Waiting of Child Thread until Completing Main Thread

We can make the main thread wait for the child thread as well as we can make the child thread also wait for the main thread.

sleep()

If a thread doesn't want to perform any operation for a particular amount of time then we should go to sleep().

Signature

```
public static native void sleep(long ms) throws InterruptedException
public static void sleep(long ms,int ns) throws InterruptedException
```

Every sleep method throws InterruptedException, which is a checked exception so we should compulsorily handle the exception using
try catch or by throws keyword otherwise it would result in a compile time error.

```
Thread t=new Thread(); //new or born state
 t.start() // ready/runnable state
```

=> If T.S allocates cpu time then it would enter into running state.
=> If run() completes then it would enter a dead state.
=> If running thread invokes sleep(1000)/sleep(1000,100) then it would enter into Sleeping state
=> If time expires/ if sleeping thread got interrupted then thread would come back to "ready/runnable state".

synchronization

1. synchronized is a keyword applicable only for methods and blocks
2. if we declare a method/block as synchronized then at a time only one thread can execute that method/ block on that object.
3. The main advantage of synchronized keywords is we can resolve data inconsistency problems.
4. But the main disadvantage of synchronized keyword is it increases waiting time of the Thread and effects performance of the system.
5. Hence if there is no specific requirement then never recommended to use synchronized keyword.
6. Internally synchronization concept is implemented by using lock concept.
7. Every object in java has a unique lock. Whenever we are using synchronized keyword then only the lock concept will come into the picture.
8. If a Thread wants to execute any synchronized method on the given object 1st it has to get the lock of that object. Once a Thread gets the lock of that object then it's allowed to execute any synchronized method on that object. If the synchronized method execution completes then automatically Thread releases lock.
9. While a Thread executing any synchronized method the remaining Threads are not allowed to execute any synchronized method on that object simultaneously. But remaining Threads are allowed to execute any non-synchronized method simultaneously. [lock concept is implemented based on object but not based on method].

Note: Every object will have 2 area[Synchronized area and NonSynchronized area]
Synchronized Area => write the code only to perform update,insert,delete
NonSynchronized Area => write the code only to perform select operation

```
class ReservationApp{
 checkAvailablity(){
  //perform read operation
        }
 synchronized bookTicket(){
  //perform update operation
 }
}
```

class level lock

1. Every class in java has a unique level lock.
2. If a thread wants to execute static synchronized method then the thread requires "class level lock".
3. While a Thread executing any static synchronized method the remaining Threads are not allowed to execute any static synchronized method of that class simultaneously.
4. But remaining Threads are allowed to execute normal synchronized methods, normal static methods, and normal instance methods simultaneously.
5. Class level lock and object lock both are different and there is no relationship between these two.

synchronized block

```
synchronized void m1(){
 …
 …
 …
 …
 …
 ======
 ======
 ======
 ======

 …
 …
 …
 …
 …
}
```

if a few lines of code is required to get synchronized then it is not recommended to make the method only as synchronized.
If we do this then for threads performance will be low, to resolve this problem we use "synchronized block", due to synchronized block performance will be improved.

DeadLock

If 2 Threads are waiting for each other forever(without end) such type of situation (infinite waiting) is called dead lock.
There are no resolution techniques for deadlock but several prevention(avoidance) techniques are possible.
Synchronized keyword are the cause for deadlock hence whenever we are using synchronized keyword we have to take special care.