Gijos

Įvadas į gijų modelį Javoje

1

Turinys

- Motyvacija
- Sukūrimas
- Valdymas
- Sinchronizacija
- Susijusios klasės

2

Motyvacija

- Gijos reikalingos tam, kad išreikšti lygiagretumą vieno proceso rėmuose
- Pavyzdžiai:
  - Naršyklė – geba vienu metu siųsti ir vaizduoti kelis puslapio elementus
  - UI programa, galinti reaguoti į vartotojo veiksmus vykdant "foninę" veiką
  - Apsaugos sistema, aptarnaujanti daug asinchroninių daviklių.

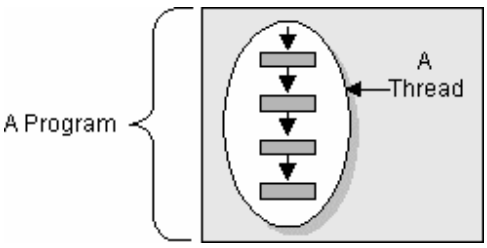
3

Gijos

- Tai "lengvi" lygiagretūs procesai programos viduje
- Tai nuoseklaus vykdymo / valdymo srautai proceso rėmuose, turintys
  - Nuosavą steką (lokalius kintamuosius)
  - Komandų skaitliuką
- tačiau
  - naudojantys "globalią" atmintį bei kitus programos resursus

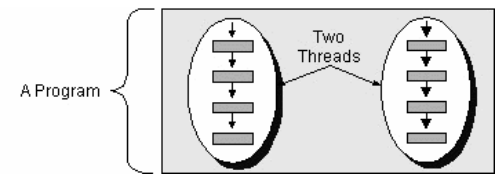
4

Vienos gijos programa



5

Daugelio gijų programa



6

Java ir gijos

- Java ne vienintelė daugiagijė program. sistema, tačiau išskirtinė.
- Gijų priemonės:
  - kalbinės: ***volatile, synchronized***
  - klasės:*Thread, Runnable, InterruptedException, Object* ir kt.
  - Vykdomoji (*Runtime*) aplinka

7

Javos gijos sukūrimas ir paleidimas

- Visuomet yra ***main()*** gija
- Sukonstruojamas ***Thread*** objektas (gijos paleidimui ir valdymui)
- Kviečiamas gijos metodas ***start()***
- Paleidžiama gija (kaip lyg. mini-procesas)
- Kuriame vykdomas metodas ***run()***
- Gija baigia darbą , kai grįžtama iš *run()*

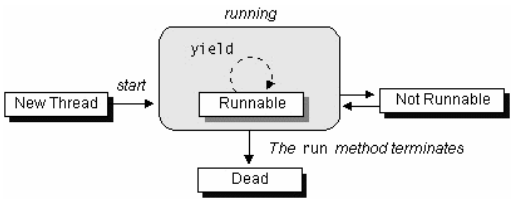
8

Sukūrimo kodas

- Kodą kuris bus vykdomas gijoje apibrėžiame:
  - Užklodami metodą *run()* išvestinėje iš *Thread* klasėje
  - Arba realizuodami metodą *run() Runnable* interfeiso išvestyje.
- Paprastai išvestinis *Thread* objektas saugo papildomą informaciją, panaudojamą gijos vykdymo metu

9

Gijos "gyvavimo" ciklas



10

Gijos valdymas

- Vykdomas klasės ***Thread*** metodais.
- Paprastai statiniai metodai įtakoja vykdomąją giją
- static Thread **currentThread**() – vykdomosios gijos objektas

11

Prioritetai

- **MAX_PRIORITY** , **MIN_PRIORITY**, **NORM_PRIORITY**
- int **getPriority**()
- void **setPriority**(int newPriority)
- static void **yield**()
  - vykdymo iniciatyvos perdavimas kt. gijai
- Pastaba: programos "teisingas" funkcionavimas neturi remtis prioritetais

12

(Pri)stabdymas / pratęsimas

- static void **sleep**(long millis)
- void **suspend**() (**Deprecated**)
- void **resume**() (**Deprecated**)
- **void stop** () (**Deprecated**)

13

Gijos pažadinimas

- void **interrupt**()
  - pažadina miegančią giją
- static boolean **interrupted**(),
- boolean **isInterrupted**()
  - patikrina, ar gija nebuvo pažadinta

14

Gijos darbo užbaigimas požymiu

```
class X extends Thread {
        volatile boolean finish  = false;
        //...
        public void run(){
                while ( ! finish ) {
                        // Do something
                }
        }
}
```

15

Ar gija vykdoma ?

- boolean **isAlive**()
  - ar gija vykdoma
- void **join**([long millis[, int nanos]])
  - laukti iki (kita) gija baigs darbą

```
//...
Thread[] threads; // Initialize...
for (int i = 0; i < threads.length; i++)
        threads[i].join();
//...
`
```

Gijų sinchronizacija

- Reikalinga, kad įgalintų lygiagretaus (*concurrent*) atminties duomenų skaitymo / modifikavimo operacijų korektišką veikimą
- Priemonės:
  - **synchronized** metodai
  - **synchronized** sakinys
- Gijos sinchronizuojamos konkrečiu objektu

17

Garantijos

- Skaitymo/rašymo operacijos su *reference* arba primityvių tipų laukais, išskyrus *double* ir *long int*, yra atomarinės.
- Jeigu klasės laukas paskelbtas kaip *volatile,* prieš atliekant skaitymo/rašymo operaciją su šiuo lauku, “lokalus” atminties objektas bus sinchronizuotas su “globalia” reikšme.

18

Monitoriai

- Kiekvienas java objektas aprūpintas monitoriumi - užraktu
- Kviečiant objekto sinchronizuotą (užraktu aspsaugotą) metodą, tik viena gija gali būti monitoriaus viduje
- Kitos gijos laukia (pristabdytos) monitoriaus išorėje
- Pavidalas:
  - **synchronized** *metodas (parametrai) {...}*
  - **synchronized** *(objektas) {sakiniai}*

19

Synchronized subtilybės

- Gija gali vykdyti "įdėtuosius" sinchronizuotų metodų kvietinius tam pačiam objektui nesiblokuodama.
- Išvestinėje klasėje bazinės klasės *synchronized* metodas gali būti perrašytas kaip nesinchronizuotas, taip pažeidžiant bazinės klasės kontraktą.
- Monitorius neapsaugo nesinchronizuotus kvietinius
- *static synchronized* – sinchr-ja klasės objektu

20

Įvykiai:**Object.wait()/notify()** šeima

- Įvykio laukimas:
  - *object*.wait() / *object*.wait(millis)
- Pranešimas apie įvykį:
  - *object*.notify() / *object*.notifyAll()

// Thread2:	// Thread1:

condition = true;
object.**notifyAll()**;
 	
//--------->
	while(!*condition*){
	object.**wait()**;
	}
	// ...

21

Nuorodos

- ***ThreadGroup***
  - įgalina grupines operacijas su gijomis
- ***Timer***'iai
  - palengvina panaudojima UI
- ***SwingUtilities*.invoke**[**Later**](*Runnable*)
  - užtikrina foninės gijos ir SWING'o UI gijų sąveikas

22

Pabaiga

23