Objektno orijentisano programiranje 2

Niti u C#

Uvod

- Kao i Java, C# podržava konkurentno programiranje
- Glavnina podrške za rad sa nitima je u biblioteci jezika
- Pojam aktivne klase i objekta drugačije koncipiran nego u Javi
  - aktivan objekat sadrži objekat niti
- Podržano:
  - stvaranje niti
  - pokretanje niti (`Start`)
  - dohvatanje svojstava niti (`Name`, `Priority`, `ThreadState`)
  - prekidanje/deblokiranje niti (`Interrupt`)
  - uspavljivanje niti (`Sleep`)
  - ustupanje procesora (`Yield`)
  - čekanje da druga nit završi (`Join`)
  - sinhronizacija niti (naredba `lock` i metodi klase `Monitor`)
  - komunikacija niti – čekanje na signal (`Wait-Pulse`)

Stvaranje niti

- Klasa `Thread` – klasa čiji objekti imaju vlastite niti

```
sealed class Thread{...}
```

- Klasa se nalazi u prostoru imena `System.Threading`
- Iz ove klase se ne izvode aktivne klase
- Aktivna klasa sadrži polje tipa `Thread`
- Nit se stvara kao objekat klase `Thread` konstruktorima:

```
Thread(ThreadStart aktivnost)
Thread(ParametrizedThreadStart aktivnost)
```

gde su `ThreadStart` i `ParametrizedThreadStart` delegati:

```
delegate void ThreadStart()
delegate void ParametrizedThreadStart (object  parametar)
```

Delegat i poziv konstruktora

- Parametar delegata je tipa `object`, pa se mora konvertovati u ciljni tip
- U pozivu konstruktora argument može da bude metod
  - metod se automatski konvertuje u odgovarajući tip delegata
- Primer:

```
class A {
   public static void A1(){...}
   public static void A2(object parametar){
      int p=(int) parametar; //...
   }
}
    Thread nit1 = new Thread(A.A1);
    Thread nit2 = new Thread(A.A2);
```

Aktivna klasa

- Aktivna klasa se definiše tako da ima privatno polje tipa `Thread`
- U konstruktoru aktivne klase može da se stvori objekat niti
- Konstruktoru se prosleđuje metod delegata (ili delegat) koji definiše telo niti
- Primer:

```
class Aktivna{
  private void Aktivnost(){...}
  private Thread nit;
  //...
  public Aktivna(){nit=new Thread(Aktivnost);}
  //...
}
```

- Stvaranjem objekta niti ova još nije aktivirana (spremna za izvršenje)

Pokretanje i završavanje niti

- Nit se aktivira metodom `Start()`
- Postoji i metod `Start(object parametar)`
  - koristi se za niti stvorene konstruktorom čiji delegat ima parametar
  - argument se prosleđuje metodu delegata
- Izvršno okruženje poziva metod delegata kad niti dodeli procesor
- Metod `Start()` se poziva samo jednom, kada nit završi – ne može ponovo
  - ako se pokuša ponovo – izuzetak `ThreadStateException`
- Primer aktiviranja niti (u konstruktoru klase `Aktivna`):

```
nit.Start();
```

- Nit se završava završetkom metoda prosleđenog delegata
  - tada objekat aktivne klase postaje pasivan, ali i dalje postoji

Režimi izvršavanja niti

- Nit se može izvršavati:
  - u prvom planu (*foreground*)
  - u drugom planu, odnosno u pozadini (*background*)
- Koncept sličan korisničkim/demonskim nitima u Javi
  - niti u prvom planu odgovaraju korisničkim nitima
  - niti u pozadini odgovaraju demonskim nitima
- Niti u pozadini se završavaju kada se sve niti u prvom planu završe
- Glavna programska nit je u prvom planu
- Režimom izvršavanja niti se upravlja preko svojstva

```
bool isBackground;  // podrazumevano false
```

- Svojstvo može da se menje i tokom izvršavanja niti

Svojstva niti i metodi

- Ime niti (svojstvo):

```
string Name
```

- Prioritet niti (svojstvo):

```
enum ThreadPriority{Highest,AboveNormal,Normal,BelowNormal,Lowest}
ThreadPRiority Priority
```

- Stanje niti (svojstvo):

```
enum ThreadState{Unstarted,Running,WaitSleepJoin,Stopped,...}
ThreadState ThreadState
```

- Provera da li je nit aktivna (metod):

```
bool isAlive()
```

- Dohvatanje tekuće niti (statičko svojstvo):

```
static Thread currentThread
```

Sinhronizacija niti

- Klasa `Monitor` obezbeđuje mehanizme za sinhronizaciju
- Apstrakcija brave objekta
- Samo statički metodi koji rukuju bravom objekta koji je njihov parametar
- Objekat brave sadrži:
  - referencu na štićeni objekat kojem je brava pridružena
  - referencu na nit koja trenutno drži bravu zaključanom
  - red niti koje čekaju na bravu
  - Red niti koje čekaju na signal za nastavak korišćenja štićenog objekta
- Kritična sekcija - implicitan ulaz/izlaz iz sekcije:

```
lock (objekat){...}
```

Eksplicitan ulaz/izlaz iz krit. sekcije

- Ostvaruje se statičkim metodima klase Monitor
- Metodi za ulazak u kritičnu sekciju uz blokiranje:

```
static void Enter(object o)
static void Enter(object o, ref bool dodeljena)
```

- Metodi za pokušaj ulaska u kritičnu sekciju bez blokiranja:

```
static void TryEnter(object o)
static void TryEnter(object o, ref bool dodeljena)
static void TryEnter(object o, int milisek)
static void TryEnter(object o, int milisek, ref bool dodeljena)
```

- Metod za izlazak iz kritične sekcije (otključavanje brave):

```
static void Exit(object o)
```

Komunikacija niti

- Ostvarju se statičkim metodima klase `Monitor`
- Čekanje na događaj

```
static bool Wait(object o)
static bool Wait(object o, int milisek)
```

- Slanje signala

```
static void Pulse(object o)
static void PulseAll(object o)
```