1. What is thread deadlock? Explain thread deadlock with an example?

**Ans**: A deadlock occurs when two threads each lock is different variables at the same time and then try to lock the variables that the other thread already locked. As a result, each thread stops executing and waits for the other thread to release the variable. Because each thread is holding the variable that the other thread wants, nothing occurs, and the threads remain deadlock.

The example of deadlock is:

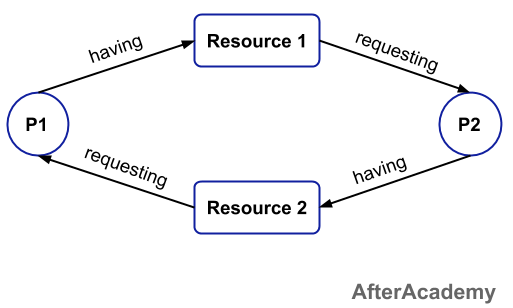


Fig: Example of deadlock

Let us assume, we have two proceese P1 and P2. Now, process P1 is holding the resource R1 and is waiting for the resource R2. At the same time , the process P2 is having the resource R2 and is waiting for the resource R1. At the same time, the process P2 is having the resource R2 and is waiting for the resource R1.  So, the process P1 is waiting for process P2 to release its resource and at the same time, the process P2 is waiting for process P1 to release its resource. And no one is releasing any resource. So, both are waiting for each other to release the resource. This leads to infinite waiting and no work is done here. This is called Deadlock.

1. How to detect a deadlock condition? How can it be avoided?

**Ans:** We can detect a deadlock by following condition:

1. First, we look and understand the code if we found nested synchronized block or trying to get a lock on a different object or calling a synchronized method from other synchronized method, these reason leads to a deadlock situation.
2. Another way to detect deadlock is to use the io portal. It allows us to upload a thread dump and analyze it.
3. We can also use J Console or Visual VM to detect deadlock. It shows us which threads are getting locked and on which object.

We can avoid the deadlock by following ways:

1. Avoid Unnecessary Locks: We should use locks only for those members on which it is required. Unnecessary use of locks leads to a deadlock situation. We recommend you to use a lock-free data structure. If possible, keep your code free form locks. For example, instead of using synchronized Array List use the Concurrent Linked Queue.
2. Avoid Nested Locks: Another way to avoid deadlock is to avoid giving a lock to multiple threads if we have already provided a lock to one thread. Since we must avoid allocating a lock to multiple threads.
3. Using Thread. Join () Method: You can get a deadlock if two threads are waiting for each other to finish indefinitely using thread join. If your thread has to wait for another thread to finish, it's always best to use join with the maximum time you want to wait for the thread to finish.
4. Use Lock Ordering: Always assign a numeric value to each lock. Before acquiring the lock with a higher numeric value, acquire the locks with a lower numeric value.
5. Lock Time-out: We can also specify the time for a thread to acquire a lock. If a thread does not acquire a lock, the thread must wait for a specific time before retrying to acquire a lock.
6. Differentiate between synchronized keyword and synchronized block

|  |  |
| --- | --- |
| Synchronized Keyword | Synchronized block |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. What is the purpose of synchronized block?

**Ans:** A synchronized block is a piece of code that can be used to perform synchronization on any specific resource of the method. A synchronize block is used to lock an object for any shared resource and the scope of a synchronized block is smaller than the synchronized method.

1. Differentiate between wait () and sleep () method

**Ans:** Differentiate between wait () and sleep () are:

|  |  |
| --- | --- |
| Wait () | Sleep () |
| Wait () belong to object class. | Sleep () belong to thread class. |
| Wait () method releases lock on object during synchronization. | Sleep () method does not release lock on object during synchronization. |
| It is not static method. | It is static method. |
| Public final void wait (long timeout) | Public static void sleep (long Millis) throws Interrupted Exception |
| Wait () has three overloaded methods:   * 1. Wait ()   2. Wait (long timeout)   3. Wait (long timeout, int nanos) | Sleep () has two overloaded methods:   * 1. Sleep (long Millis)   2. Sleep (long Millis, int nanos) |

String

1. Differentiate between String buffer and string builder

**Ans:**

|  |  |
| --- | --- |
| Sting Buffer | String Builder |
| In string buffer Thread is safe. | In string builder Thread is not safe. |
| String Buffer is synchronized. It means two threads can't call the methods of String Buffer simultaneously. | StringBuilder is non-synchronized. It means two threads can call the methods of StringBuilder simultaneously. |
| String buffer is less efficient than string builder. | String builder is more efficient than string buffer. |
| String Buffer was introduced in Java 1.0 | String Builder is introduced in java 1.5 |

1. Why Char array is preferred over string for string password?

**Ans:** Char array is preferred over string for string password because of following ways:

1. Strings are immutable: Strings are immutable in java and therefore if a password is stored as plain text, it will be available in memory until garbage collector clears it and as string are used in the string pool for re-usability there are high chances that it will remain in memory for long duration, which is security thread. String is immutable and there is no way that the contain of string can be changed because any change will produce new string. Within an array the data can be wiped explicitly after its work is completed. The array can be overwritten and the password won’t be present anywhere in the system, even before garbage collection.
2. Security: Anyone who has access to memory dump can find the password in clear text and that’s another reason to use encrypted password that plain text. So, storing password in character array clearly mitigates security risk of stealing password.
3. **Log files safety: With an array, one can explicitly wipe the data, overwrite the array and the password won’t be present anywhere in the system. With plain string, there are much higher chances of accidently printing the password to logs, monitors or some other insecure place. Char () is less vulnerable.**
4. Why string is immutable in java?

**Ans:** A String is an unavoidable type of variable while writing any application program. String references are used to store various attributes like username, password, etc. In Java, **String objects are immutable**. Immutable simply means unmodifiable or unchangeable. Once String object is created its data or state can't be changed but a new String object is created.

Exception handling

1. How do you invite a custom in java?

**Ans:**

1. What are the rules that we need to follow when overriding a method that throws an exception?

**Ans:** The rules that we need to follow when overriding a method that throws an exception are:

1. When the parent class method doesn’t throw any exceptions, the child class method can’t throw any checked exception, but it may throw any unchecked exceptions.
2. When the parent class method throws one or more checked exceptions, the child class method can throw any unchecked exception.
3. When the parent class method has a throws clause with an unchecked exception, the child class method can throw none or any number of unchecked exceptions, even though they are not related.
4. Differentiate between final finally and finalize keyword in java.

**Ans:** Differentiate between final finally and finalize keyword:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |