Programming threads

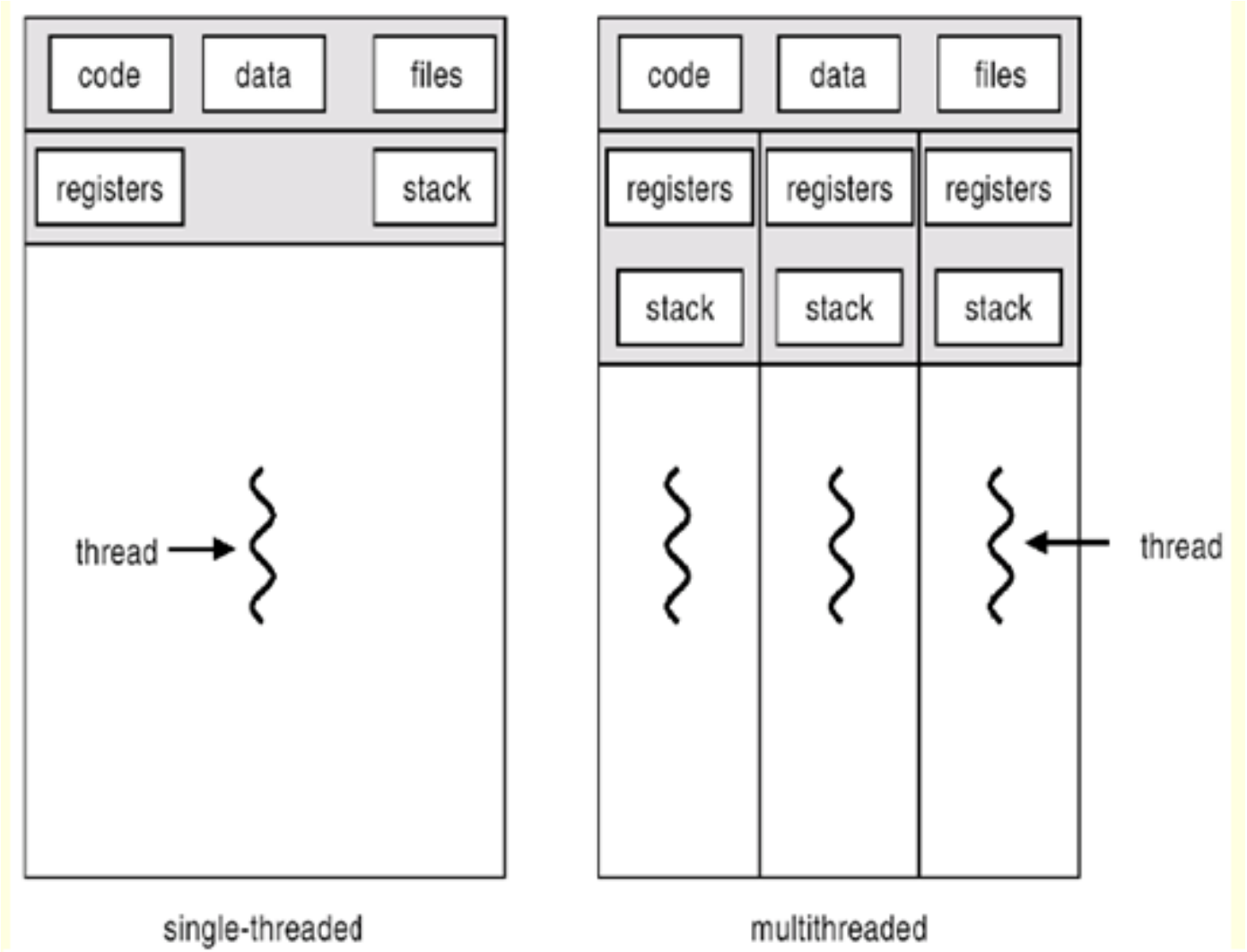
Διεργασίες

- Μια **διεργασία** (**process**) είναι ένα αυτοδύναμο περιβάλλον εκτέλεσης προγράμματος.
- Διαθέτει τους δικούς της αποκλειστικούς πόρους κατά το χρόνο εκτέλεσής της (με πιο σημαντικό αυτό της μνήμης).
- Μια εφαρμογή είναι δυνατό να αποτελείται από πολλές συνεργαζόμενες διεργασίες.

Νήματα

- Ένα **νήμα** (**thread**) είναι μια ακολουθιακή ροή ελέγχου (δηλαδή, κώδικας που έχει αρχή, περιέχει μια ακολουθία εντολών και έχει τέλος) σε ένα πρόγραμμα.
- Σκοπός είναι η απομόνωση (ή αυτονόμηση) κάποιων εργασιών (tasks), ώστε να μπορούν να εκτελούνται ταυτόχρονα (δυνητικά παράλληλα).
- Ένα νήμα δεν είναι από μόνο του ένα πρόγραμμα, αλλά ούτε διεργασία.
- Ένα νήμα εκτελείται στο εσωτερικό ενός προγράμματος.
- Ένα μόνο νήμα δεν προσφέρει ευκαιρίες πολυεπεξεργασίας.
- Δύο ή περισσότερα νήματα μπορούν να εκτελούνται ταυτόχρονα πραγματοποιώντας διαφορετικές εργασίες.

Πολυνηματικές εφαρμογές



Διεργασίες έναντι Νημάτων

- Οι διεργασίες :
  - είναι ανεξάρτητες
  - κουβαλούν σημαντική πληροφορία κατάστασης (process control block)
  - διαθέτουν ξεχωριστό χώρο διευθύνσεων (address space)
  - και αλληλεπιδρούν μόνον μέσω μηχανισμών διαδιεργασιακής επικοινωνίας που παρέχονται από το ΛΣ
- Τα νήματα:
  - Μοιράζονται πληροφορίες κατάστασης μιας διεργασίας
  - Μοιράζονται μνήμη και άλλους πόρους απ' ευθείας
- Το context-switching μεταξύ νημάτων της ίδιας διεργασίας είναι τυπικά ταχύτερο (1:10!!) από το context-switching μεταξύ διεργασιών → τα νήματα είναι πιο αποδοτικά

Δημιουργία νημάτων στην Java

- Δύο τρόποι δημιουργίας νημάτων:
  - Ως στιγμιότυπα υποκλάσεων της κλάσης **`java.lang.Thread`**:
    + Πιο απλή προσέγγιση, κατάλληλη για απλές εφαρμογές.
    – Τα νήματα περιορίζονται στο να είναι υποκλάσεις της συγκεκριμένης κλάσης.
  - Ως στιγμιότυπα υλοποιήσεων του interface **`java.lang.Runnable`**:
    + Η κλάση που υλοποιεί τη διεπαφή μπορεί να είναι υποκλάση οποιασδήποτε άλλης κλάσης.
- Και στις δύο περιπτώσεις πρέπει να οριστεί το σώμα της μεθόδου **`run`**, που είναι κενό και προσδιορίζει τι κάνει το νήμα.
- Η μέθοδος **`run`** είναι αφαίρεση μεθόδου της **`Runnable`**, αλλά και της **`Thread`** (καθώς η **`Thread`** υλοποιεί την **`Runnable`**).

Επεκτείνοντας την κλάση Thread

- Δημιουργία υποκλάσης της `Thread`

```
public class ExampleThread extends Thread
{
  public void run()
  {
    // perform whatever thread needs to do
  }
}
```

- Για να ξεκινήσει ένα νέο thread

```
ExampleThread et = new ExampleThread();
et.start();
```

Η start() ορίζεται στην Thread κλάση

Υλοποιώντας το interface Runnable

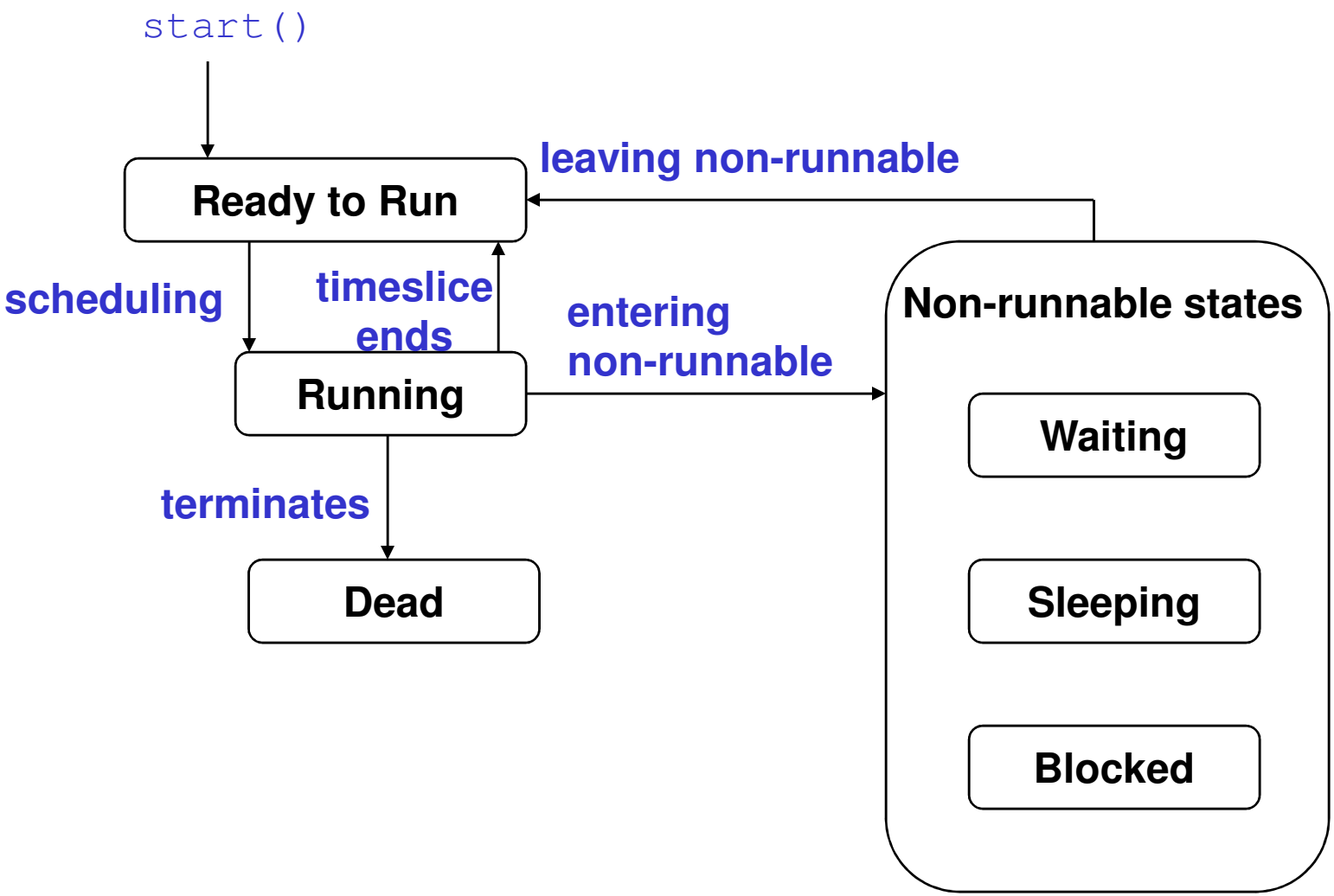
- Δημιουργία μιας κλάσης που υλοποιεί το `Runnable` interface

```
public class ExampleRunnable implements Runnable
{
  public void run()
  {
    // perform whatever thread needs to do
  }
}
```

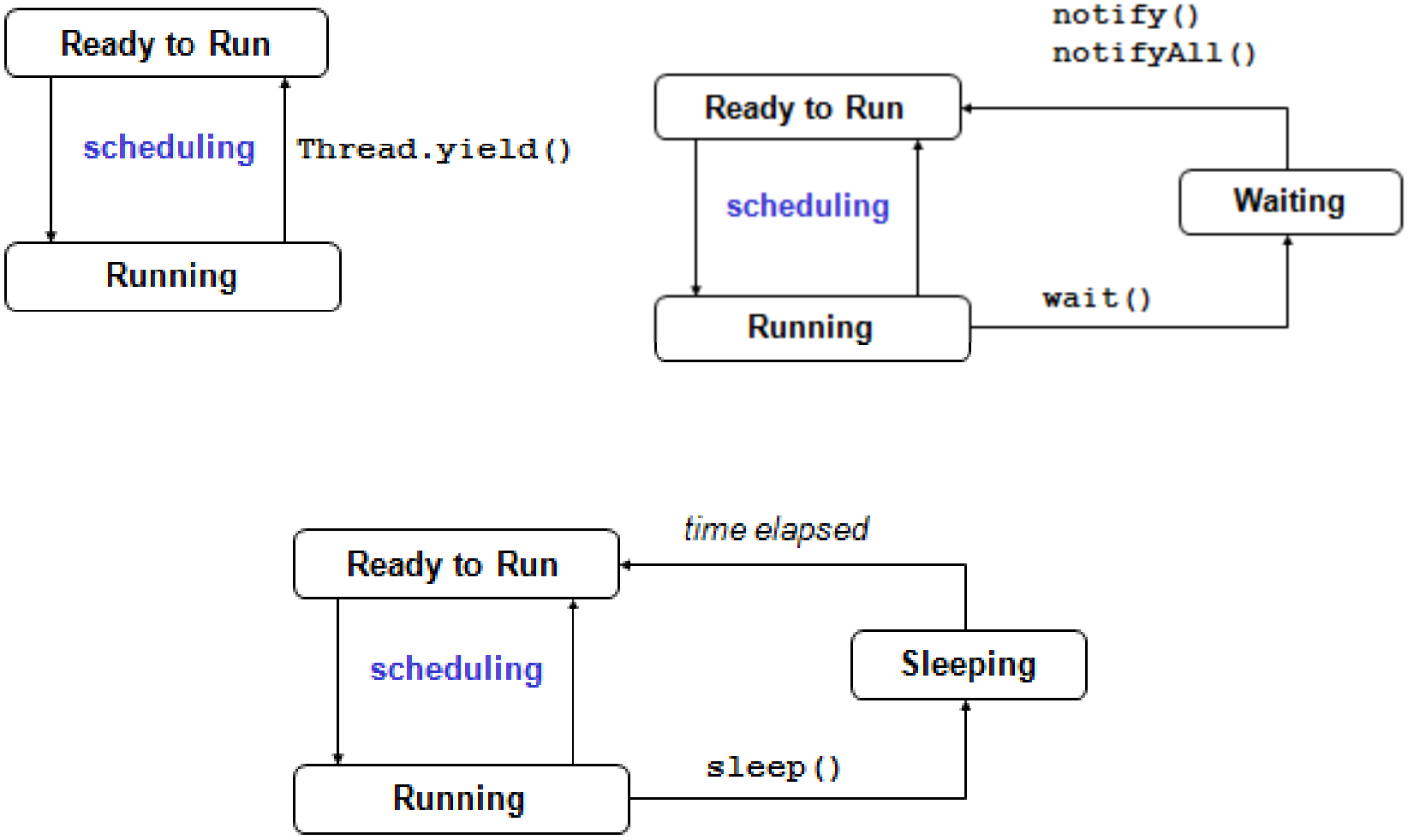
- Για να ξεκινήσει ένα νέο thread

```
ExampleRunnable er = new ExampleRunnable();
new Thread(er).start();
```

Καταστάσεις νημάτων



Μεταβάσεις από κλήση μεθόδων



Συγχρονισμός νημάτων

- Τα νήματα μοιράζονται τον ίδιο χώρο μνήμης
  - κάθε νήμα διαθέτει το δικό του stack space,
  - αλλά μοιράζονται το ίδιο heap (για διαχείριση δυναμικής μνήμης)

- Ο συγχρονισμός είναι απαραίτητος για :
  - ακεραιότητα δεδομένων
  - προσπέλαση κοινών πόρων

Monitor: μηχανισμός συγχρονισμού

- Μια περιοχή του κώδικα που αναπαριστά έναν κοινό πόρο (κρίσιμη περιοχή) μπορεί να συσχετισθεί με ένα monitor (αλλιώς semaphore).
- Τα νήματα κάνουν προσπέλαση του κοινού πόρου κλειδώνοντας πρώτα το monitor
- Μόνον ένα νήμα μπορεί να κατέχει το monitor σε μια δεδομένη χρονική στιγμή
- Τα monitors υλοποιούν έναν μηχανισμό αμοιβαίου αποκλεισμού (αλλιώς mutex)

Συνθήκες ανταγωνισμού (race conditions)

- **Συνθήκη ανταγωνισμού** (race condition) έχουμε όταν δύο νήματα προσπαθούν να προσπελάσουν το ίδιο αντικείμενο ταυτόχρονα και η συμπεριφορά του προγράμματος δεν εξαρτάται πλέον μόνο από τις εντολές του, αλλά και από το ποιο νήμα προηγείται στη σειρά προσπέλασης (δεν είναι προβλέψιμο αφού αυτό εξαρτάται από το λειτουργικό σύστημα).

- Αν όλα τα νήματα εκτελούν αποκλειστικά λειτουργίες ανάγνωσης, τότε δεν υπάρχει κίνδυνος συνθήκης ανταγωνισμού.

- Στην περίπτωση συνθήκης ανταγωνισμού τα αποτελέσματα των υπολογισμών δεν είναι συνεπή σε διαδοχικές επαναλήψεις εκτέλεσης του προγράμματος με την ίδια είσοδο.

Συνθήκες ανταγωνισμού (race conditions)

- Πρέπει να εξασφαλιστεί ότι το ένα νήμα δεν θα αλλάξει τα δεδομένα αυτά, ενώ τα διαχειρίζεται κάποιο άλλο νήμα

  ➔ αναγκαίος ο συγχρονισμός νημάτων.

- Για το λόγο αυτό, η Java επιτρέπει το κλείδωμα αντικειμένων και μεθόδων με τη χρήση της δεσμευμένης λέξης synchronized.

Συγχρονισμός νημάτων

- Συγχρονισμός νημάτων επιτρέποντας την προσπέλαση μιας μεθόδου (ή ενός αντικειμένου) σε ένα μόνο νήμα κάθε φορά:
  - **`public synchronized void xmethod {…}`**
  - Για να προσπελάσει ένα νήμα τη μέθοδο **`xmethod`** πρέπει να περιμένει να τελειώσει το τρέχον νήμα (blocked).
- Μπορεί να δηλωθεί ως συγχρονισμένο ένα τμήμα κώδικα στο σώμα μιας μεθόδου (και όχι όλη η μέθοδος):

```
public int ymethod {
     ...
     synchronized (this) {
          ...                }
}
```

Αδιέξοδα (Deadlocks)

- Μπορούν να υπάρξουν καταστάσεις όπου ένα νήμα περιμένει κάποια απάντηση από ένα άλλο, που με την σειρά του μπορεί να περιμένει κάτι από το πρώτο ➔ ***Αδιέξοδο*** (deadlock).

- Η Java δεν ανιχνεύει και δεν επιλύει τα αδιέξοδα.

- Ο προγραμματιστής είναι υπεύθυνος γι' αυτό.