minimized system resource usage. Threads impose minimal impact on system resources. Threads require less overhead to create, maintain, and manage than a traditional process.
- Program structure simplification. Threads can be used to simplify the structure of complex applications, such as server-class and multimedia applications. Simple routines can be written for each activity, making complex programs easier to design and code, and more adaptive to a wide variation in user demands.
- Better communication. Thread synchronization functions can be used to provide enhanced process-to-process communication. In addition, sharing large amounts of data through separate threads of execution within the same address space provides extremely high-bandwidth, low-latency communication between separate tasks within an application.

3. What is Thread in Java?

Ans: A thread is a lightweight process, or we can say the smallest part of the process that allows a program to operate more efficiently by running multiple tasks simultaneously.

4. What are the two ways of implementing thread in Java?

Ans:

- By extending thread class

- By implementing Runnable interface

5. What are the difference between thread and process?

Ans:

S.No	Process	Thread
1.	When a program is under execution, then it is known as a process.	A segment of a process is known as thread.
2.	It consumes maximum time to stop.	It consumes minimum time to stop.
3.	It needs more time for work and conception.	It needs less time for work and conception.
4.	Context switching takes maximum time here.	Here, context switching takes minimum time.
5.	It is not that effective in terms of communication.	It is effective in terms of communication.
6.	It takes more resources.	It takes less resources.
7.	It is a heavy weight process.	It is a lightweight process.
8.	If one process is obstructed, then it will not affect the operation of another process.	If one thread is obstructed, then it will affect the execution of another process

6. How can we create daemon threads?

Ans: We can create daemon threads in java using the thread class setDaemon(true). It is used to mark the current thread as daemon thread or user thread. isDaemon() method is generally used to check whether the current thread is daemon or not. If the thread is a daemon, it will return true otherwise it returns false.

```
NewThread daemonThread = new NewThread();

daemonThread.setDaemon(true);

daemonThread.start();
```

7. What are the wait() and sleep() method?

Ans:

- **Wait():** this is associated with the Object class. The Wait() function is in charge of placing the calling thread in the waiting state. The Thread will stay in the waiting state until another thread calls the notify() or notifyAll() function on that object. After gaining control of the monitor, the Thread resumes execution. It is a non-static method.
- **Sleep():** It is a static method that pauses or stops the current thread's execution for a defined period of time. It does not release the lock while waiting and is primarily used to pause execution. It is specified in the thread class, and there is no need to invoke it from a synchronized context.