Zavod za elektroniku, mikroelektroniku
računalne i inteligentne sustave

18. svibnja 2021.

Oblikovni obrasci u programiranju

3. laboratorijska vježba

1. Razmatramo transparentno rukovanje životinjama koje su opisane u binarnim dinamičkim bibliotekama. Vaši zadatci su:

   (a) Napisati kod u C-u koji stvara polje od 5 objekata opisanih u dinamičkoj biblioteci `parrot.so`. Neka imena objekata budu Hera, Atena, Artemida, Zeus i Hermo.

   (b) Dodati biblioteku `Cow` koja modelira životinju koja pozdravlja s "muuu" i voli jesti "novine".

   (c) Dodati funkciju koja modelira metodu `Cow.love(person, answer)`. Ciljana metoda prima znakovni niz `person`, a u znakovni niz `answer` upisuje `"<ime> voli <person>!"`. Pri tome je oznake `<ime>` i `<person>` potrebno zamijeniti imenom koje je definirano prilikom incijaliziranja životinje odnosno sadržajem argumenta metode. Protokolom je utvrđeno da znakovni niz `answer` treba imati barem 1024 znaka.

2. U okviru 2. zadatka laboratorijske vježbe ostvarili ste jednostavnu grafičku komponentu za uređivanje teksta koja se temelji na razrađenom modelu tekstovnog dokumenta. Sada bismo željeli proširiti postojeće rješenje tako da korisniku omogući nove naprednije funkcionalnosti. Konkretno, htjeli bismo podržati funkcionalnosti poput:

   - na pritisak CTRL i lijeve kursorske tipke da se kursor pomakne na početak prethodne riječi,
   - na pritisak CTRL+SHIFT i lijeve kursorske tipke da se kursor pomakne na početak prethodne riječi i proširi selekciju,
   - na pritisak tipke CTRL+HOME ili alternativno na pritisak tipke F2 da se kursor pomakne na početak dokumenta,
   - na pritisak tipke CTRL+SHIFT+HOME ili alternativno na pritisak tipke SHIFT+F2 da se kursor pomakne na početak dokumenta i "preskočeni" dio uključi u selekciju, i slično.

   Potpun skup ovih proširenja trenutno nije poznat, te korisniku treba omogućiti da sam dodaje nove funkcionalnosti koje se pozivaju preko tipkovnice, na način da zbog toga ne mora mijenjati postojeći kod razvijenih komponenata. Primijetite također da se neka funkcionalnost potencijalno može pozvati i na više načina.

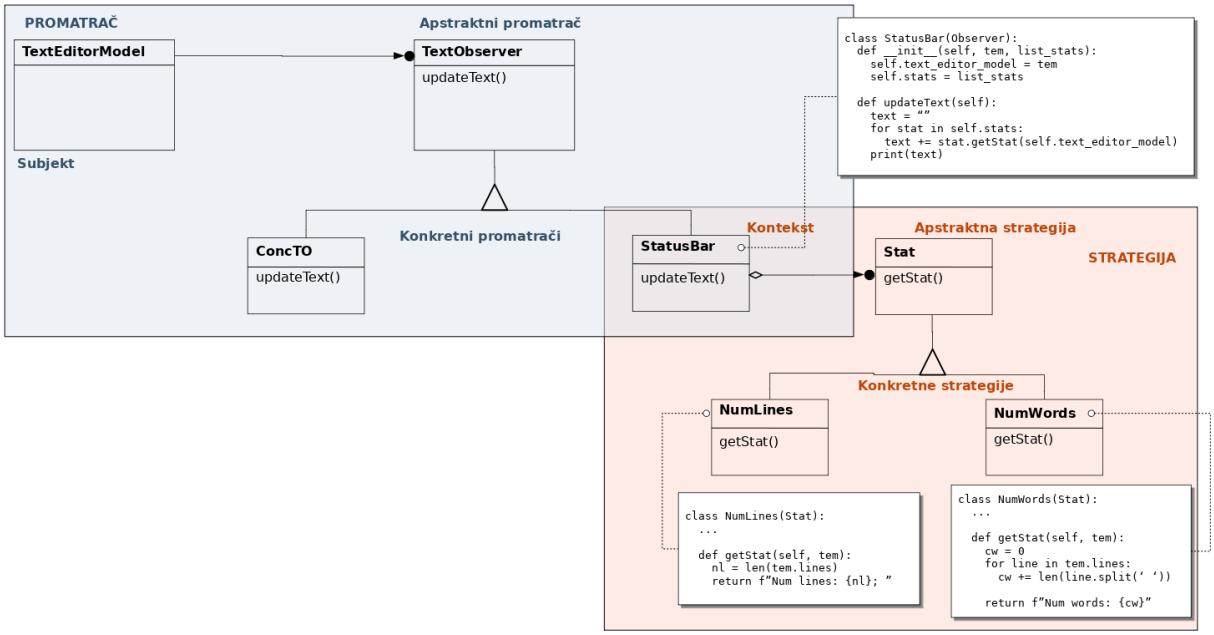
   Predložite prikladno programsko rješenje koje će biti usklađeno s načelima. U skladu s Vašim rješenjem napišite konkretan programski kod koji ostvaruje prethodno opisanu funkcionalnost pomicanja kursora na početak dokumenta uz proširenje trenutne selekcije. Prikažite dijagramom razreda relevantne sudionike i povežite ih s primijenjenim oblikovnim obrascima.

3. Redizajnirajte statusnu traku tako da klijenti mogu konfigurirati njen sadržaj. Jedna moguća konfiguracija prikazala bi broj redaka, riječi i slova. Druga konfiguracija prikazala bi broj redaka i položaj kursora. Očekujemo da će nove verzije programa nuditi neke nove oblike prikaza koji će odražavati različita svojstva dokumenta. Vaše rješenje mora omogućiti proizvoljno kombiniranje elemenata prikaza uz minimiziranje ponavljanja koda. Vaša komponenta treba ponuditi sučelje za dinamičko konfiguriranje sadržaja statusne trake, ali ne treba razmatrati komponente koje to sučelje koriste. Nacrtajte dijagram razreda svog rješenja, i povežite ga s korištenim obrascima.

Rješenje prvog zadatka

```
// cow.c -> cow.so
static char const* love(void* p, char const* person, char* answer){
  struct Cow *this = (struct Cow*) p;

  int lenname=strlen(this->itsName);
  int lenperson=strlen(answer);

  if (lenname+lenperson+8>1024){
    printf("Instance␣name␣or␣person␣is␣too␣long\n");
    return 0;
  }

  sprintf(answer, "%s␣voli␣%s!", this->itsName, person);
  return "";
}

// main.c -> a.out
char const* names[]={"Hera", "Atena", "Artemida", "Zeus", "Hermo"};

int main(void){
  int szcow = myfactory_szof("cow");

  char* p=malloc(5*szcow);
  for (int i=0; i<5; ++i){
    myfactory_construct("cow", &p[i*szcow], names[i]);
  }

  for (int i=0; i<5; ++i){
    char answer[1024];
    struct Animal* a_i = (struct Animal*)&p[i*szcow];
    a_i->vtable[3](a_i, "Sinisu", answer);
    printf("%s\n", answer);
  }
}
```

Rješenje trećeg zadatka



```
class TextEditorModel{
  ...
  void setCursorLocation(Location cl){
    this.cursorLocation = cl
  }
  ...
}

class TextEditorKeyListener{
        private Map<KeyCombination, MoveCursorAction> keyCursorActionMap;

        public void keyPressed(KeyEvent e) {
        ...
        KeyCombination k = new KeyCombination(e.getKeyCode(), e.isShiftDown(), e.isControlDown()...);
                if keyCursorActionMap.contains(k) {
                        keyCursorActionMap.get(k).execute();
                }
                ...
        }
}

public abstract class MoveCursorAction {
    private TextEditorModel model;

    public MoveCursorAction(TextEditorModel model) {
        this.model = model
    }

    public void execute();
}

public class MoveToStart extends MoveCursorAction {
    public void execute() {
        this.model.setCursorLocation(new Location(0, 0));
    }
}
```