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

- **Ans:-** Executing several tasks simultaneously where each task is a separate independent part of the same program is called "Thread based multitasking".
- Each independent part is called a "Thread".
- Multithrerading refers to two or more tasks executing concurrently within a single program.
- Every thread in java is created and controlled by the java.lang.Thread class.

  Why is it important?

  Every program has several independent task which is known as Thread and if some thread needs user input then another thread needs to wait till it get user input and therefore cpu lifecycle will get waste to overcome these problem we need multithreading concept.

2) What are the benefits of using multithreading?

Ans:-

- Improved throughput. Many concurrent compute operations and I/O requests within a single process.
- 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.
- 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.
- Threads are lightweight as compare to process based multitasking.

3) What is Thread in Java?

Ans:- A Thread in java is a thread of execution in a program that defines a separate path for execution.

All java programs have at least one thread that is main thread which is created by JVM at the program start.

4) What are the two ways of implementing thread in java?

Ans:-

1) By extending Thread class
2) By implementing Runnable class

5) what are the difference between Thread and process?

Ans:-

thread	process
Thread means a segment of process	Program means any program in execution
The thread takes less time to terminate	The process take more time to terminate
It takes less time for creation	It takesmore time for creation
It takes less time for context switching	It takes more time for contect switching
Thread is more efficient in terms	Process is less efficient in terms

of execution	of execution
We don't need multiprogramming concept here because single process consists of many thread.	Multiprogramming concept hold the concept of multi process

6) How we can create daemon threads?

Ans:-

Creating a thread as a daemon in Java is as simple as calling the setDaemon() method. A setting of true means the thread is a daemon; false means it is not. By default, all threads are created with an initial value of false.

```
 Thread daemonThread = new Thread(daemonRunner);
      daemonThread.setDaemon(true);
      daemonThread.start();
```

7) what are the wait() and sleep() method?
Ans:-
wait() - wait() method is part of java.lang.Object class. When wait() method is called, the calling thread stops its execution until notify() or notifyAll() method is invoked by some other Thread.
In java, synchronized methods and blocks allow only one thread to acquire the lock on a resources at a time. So, when wait() method is called by thread, then gives up the lock on that resource and goes to sleep until some other threads enter the same monitor and invokes to notify() or notifyAll() method.

Declaration Syntax:-
public final void wait() throws InterruptedException

sleep() - The sleep() method is a static method of Thread class and it makes the thread sleep/stop working for a specific amount of time. The sleep() method throws an InterruptedException if a thread is interrupted by other threads, that means Thread.sleep() method must be enclosed within the try and catch blocks or it must be specified with throws clause. Whenever we call the Thread.sleep() method, it can interact with the thread scheduler to put the current thread to a waiting state for a specific period of time. Once the waiting time is over, the thread changes from waiting state to runnable state.

Syntax

public static void sleep(long milliseconds)

public static void sleep(long milliseconds, int nanoseconds)