Java
11
Sincronismo

Vitor Vaz da Silva

Exemplo

```
public class Rectangulo {
   int textoX, textoY;          int conta=0;
   Polygon rectangulo;       Graphics2D tela;          Color cor;

   Rectangulo(int posX,int posY,int largura,int altura, Graphics2D g){
      tela=g;              rectangulo = new Polygon();
       rectangulo.addPoint(posX, posY);
       rectangulo.addPoint(posX + largura, posY);
       rectangulo.addPoint(posX + largura, posY + altura);
       rectangulo.addPoint(posX, posY + altura);
       textoX=posX+largura;     textoY=posY+altura;
       tela.setFont(tela.getFont().deriveFont(64F));
   }

   public void pinta(Color cor){
      tela.setColor(cor);          tela.fillPolygon(rectangulo);
      tela.drawString(""+conta, textoX, textoY);
   }

   public void desenha(){
      conta++;
      cor = (conta == 1) ? Color.GREEN:
         (conta == 2) ? Color.RED : Color.BLACK;
      pinta(cor);
      conta--;
   }
}
```

```
class Processo extends Thread {
   private  Rectangulo figura;
   private  int tempo;

   Processo(Rectangulo fig, int t){
      figura = fig;
      tempo = t;
      this.start();
   }

   public void run(){
      while(true){
         sleep(tempo);
         figura.desenha();
      }
   }
}
```

```
public void inicio(){
   Rectangulo rect = new Rectangulo(50, 50, 100, 20,
                                    (Graphics2D) getGraphics());
   Processo proc1 = new Processo(rect, 300);
   Processo proc2 = new Processo(rect, 300);
   Processo proc3 = new Processo(rect, 300);
}
```

- Um objecto da Classe Rectangulo (rect)
- Três objectos da Classe Processo (proc1—3)
- Os objectos proc1—3 correm a cada 500 ms
- Os objectos proc1, proc2 e proc3 chamam o método desenha do mesmo objecto rect
- O método rect.desenha() pinta o rectângulo e escreve o valor de conta (iniciado a 0)

Exemplo

```
public void desenha(){
    conta++;

    if(conta == 1) cor = Color.GREEN;
    else  if(conta == 2)  cor =Color.RED;
    else cor = Color.BLACK;

    pinta(cor);

    conta--;
}
```

Ao executar …

- Mexa a janela!
- Porque é que o rectângulo não é sempre verde com o número 1 ?
- Porque é que os números nem sempre são 1, 2 ou 3 ?

ZOOM

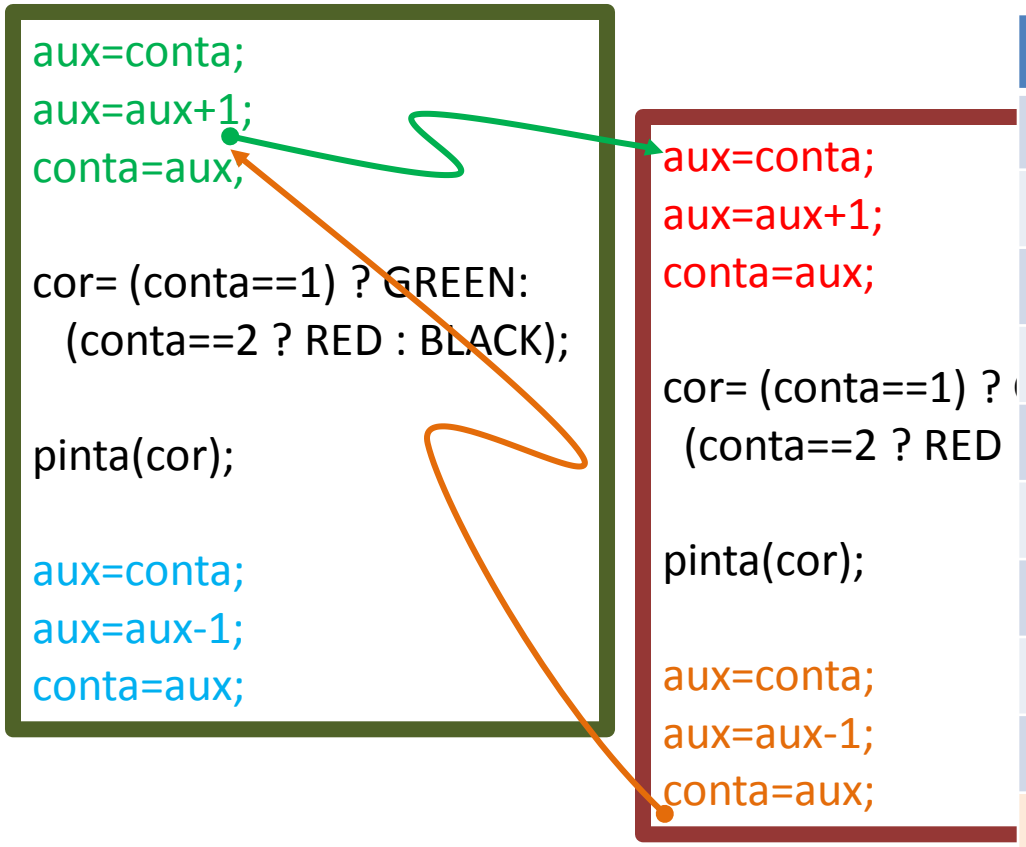
```
public void desenha(){
   conta++;

   cor= (conta==1) ? GREEN: (conta==2 ? RED : BLACK);

   pinta(cor);

   conta--;
}
```

```
aux=conta;
aux=aux+1;
conta=aux

cor= (conta==1) ? GREEN: (conta==2 ? RED : BLACK);

pinta(cor);

aux=conta;
aux=aux-1;
conta=aux;
```

```
aux=conta;
aux=aux+1;
conta=aux;
```

```
aux=conta;
aux=aux-1;
conta=aux;
```

Substituindo estas duas instruções



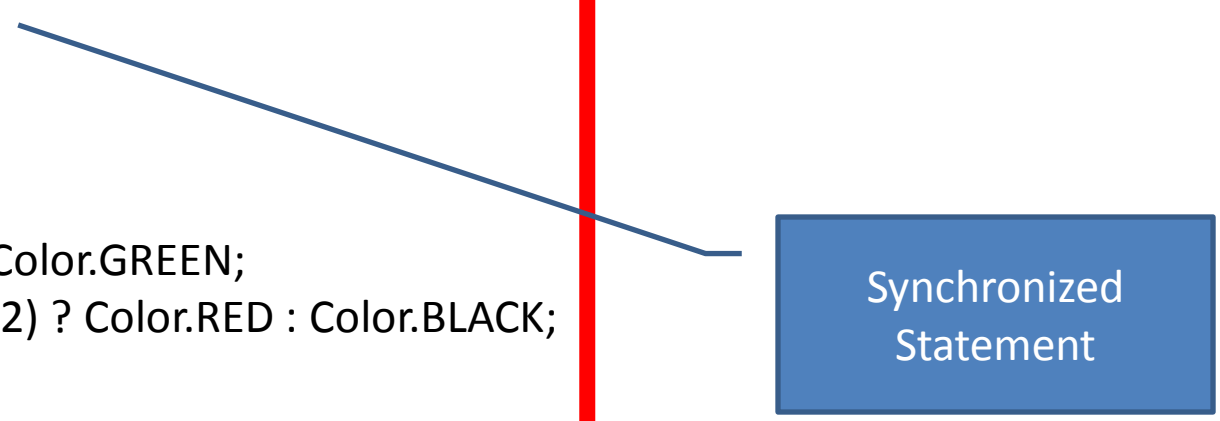
Processo	conta	aux	Instrução
1	0	0	aux=conta
1	0	1	aux=aux+1;
2	0	1	aux=conta
2	0	1	aux=aux+1;
2	1	1	conta=aux;
pinta	1	1	pinta(GREEN);
2	1	1	aux=conta
2	1	0	aux=aux-1;
2	0	0	aux=conta
1	0	0	conta=aux;
pinta	0	0	pinta(BLACK);
1	0	0	aux=conta
1	0	-1	aux=aux-1;
1	-1	-1	aux=conta

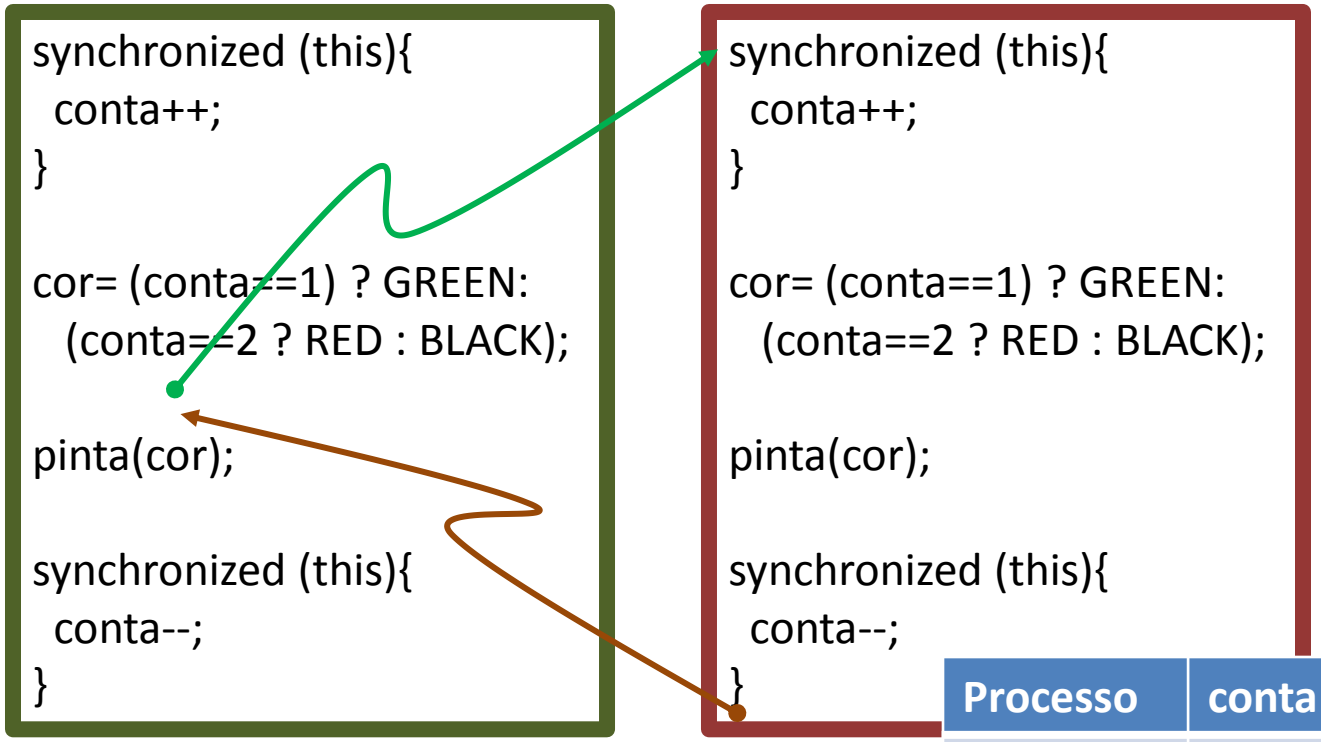
O que acontece se um processo interrompe outro sem que este tenha terminado?

Solução?!

- Executar conta++ e conta– sem permitir que o processo seja interrompido!

```
public void desenha(){
   synchronized (this){
      conta++;
   }

   if(conta == 1) cor = Color.GREEN;
   else cor = (conta == 2) ? Color.RED : Color.BLACK;

   pinta(cor);

   synchronized (this){
      conta--;
   }
}
```





O que acontece se um processo interrompe outro sem que este tenha terminado?

Processo	conta	cor	Instrução
1	1	---	conta++;
1	1	GREEN	cor=GREEN;
2	2	GREEN	conta++
2	2	RED	cor=RED;
pinta	2	RED	pinta(RED);
1	1	RED	conta--;
pinta	1	RED	pinta(RED);
1	0	RED	conta--;

Solução

- Esta solução já funciona!
- O resultado é sempre um rectângulo verde com o número 1 também a verde

```
public void desenha(){
   synchronized (this){
      conta++;

      cor= (conta==1) ? GREEN:
        (conta==2 ? RED : BLACK);

      pinta(cor);

      conta--;
   }
}
```

```
public synchronized void desenha(){
   conta++;

   cor= (conta==1) ? GREEN:
     (conta==2 ? RED : BLACK);

   pinta(cor);

   conta--;
}
```

Synchronized Method

Sincronismo

- Synchronized Method
  - Permite um acesso exclusivo ao método quando este é chamado por diversos processos (threads).

- Synchronized Statement
  - Permite um acesso exclusivo ao bloco de instruções quando o método é chamado por diversos processos.

Sincronismo

Efeitos de um sincronismo …

- **Deadlock** – Os processos ficam encravados porque um espera por recursos que outro tem mas que não liberta sem ter os recursos que outro tem.

Sincronismo

Efeitos de um sincronismo …

- **Livelock** – Os processos ficam activamente a pedir e libertar entre si recursos sem terem todos aqueles necessários para continuar.

Sincronismo

Efeitos de um sincronismo …

- **Starvation** – Os processos não conseguem obter recursos porque estes estão a ser utilizados por outros processos de um modo ganancioso.

Referências

- http://download.oracle.com/javase/tutorial/essential/concurrency/index.html
- www.tektonia.com