Writing a class inside a class, this concept is known as inner Classes.

We are having 4 types of inner Classes

1. Nested inner class
2. Static inner class
3. Method local inner class
4. Anonymous inner class

```java
public class ClassA
{
    class InnerClassA // Nested InnerClass
    {

    }
    static class InnerClassB// static InnerClass
    {

    }
    void meth1()
    {
        class InnerClassC // Method Local InnerClass
        {

        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA()
        {       //Anononmyous InnerClass

        }
        ;
    }
}
```

Anonymous inner class will be acting like a child class for your class

```java
package com.pack1;

public class ClassA
{
    void meth1()
    {
        System.out.println("meth1() called");
    }

    public static void main(String[] args)
    {
        ClassA aobj=new ClassA()
        {       //Anononmyous InnerClass
            @Override
            void meth1()
            {
                System.out.println("Java is awesome");
            }
        }
        ;
        aobj.meth1();

    }
}
```

```
Java is awesome
```

```java
package com.pack1;

public class ClassA
{
    void meth1()
    {
        System.out.println("meth1() called");
    }

    public static void main(String[] args)
    {
        ClassA aobj=new ClassA()
        {       //Anononmyous InnerClass
            @Override
            void meth1()
            {
                System.out.println("Java is awesome");
                super.meth1();
            }
        }
        ;
        aobj.meth1();

    }
}
```

```
Java is awesome
meth1() called
```

To increase the readability of the code we are using inner classes

```
ClassA.java   ClassB.java   ClassD.java
Training ▸ src ▸ com.pack1 ▸ ClassD ▸ main(String[]) : void ▸ new Thread() {...} ▸ run() : void
        {
            @Override
            public void run()
            {
                synchronized (B)// Thread 2 is holding Python
                {
                    System.out.println("Thread 2 locked on B");
                    try
                    {
                        Thread.sleep(100);
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                    synchronized (A)// Thread2 will be waiting fo
                    {
                        System.out.println("Thread 2 locked on A"
                    }
                }
                System.out.println("no dead lock");
            }
        }
        ;
        t1.start();
```

```
Console
ClassD [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (
Thread 2 locked on B
Thread 1 locked on A
```

Program execution will not be completed

Interthread Communication:

- Two Threads can communicate with each other by using wait(), notify() and notifyAll() methods.

Method Name	Description
public final void wait()	Causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() , or a specified amount of time has elapsed.
public final native void notify()	Wakes up a single thread that is waiting
public final void notifyAll()	Wakes up all threads that are waiting

```java
public class ClassF
{
    int amount=10000;

    synchronized void with_draw(int amount) throws InterruptedException
    {
        if(this.amount<amount)
        {
            System.out.println("Insufficient balance");

            //wait(30000);//30 sec
            //Thread.sleep(30000); // 30 sec

            System.out.println("Amount credited");
            this.amount-=amount;
            System.out.println("with draw successful \nBalance is "+this.amount);
        }
        else
        {
            this.amount-=amount;
            System.out.println("with draw successful \nBalance is "+this.amount);
        }
    }
    synchronized void deposit(int amount)
    {
        this.amount+=amount;
        System.out.println("Deposited successfully \nBalance is "+this.amount);
        //notify();
        //notifyAll();
```

```
    }
}
```

```
public class ClassQ
{
    public static void main(String[] args)
    {
        ClassF fobj=new ClassF();

        new Thread()// first Thread
        {  // Anonymous Inner Class Starts here
            @Override
            public void run()
            {
                try
                {
                    fobj.with_draw(20000);
                }
                catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
            }
        } // Anonymous Inner Class Ends here

        .start();

        new Thread() // Second Thread
        { // Anonymous Inner Class Starts here
            @Override
            public void run()
            {
                fobj.deposit(90000);
            }
        } // Anonymous Inner Class Ends here
        .start();
    }
}
```

There is a huge difference between wait() and sleep()

wait() releases the lock on the thread, here processor is free to execute other threads.

sleep() does not release the lock on the thread, here processor is not free it will stick with the same thread.

```
8       {
9           if(this.amount<amount)
10          {
11              System.out.println("Insufficient balance");
12
13              //wait(30000);
14              Thread.sleep(30000); // 30 sec
15
16              System.out.println("Amount credited");
17              this.amount-=amount;
18              System.out.println("with draw successful \nBalance is "+this.amount);
19          }
20          else
21          {
22              this.amount-=amount;
23              System.out.println("with draw successful \nBalance is "+this.amount);
24          }
25      }
26      synchronized void deposit(int amount)
27      {
28          this.amount+=amount;
29          System.out.println("Deposited successfully \nBalance is "+this.amount);
30          //notify();
31          //notifyAll();
32      }
```

Line: 27

```
<terminated> ClassQ [Java Application] C:
Insufficient balance
Amount credited
with draw successful
Balance is -10000
Deposited successfully
Balance is 80000
```

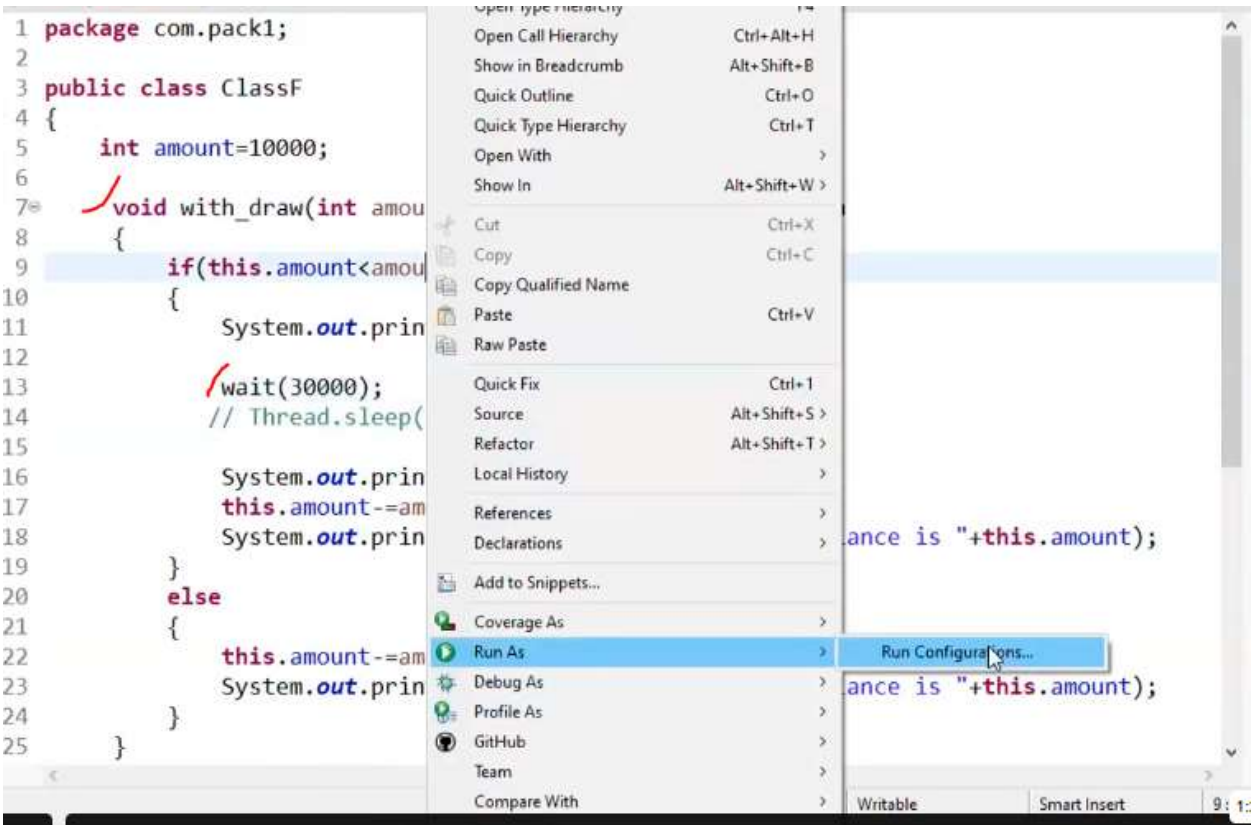
```
8       {
9           if(this.amount<amount)
10          {
11              System.out.println("Insufficient balance");
12
13              wait(30000);
14              //Thread.sleep(30000); // 30 sec
15
16              System.out.println("Amount credited");
17              this.amount-=amount;
18              System.out.println("with draw successful \nBalance is "+this.amount);
19          }
20          else
21          {
22              this.amount-=amount;
23              System.out.println("with draw successful \nBalance is "+this.amount);
24          }
25      }
26      synchronized void deposit(int amount)
27      {
28          this.amount+=amount;
29          System.out.println("Deposited successfully \nBalance is "+this.amount);
30          //notify();
31          //notifyAll();
32      }
```

```
<terminated> ClassQ [Java Application] C:
Insufficient balance
Deposited successfully
Balance is 100000
Amount credited
with draw successful
Balance is 80000
```

wait() is used only in synchronized areas.

wait() is present in object class which is of three versions.

sleep() is present in thread class which is two versions, and it is a static method.



```
ClassF.java    ClassQ.java
16                  fol
17                }
18                catch
19                {
20                    e.
21                }
22            }
23        } // Anonymous
24        .start();
25
26        new Thread() /
27        { // Anonymous
28            @Override
29            public voi
30            {
31                fobj.d
32            }
33        } // Anonymous
34        .start();
35
36    }
37 }
38
39
40
```

```
Console
<terminated> ClassQ [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (25-Apr-2025, 8:52:36 am – 8:52:36 am) [pid: 13064]
Insufficient balance
Exception in thread "Thread-0" java.lang.IllegalMonitorStateException: current thread is not owner
        at java.base/java.lang.Object.wait(Native Method)
        at Training/com.pack1.ClassF.with_draw(ClassF.java:13)
        at Training/com.pack1.ClassQ$1.run(ClassQ.java:16)
Deposited successfully
Balance is 100000
```

Difference between wait() & sleep()

wait()	sleep()
wait() method releases the lock	sleep() method doesn't release the lock.
It is the method of java.lang.Object class	It is the method of java.lang.Thread class
It is a non-static method	It is a static method
wait() should be notified by notify() or notifyAll() methods.	After the specified amount of time, sleep() is completed
wait() method must be called from synchronized context (i.e. synchronized method or block), otherwise it will throw IllegalMonitorStateException	sleep() can becalled from anywhere. there is no specific requirement.