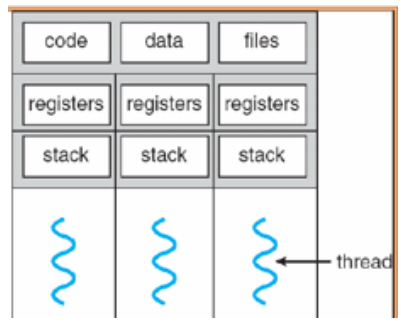
Αντικειμενοστρεφής Προγραμματισμός (Object-Oriented Programming)

(CEID_NNY106)

Ταυτόχρονος Προγραμματισμός

Concurrent Programming

Κύρια Πηγή



Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

Java

High-level programming language

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. Wikipedia

Designed by: James Gosling

First appeared: May 23, 1995; 27 years ago

Paradigm: Multi-paradigm: generic, object-oriented (class-based), functional, imperative, reflective, concurrent

Concurrent programming is difficult

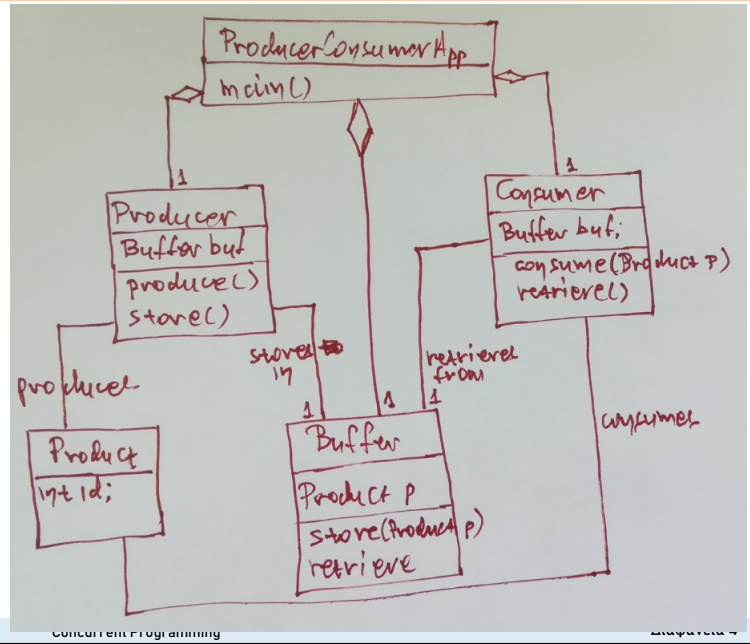
- "It is widely acknowledged that concurrent programming is difficult. Yet the imperative for concurrent programming is becoming more urgent." *Edward A. Lee, IEEE Computer 2006.*
- "humans are quickly overwhelmed by concurrency and find it much more difficult to reason about concurrent than sequential code. Even careful people miss possible interleavings among even simple collections of partially ordered operations." *Sutter and Larus, ACM Queue.*

Οργάνωση Διάλεξης

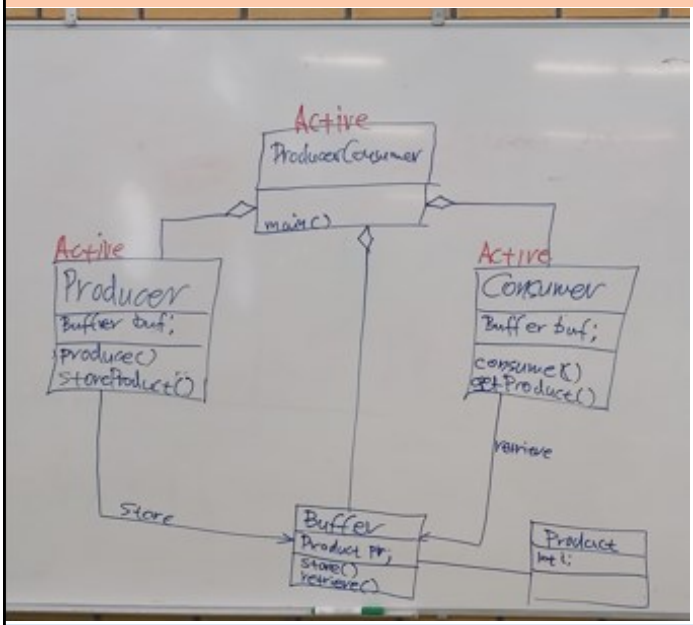
- **Producer/Consumer**
- Η κλάση Thread
- PingPongApp
- Κύκλος ζωής Νήματος
  - Thread Priorities/ Thread Scheduling
- Runnable Interface
- CounterTest Example

Producer-Consumer

- Αποτελεί την **αφαιρετική διατύπωση ενός συνόλου προβλημάτων**
- Ο παραγωγός παράγει αγαθά τα οποία αποθηκεύει μέχρι να είναι έτοιμος ο καταναλωτής να τα καταναλώσει.
- Ο καταναλωτής παίρνει αγαθά από την αποθήκη και τα καταναλώνει.



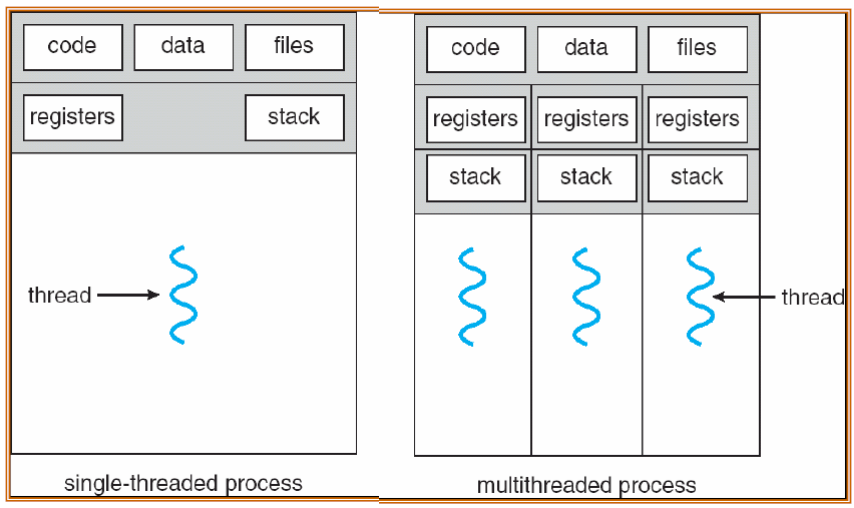
Producer/Consumer - 1st Draft Implementation




```
class ProducerConsumer {
  static Producer p;
  static Consumer c;
  static Buffer buf;
```

```
main() {
  buf = new Buffer();
  p = new Producer(buf);
  c = new Consumer(buf);
  for(int i; i<100; i++){
    var pr = p.produce();
    p.storeProduct(pr);
    pr = c.getProduct();
    c.consume(pr);
  }
}
```


Process and Thread



Οργάνωση Διάλεξης

- Producer/Consumer
- **Η κλάση Thread**
- PingPongApp
- Κύκλος ζωής Νήματος
  - Thread Priorities/ Thread Scheduling
- Runnable Interface
- CounterTest Example

Threads in Java

- A **Thread** is a thread of execution in a program
- Η VJM επιτρέπει σε μία εφαρμογή να έχει πολλά νήματα που εκτελούνται παράλληλα
- Η Java υποστηρίζει τα νήματα με
  - ειδικές κατασκευές
    - κλάσεις της βασικής βιβλιοθήκης
    - interfaces της βασικής βιβλιοθήκης

java.lang

Class Thread

java.lang.Object
    java.lang.Thread

All Implemented Interfaces:

Runnable

java.lang

Interface Runnable

java.util.concurrent

Interface Executor

Με την εκκίνηση της VJM υπάρχει ένα non-deamon thread που καλεί την main

Κλάση Thread

Δημιουργοί

- public Thread()
  - Το σύστημα δίνει ονόματα Thread-1, Thread-2, ...
- public Thread(String threadName)
  - Ο προγραμματιστής ορίζει το όνομα του Thread

Παράδειγμα

Thread ping = new Thread(new String("ping"));

```
public class Thread
extends Object
implements Runnable
```

Constructor and Description
`Thread()` Allocates a new Thread object.
`Thread(Runnable target)` Allocates a new Thread object.
`Thread(Runnable target, String name)` Allocates a new Thread object.
`Thread(String name)` Allocates a new Thread object.
`Thread(ThreadGroup group, Runnable targ` Allocates a new Thread object.
`Thread(ThreadGroup group, Runnable targ` Allocates a new Thread object so that it has ta to by group.
`Thread(ThreadGroup group, Runnable targ` Allocates a new Thread object so that it has ta to by group, and has the specified *stack size*.
`Thread(ThreadGroup group, String name)` Allocates a new Thread object.

Κλάση Thread – Methods

- **run** - *edit-time*
  - προσδιορίζει το έργο του νήματος
  - Subclasses of Thread should override this method (Ορίζεται από τον προγραμματιστή).
  - Καλείται από το σύστημα σαν αποτέλεσμα της κλήσης της start
  - αν κληθεί άμεσα δεν υποστηρίζεται multithreading

`run()`
If this thread was constructed using a separate `Runnable` run object, then that `Runnable` object's run method is called; otherwise, this method does nothing and returns.

- **start** - *run-time*
  - Έχει ως αποτέλεσμα την ενεργοποίηση του νήματος
  - Η JVM καλεί την μέθοδο run του νήματος.
  - Ως αποτέλεσμα 2 νήματα είναι ενεργά ταυτόχρονα:
    - το νήμα που καλεί την start, για το οποίο ο έλεγχος επιστρέφει στο σημείο κλήσης της start, και
    - το νέο που δημιουργήθηκε, το οποίο εκτελεί την μέθοδο run του

`void`	`start()` Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.

Μέθοδοι κλάσης Thread

start - δημιουργεί νέο νήμα εκτέλεσης, ενεργοποιεί την run
interrupt - έχει σαν αποτέλεσμα τη διακοπή ενός νήματος εκτέλεσης.
checkAccess() Determines if the currently running thread has permission to modify this thread.
interrupted – επιστρέφει true αν το νήμα είναι σε διακοπή αλλιώς επιστρέφει false.

suspend - αναστέλλει την λειτουργία ενός νήματος εκτέλεσης *(Deprecated)*
resume - (called by another thread) ξαναρχίζει την εκτέλεση του νήματος *(Deprecated)*
stop - τερματίζει την εκτέλεση ενός νήματος. *(Deprecated)*

isAlive - επιστρέφει true εάν έχει κληθεί η start για ένα συγκεκριμένο νήμα και το νήμα δεν έχει τερματιστεί.

join Waits for this thread to die.

static void **yield**()

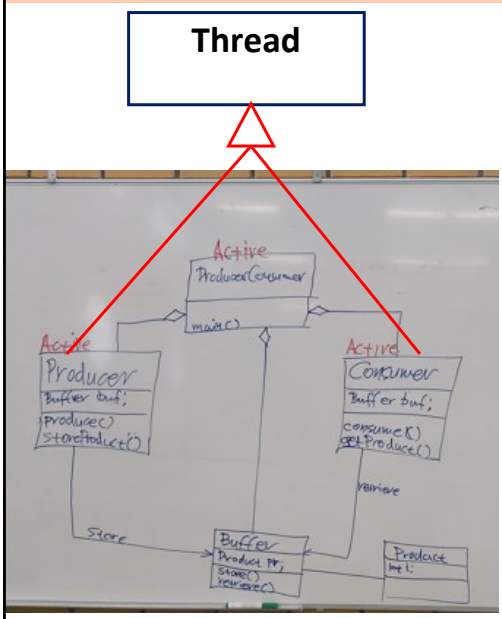
Causes the currently executing thread object to temporarily pause and allow other threads to execute.

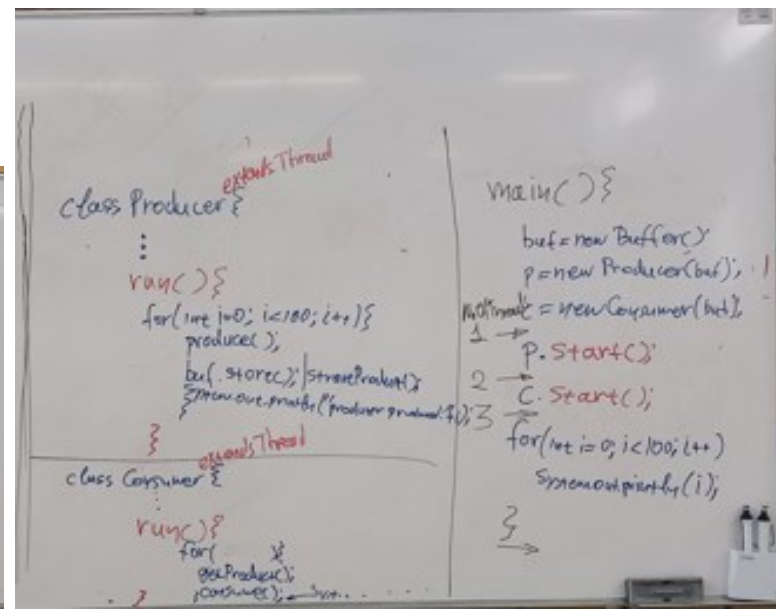
static **currentThread** επιστρέφει μια αναφορά στο τρέχον νήμα

setName, getName και **setPriority**, **getPriority**

παρέχουν πρόσβαση στο όνομα και στην προτεραιότητα του νήματος

Producer/Consumer with Active objects





Producer-Consumer – 1η προσπάθεια υλοποίησης

```
class Buffer {
        int i=0;

        void store(int i) {
                this.i = i;
                }

        int retrieve(){
                return i;
                }
        }
```

Απαιτήσεις

- Ο παραγωγός δεν πρέπει να βάζει στην αποθήκη προϊόν αν δεν έχει καταναλωθεί το προηγούμενο (product overwrite).
- Ο καταναλωτής δεν πρέπει να καταναλώνει αγαθό που δεν είναι διαθέσιμο (αγαθό που έχει ήδη καταναλωθεί)

Οργάνωση Διάλεξης

- Producer/Consumer
- Η κλάση Thread
- **PingPongApp**
- Κύκλος ζωής Νήματος
  - Thread Priorities/ Thread Scheduling
- Runnable Interface
- CounterTest Example

PingPong App

```
class PingPongApp {
   public static void main(String [] args) {
      PingPong ping=new PingPong("PING",2);
      PingPong  pong = new PingPong("pong",4);

      ping.start();
      pong.start();

      for(int i=0;i<100;i++)
         System.out.print(i + " ");
      ping.stop();
      pong.stop();
      }
   }
```

```
class PingPong extends Thread {
    String word;              // λέξη για εκτύπωση
    int delay;                // χρόνος ανάπαυσης

    PingPong(String wordToPrint, int pauseTime){
          word = wordToPrint;
          delay = pauseTime;
          }
    public void run() {
          try {
             for(int i=0;;i++){
                   System.out.print(word +"-" + i + " ");
                   sleep(delay);
                   }
          }
          catch (InterruptedException e){
                   return;    // τερματισμός του thread
                   }
          }
    }
```


Μέθοδος sleep

public static void **sleep**(long millis)
throws InterruptedException

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds. The thread does not lose ownership of any monitors.

Parameters: millis
- the length of time to sleep in milliseconds.

Throws: InterruptedException
- if another thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

PingPong App – Μία πιθανή έξοδος

PING-0 ***pong-0*** 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 ***pong-1*** **PING-1** 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 ***pong-2*** **PING-2** ***pong-3*** 63 **PING-3** ***pong-4*** **PING-4** 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 ***pong-5*** **PING-5** ***pong-6*** 83 84 85 86 87 88 89 90 91 92 93 94 95 **PING-6** ***pong-7*** 96 97 98 99