**Advanced Java: Multi-threading Part 8 - Wait and Notify**

https://www.youtube.com/watch?v=gx\_YUORX5vk&list=PLBB24CFB073F1048E&index=8

**2nd class – secondclass.java:**

**import** java.util.Scanner;

**public** **class** secondclass{

**public** **void** produce() **throws** InterruptedException{

**synchronized** (**this**){

System.*out*.println("Producer thread running...");

wait();

System.*out*.println("Resumed.");

}

}

**public** **void** consume() **throws** InterruptedException{

Scanner scanner = **new** Scanner(System.*in*);

Thread.*sleep*(2000);

**synchronized**(**this**){

System.*out*.println("Waiting for return key.");

scanner.nextLine();

System.*out*.println("Return key pressed.");

notify();

Thread.*sleep*(5000);

}

}

}

**1st class – firstclass.java:**

**public** **class** apples {

**public** **static** **void** main(String args[]) **throws** InterruptedException{

**final** secondclass processor = **new** secondclass();

Thread t1 = **new** Thread(**new** Runnable(){

**public** **void** run(){

**try** {

processor.produce();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

});

Thread t2 = **new** Thread(**new** Runnable(){

**public** **void** run(){

**try** {

processor.consume();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

});

t1.start();

t2.start();

t1.join();

t2.join();

}

}

**Result:**

Producer thread running...

Waiting for return key.

Return key pressed. 5 seconds passes

Resumed.

**Important notes:**

* *wait()* method sleeps until *notify()* method is called.
* *notify()* method works only in **synchronized** blocks.
* To illustrate notify() method don’t give away lock to the thread with the wait method(), the programmer made the second thread sleep for 5 seconds.

Keep in mind that:

public void blah() {

synchronized (this) {

// do stuff

}

}

is semantically equivalent to:

public synchronized void blah() {

// do stuff

}