Programmation Réseau

Système d'exécution Java

université PARIS DIDEROT PARIS 7

- processus
  programme en cours
  d’exécution
- thread fil d’exécution
- multi-threadé

-
  - point courant d'exécution
    program counter
  - pile d'exécution stack
  - Thread
    Thread

- java.lang.Runtime
  - Runtime getRuntime();

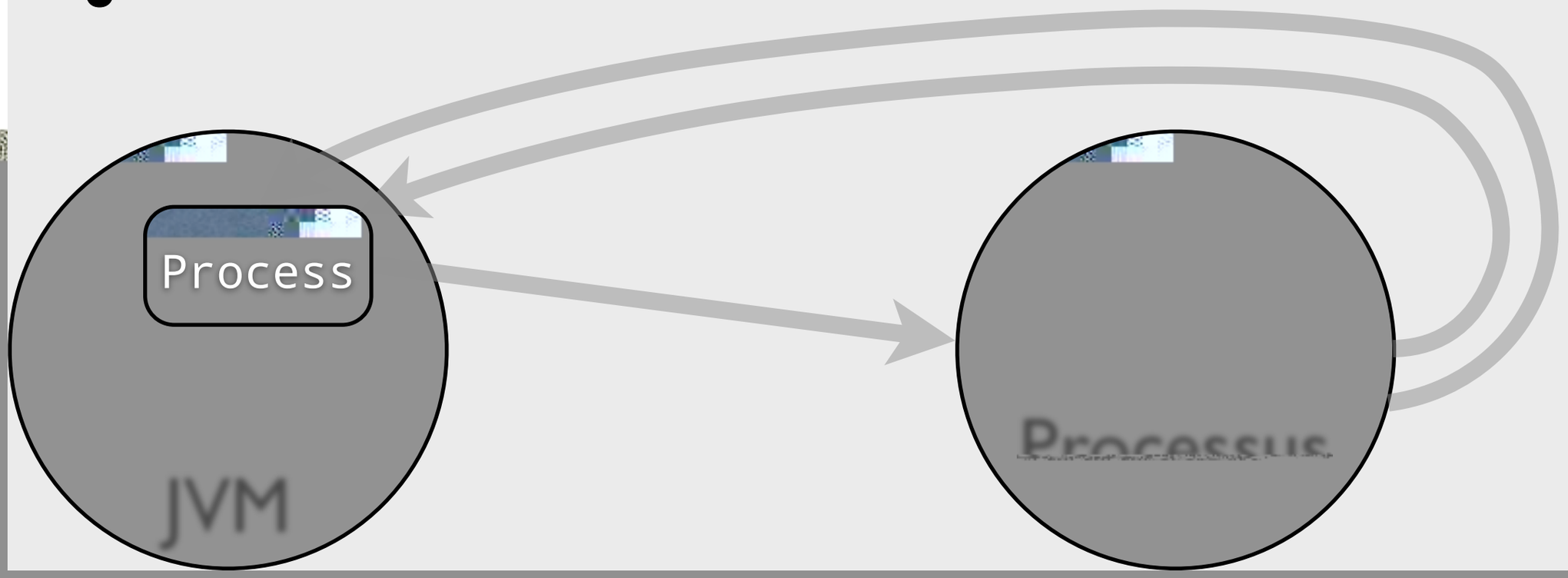
- Runtime
-
  Process e ec( );
-

- `java.lang.Process`

-

```
InputStream getErrorStream();
InputStream getInputStream();
OutputStream getOutputStream();
```

-



```
int e itValue();
int  aitFor();
```

```
aitFor()
e itValue()
```

-
-
- Runnable
- Thread

- Thread Runnable

  -

- Runnable
  Thread

java.lang.Runnable

-

  void run()

- Thread
- run
- Runnable
-

java.lang.Thread

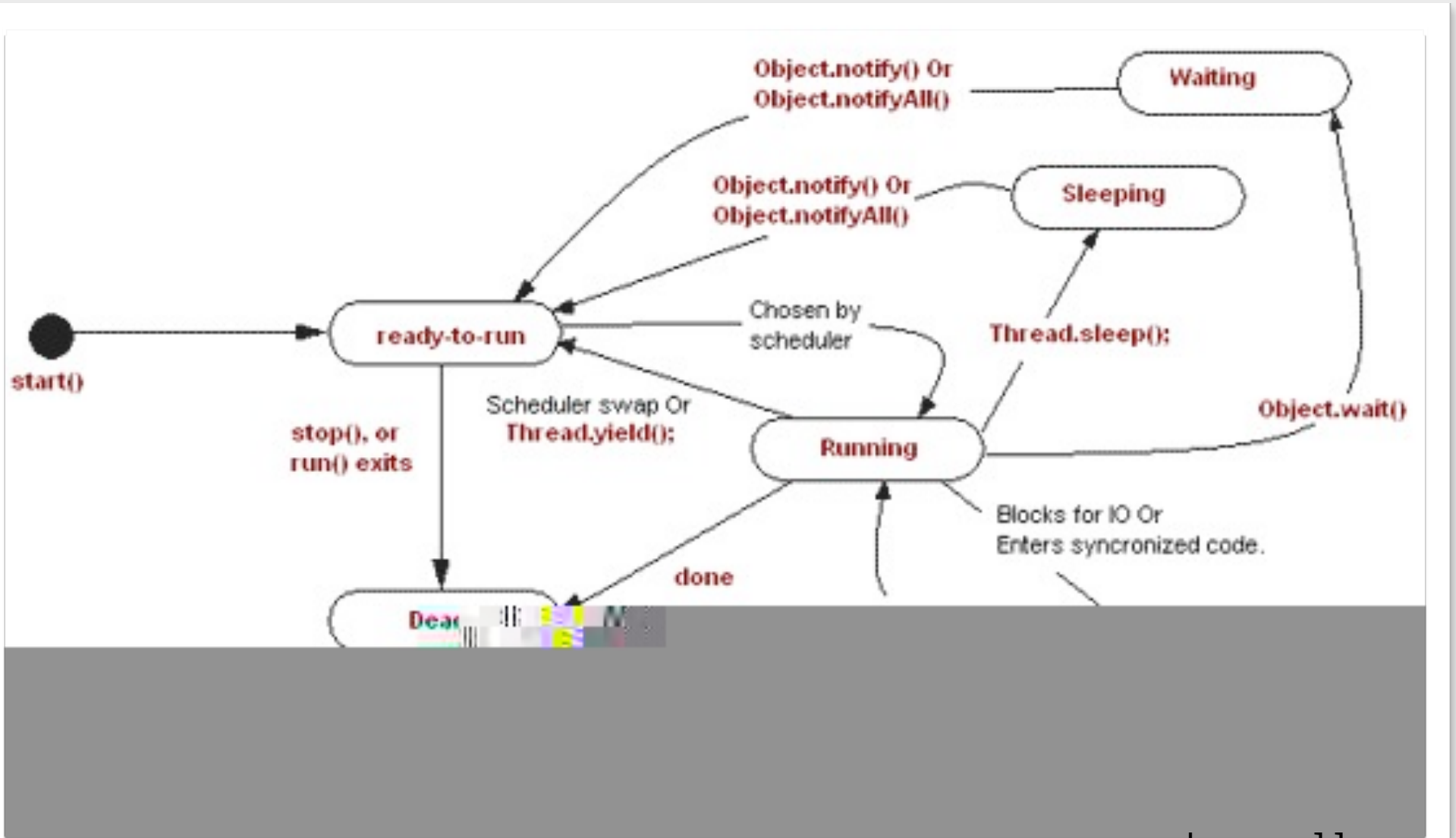
- Thread **attributs**
  - String name
  - long id
  - int priorit
    Thread
  - boolean daemon

- Thread.State state
  NEW RUNNABLE BLOCKED WAITING
  TIMED_WAITING TERMINATED

-

-

-


Object.notify() Or
Object.notifyAll()
Waiting
Object.notify() Or
Object.notifyAll()
Sleeping
Chosen by
scheduler
ready-to-run
start()
Thread.sleep();
Object.wait()
Scheduler swap Or
Thread.yield();
stop(), or
run() exits
Running
Blocks for IO Or
Enters syncronized code.
done

- **la terminaison d’une JVM**
  -
    - _____ ___ _____
  - _____
    - **Thread** **il faut attendre que TOUS les s’arrêtent**

- Thread
  -
-
  - Thread

- Thread
  - void start()
    Thread
    run() Runnable
  - void join()
    Thread

- `void interrupt()`

  `Thread`

- `Thread`

  `Thread`

- **il n’existe pas d’autre technique pour arrêter un** `Thread`

  `run()`

- Thread

  - Thread currentThread()
    Thread
    Thread

  - boolean interrupted()
    Thread

- void sleep(long ms)
  void sleep(long ms,long ns)
  Thread

- void ield() Thread

- Thread

  Runnable

-

- Thread(Runnable)

- Thread(Runnable,String)

```
class M Code implements Runnable
    public void run()
        int N = (int)(Math.random()*5);
        for (int i=0; i<10000*N; i++)
            S stem.out.println(Thread.currentThread().getName()+" i="+i);



public class E ample
    public static final int N = 20;
    public static void main(String []args)
        M Code code = ne  M Code();
        Thread []t = ne  Thread[N];

        for (int i=0; i<N; i++) t[i] = ne  Thread(code,"T"+i);

        for (int i=0; i<N; i++) t[i].start();

        for (int i=N-1; i>=0; i--)
            tr
            t[i].join();
              catch(InterruptedE ception e)
         S stem.out.println("Terminaison de "+t[i].getName());

        S stem.out.println("tout est fini ");
```

- Thread
  Runnable
  run()
- Thread
  run()
  Thread

```
class Compteur
    private int valeur;
    public Compteur()   valeur = 0;
    public int getValeur()   return valeur;
    public void setValeur(int v)   valeur=v;
```

```
class M Code implements Runnable
    private Compteur c;
    public M Code(Compteur c)   this.c = c;
    public void run()
        for (int i=0; i<100000000; i++)
            c.setValeur(c.getValeur()+1);
```

```
public class E ample
    public static final int N = 20;
    public static void main(String []args)
        Compteur c = ne  Compteur();
        M Code code = ne  M Code(c);
        Thread []t = ne  Thread[N];
        for (int i=0; i<N; i++)
            t[i] = ne  Thread(code,"T["+i+"]");
        for (int i=0; i<N; i++) t[i].start();
        for (int i=0; i<N; i++)
    tr
            t[i].join();
          catch(InterruptedE ception e)

S stem.out.println("tout est fini ");
S stem.out.println("Le compteur est  gal    "+
                        c.getValeur());
```

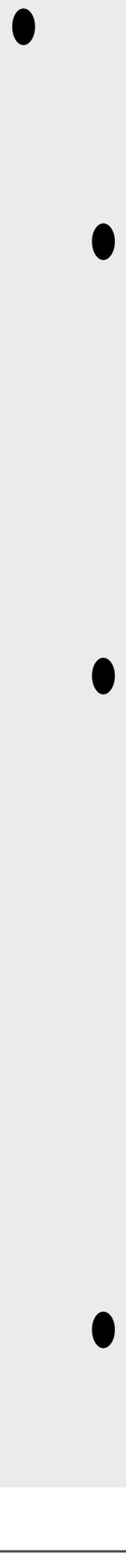

Terminal — tcsh — 80×24
[Trotinette:Programmation reÌ^Áseau/sources/threads] yunes% java Example
tout est finiâ¦
Le compteur est Ã@gal Ã 1716791363
[Trotinette:Programmation reÌ^Áseau/sources/threads] yunes%

≠

-
  -
-
  - **offensif**
  -

-
-
- **section critique**
- Thread

- **verrou**
- **exclusion mutuelle**
- s nchroni ed

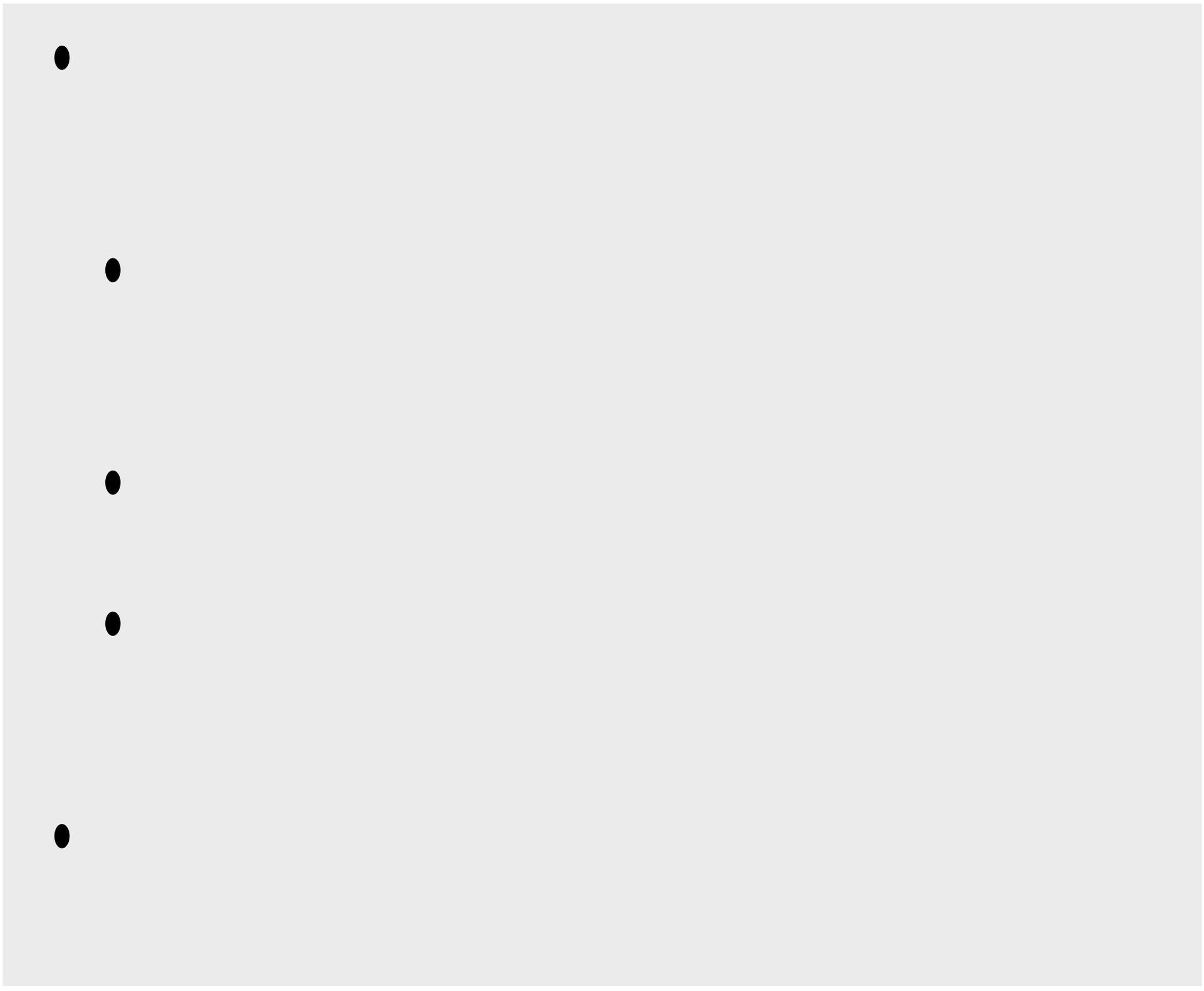


```
•
  s nchroni ed   m thode( )


•
    m thode( )
      s nchroni ed(this)
```

```
class M Code implements Runnable
    private Compteur c;
    public M Code(Compteur c)   this.c = c;
    public void run()
        for (int i=0; i<100000000; i++)
            s nchroni ed(this)
                c.setValeur(c.getValeur()+1);
```

-
-
-



```
class Produit
    private String nom;
    public Produit(String nom)
        this.nom = nom;

    public String toString()
        return nom;
```

```
class Stock
    private Produit []leStock;
    private int niveauCourant;
    private static final int capacit Ma imale = 10;
    public Stock()
        leStock = ne  Produit[capacit Ma imale];
        niveauCourant = 0;

    public boolean addProduit(Produit p)
        if (niveauCourant==capacit Ma imale)
            return false;
        leStock[niveauCourant++] = p;
        return true;

public Produit removeProduit()
    if (niveauCourant==0)
        return null;
    Produit p = leStock[niveauCourant-1];
    leStock[--niveauCourant] = null;
    return p;
```

```
class Producteur e tends Thread
    private Stock stock;
    private String nomProduit;
    private Random random;
    Producteur(Stock stock,String nomProduit)
        this.stock = stock;
        random = ne  Random();
        this.nomProduit = nomProduit;

    public void run()
         hile (true)
            Produit p = ne  Produit(nomProduit);
            tr
                Thread.sleep(random.ne tInt(1000)+1000);
              catch(InterruptedE ception e)
        if (!stock.addProduit(p))
            do
                tr
                    S stem.out.println("Producteur "+getId()+" plein");
                    Thread.sleep(random.ne tInt(100)+100);
                  catch(InterruptedE ception e)
               hile (!stock.addProduit(p));

        S stem.out.println("Producteur "+getId()+" a rajoute "+p);
```

```
class Consommateur e tends Thread
    private Stock stock;
    private Random random;
    public Consommateur(Stock stock)
        random = ne  Random();
        this.stock = stock;

    public void run()
         hile (true)
            tr
                Thread.sleep(random.ne tInt(10000)+1000);
              catch(InterruptedE ception e)
            Produit p;
            p = stock.removeProduit();
            if (p==null)
                do
                    tr
                        S stem.out.println("Consommateur "+getId()+ " vide");
                        Thread.sleep(random.ne tInt(100)+100);
                      catch(InterruptedE ception e)
                    p = stock.removeProduit();
                   hile (p==null);

            S stem.out.println("Consommateur "+getId()+" a enlev  "+p);
```

```
public class ProdCons
    public static void main(String []args)
        Stock stock = ne  Stock();
        Producteur []p = ne  Producteur[2];
        p[0] = ne  Producteur(stock,"banane");
        p[1] = ne  Producteur(stock,"carambar");
        p[0].start();
        p[1].start();
        Consommateur []c = ne  Consommateur[3];
        c[0] = ne  Consommateur(stock);
        c[1] = ne  Consommateur(stock);
        c[2] = ne  Consommateur(stock);
        c[0].start();
        c[1].start();
        c[2].start();
```

-
-
-
-
  - s nchroni ed

attente active

```
ait()   notif ()          Object

  ait()
```

```
notif ()  notif All()
```

```
class Stock
    private Produit []leStock;
    private int niveauCourant;
    private static final int capacit Ma imale = 10;
    public Stock()
        leStock = ne  Produit[capacit Ma imale];
        niveauCourant = 0;

    public s nchroni ed void addProduit(Produit p)
         hile (niveauCourant==capacit Ma imale)
        S stem.out.println("plein!");
        tr
             ait();
          catch(InterruptedE ception e)

leStock[niveauCourant++] = p;
notif All();
```

```
    public s nchroni ed Produit removeProduit()
         hile (niveauCourant==0)
            S stem.out.println("vide!");
            tr
                 ait();
              catch(InterruptedE ception e)

        Produit p = leStock[niveauCourant-1];
        leStock[--niveauCourant] = null;
        notif All();
        return p;

// fin classe Stock
```

```
class Producteur e tends Thread
    private Stock stock;
    private String nomProduit;
    private Random random;
    Producteur(Stock stock,String nomProduit)
        this.stock = stock;
        random = ne  Random();
        this.nomProduit = nomProduit;

    public void run()
         hile (true)
            Produit p = ne  Produit(nomProduit);
            tr
                Thread.sleep(random.ne tInt(1000)+1000);
              catch(InterruptedE ception e)
            stock.addProduit(p);
            S stem.out.println("Producteur "+getId()+" a rajoute "+p);
```

```
class Consommateur e tends Thread
    private Stock stock;
    private Random random;
    public Consommateur(Stock stock)
        random = ne  Random();
        this.stock = stock;

    public void run()
         hile (true)
             tr
                Thread.sleep(random.ne tInt(10000)+1000);
  catch(InterruptedE ception e)
Produit p = stock.removeProduit();
S stem.out.println("Consommateur "+getId()+" a enlev  "+p);
```

-
  - java.util.concurrent
  - java.util.concurrent.atomic
  - java.util.concurrent.locks