Klasa kao realizacija softverske mašine

- Problem: ako ponašanje nekog aktivnog objekta postane suviše složeno i dugotrajno, da bi se postigao privid paralelizma, to ponašanje se mora deliti na manje intervale; tada se kontekst (stanje) mora čuvati i prenositi između različitih poziva operacije *step*, pa programiranje postaje teško, a kod nepregledan
- Osim toga, šta ako aktivni objekat ne može ili ne treba da nastavi svoju aktivnost, odnosno treba da je suspenduje dok se ne ispuni neki uslov?
- Na primer: program za obradu teksta; korisnik pokreće operaciju snimanja dokumenta u fajl (*save*); da li čekati da se ova operacija završi i ne dozvoliti korisniku da bilo šta radi sve dok ona traje? Šta ako to potraje previše? Bespotrebno!
- Rešenje: ponašanje aktivnih objekata realizovati kao *niti* (*thread*)
- *Nit* (*thread*) je sekvenca izvršenih akcija (naredbi) koja teče *uporedo* sa drugim takvim sekvencama (nitima)
- Nit se kreira nad pozivom neke funkcije, a kontrola niti dalje ide kako diktira kod te funkcije i onoga što ona dalje poziva
- Svaka nit ima svoj *tok kontrole* (*control flow*), koji uključuje i automatske promenljive — svaka nit ima *svoj stek poziva*

Copyright 2018-2024 by Dragan Milićev

Klasa kao realizacija softverske mašine

```
class DocumentSaver {
public:
  …
  DocumentSaver (Document*);
 ~DocumentSaver ();
  …
private:

  static void run (DocumentSaver*);

  void save (); // Performs document saving
  Document* myDocument;
  thread* myThread;
  …
};

void DocumentSaver::run (DocumentSaver* ds) {
  if (ds) ds->save();
  delete ds;
}

DocumentSaver::DocumentSaver (Document* d) {
  this->myDocument = d;
  if (d) myThread = new thread(run, this);
}

DocumentSaver::~DocumentSaver () {
  delete myThread;
}
```

- Kada treba pokrenuti akciju snimanja dokumenta, samo treba kreirati jedan objekat klase *Document Saver*:

```
new DocumentSaver(theDocument);
```

- Sada su objekti klase *DocumentSaver aktivni*: svaki objekat ove klase ima jednu nit u kojoj se izvršava operacije *save*
- Kada se ova operacija završi, objekat ove klase se uništava
- Ova nit, pa time i operacija *save*, izvršava se uporedo sa svim drugim nitima u programu
- Ova nit, odnosno operacija *save* koju ona izvršava, može da pristupa *deljenim podacima*: pristupa strukturi podataka koja predstavlja dokument (*Document*), kako bi ga snimila u fajl
- Uporedo sa tim, ostale niti rade svoj posao: na primer, "glavna" nit može da obrađuje akcije korisnika, pa korisnik ne mora više da čeka na završetak operacije *save*
- Operacija *save* ne mora da se deli na korake, već se programira kao jedinstvena, sekvencijalna operacija, što olakšava programiranje

Copyright 2018-2024 by Dragan Milićev