JAVA THREADS

Viva Voce on Java Threads

Q1. What is the thread?

A thread is a lightweight sub process.

Q2. What is multithreading?

Multithreading is a process of executing multiple threads simultaneously. Multithreading is used to obtain the multitasking.

Q3. Differentiate between process and thread?

There are the following differences between the process and thread.

o A Program in the execution is called the process whereas; A thread is a subset of the process

o Process have different address space in memory, while threads contain a shared address space.

o Context switching can be faster between the threads as compared to context switching between the threads.

Q4. What do you understand by inter-thread communication?

o The process of communication between synchronized threads is termed as inter-thread communication.

Q5. What is the purpose of wait() method in Java?

This method is used for inter-thread communication in Java. The java.lang.Object.wait() is used to pause the current thread.

Q6. What does join() method?

The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

Q7. Describe the purpose of sleep() method.

The sleep() method in java is used to block a thread for a particular time, which means it

pause the execution of a thread for a specific time.

Q8. What is the synchronization?

Synchronization is the capability to control the access of multiple threads to any shared resource.

Q9. What is shutdown hook?

The shutdown hook is a thread that is invoked implicitly before JVM shuts down. So we can use it to perform clean up the resource or save the state when JVM shuts down normally or abruptly.

Q10. Can we make the user thread as daemon thread if the thread is started?

No.

Q11. What about the daemon threads?

The daemon threads are the low priority threads that provide the background support and services to the user threads.

Q12. What is the difference between wait() and sleep() method?

The wait() method is defined in Object class.

The sleep() method is defined in Thread class.

The wait() method releases the lock.

The sleep() method doesn’t release the lock.

Q13. When should we interrupt a thread?

We should interrupt a thread when we want to break out the sleep or wait state of a thread. We can interrupt a thread by calling the interrupt() throwing the InterruptedException.

Q14. What is the difference between notify() and notifyAll()?

The notify() is used to unblock one waiting thread whereas notifyAll() method is used to unblock all the threads in waiting state.

Q15. What is Thread Scheduler in java?

In Java, when we create the threads, they are supervised with the help of a Thread Scheduler, which is the part of JVM. Thread scheduler is only responsible for deciding which thread

should be executed.

Q16. What is the volatile keyword in java?

Volatile keyword is used in multithreaded programming to achieve the thread safety, as a change in one volatile variable is visible to all other threads so one variable can be used by one thread at a time.

Q17. What do you understand by thread pool?

Java Thread pool represents a group of worker threads, which are waiting for the task to be allocated.

Q18. Is it possible to start a thread twice?

No.

MCQs on Java Threads

Q1. In java multi-threading, a thread can be created by

A. Extending Thread class

B. Implementing Runnable interface

C. Using both

Q2. Which method is called internally by Thread start() method?

A. execute()

B. run()

C. launch()

Q3. What is maximum thread priority in Java

A. 10

B. 12

C. 5

Q4. Which method must be implemented by a Java thread?

A. run()

B. execute()

C. start()

Q5. Execution of a java thread begins on which method call?

A. Start ()

B. Run ()

C. Execute ()

Q6. How many ways a thread can be created in Java multithreading?

A. 1

B. 2

C. 3

Q7. Which statements is/are correct

A. On calling Thread start () method a new thread get created.

B. Thread start () method call run () method internally

C. Both A and B.

Q8.Which method is used to make main thread to wait for all child threads

A. Join ()

B. Sleep ()

C. Wait ()

Q9. Daemon thread runs in

A. Background

B. Foreground

C. Both

Q10. Default value of a java thread is

A. 10

B. 1

C. 5

MCQs Answers

Q1. (C)

Q2. (B)

Q3. (A)

Q4. (A)

Q5. (A)

Q6. (B)

Q7. (C)

Q8. (A)

Q9. (A)

Q10. (C)

References:

1. Sebesta,"Concept of programming Language", Pearson Edu
2. Louden, "Programming Languages: Principles & Practices" , Cengage Learning
3. Tucker, "Programming Languages: Principles and paradigms ", Tata McGraw –Hill.
4. E Horowitz, "Programming Languages", 2nd Edition, Addison Wesley

Related Posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants

3. Parse Tree | PPL | Prof. Jayesh Umre
4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria
16. OOP in C++ | PPL
17. OOP in C# | PPL
18. OOP in Java | PPL
19. PPL: Abstraction & Encapsulation
20. PPL: Semaphores
21. PPL: Introduction to 4GL
22. PPL: Variable Initialization
23. PPL: Conditional Statements
24. PPL: Array
25. PPL: Strong Typing
26. PPL: Coroutines
27. PPL: Exception Handler in C++
28. PPL: OOP in PHP
29. PPL: Character Data Type

30. PPL: Exceptions
31. PPL: Heap based storage management
32. PPL: Primitive Data Type
33. PPL: Data types
34. Programming Environments | PPL
35. Virtual Machine | PPL
36. PPL: Local referencing environments
37. Generic Subprograms
38. Local referencing environments | PPL | Prof. Jayesh Umre
39. Generic Subprograms | PPL | Prof. Jayesh Umre
40. PPL: Loops
41. PPL: Exception Handling
42. PPL: C# Threads
43. Pointer & Reference Type | PPL
44. Scope and lifetime of variable
45. Design issues for functions
46. Parameter passing methods
47. Fundamentals of sub-programs
48. Subprograms
49. Design issues of subprogram
50. Garbage Collection
51. Issues in Language Translation
52. PPL Previous years solved papers
53. Type Checking | PPL | Prof. Jayesh Umre
54. PPL RGPV May 2018 solved paper discussion| Prof. Jayesh Umre
55. PPL Viva Voce
56. PPL RGPV June 2017 Solved paper | Prof. Jayesh Umre

57. Concurrency
58. Basic elements of Prolog
59. Introduction and overview of Logic programming
60. Application of Logic programming
61. PPL: Influences on Language Design
62. Language Evaluation Criteria PPL
63. PPL: Sequence Control & Expression
64. PPL: Programming Environments
65. PPL: Virtual Machine
66. PPL: Programming Paradigm
67. PPL: Pointer & Reference Type
68. try-catch block in C++