Zavod za elektroniku, mikroelektroniku računalne i inteligentne sustave

16. travnja 2012.

Oblikovni obrasci u programiranju

1. međuispit

Napomena uz sve zadatke: prihvaćaju se programska rješenja u C++-u, Javi, Pythonu i C#-u.

1. U programskom jeziku C++ definirana su sljedeća dva razreda, te metoda main kako slijedi.

```
class Osnovna {
protected:
  int a;
  int b;
public:
  virtual void set(int a, int b) {
    this->a = a;
    this->b = b;
  }
};
```

```
class Izvedena: public Osnovna
{
  int c;
public:
  virtual void set(int a, int b) {
    this->a = a+b;
    this->b = a;
    this->c = a-b;
  }
};
```

```
int main() {
  Osnovna *o = new Izvedena;
  o->set(5, 12);
}
```

   (a) Skicirajte sadržaj memorije prije poziva naredbe **return** u metodi `main` (zauzete memorijske objekte, njihovu strukturu i sadržaj te tablice virtualnih funkcija i njihov sadržaj).

   (b) Koliko bi na 32-bitnim platformama iznosilo **sizeof**`(Osnovna)` a koliko **sizeof**`(Izvedena)` i zašto?

   (c) Kako se doista izvodi poziv `o->set(5,12)`? Ilustrirajte to u pseudo-C-u.

   (d) Kakve bi se operacije trebale provesti u generiranom konstruktoru razreda Osnovna?

2. Je li u zadanom programskom odsječku izvedbom razreda `NoviModul` iz razreda `Modul` prekršeno koje načelo oblikovanja? Obrazložite odgovor.

```
class Modul {
protected:
  double mysqrt(double x) {
    assert(x>=0.0);
    return sqrt(x);
  }
public:
  virtual double izracunaj(double x) {
    return mysqrt(x-4) + mysqrt(-x+20);
  }
};
```

```
class NoviModul :
  public Modul
{
public:
  virtual double izracunaj(double x) {
    return mysqrt(x-6) + mysqrt(-x+15);
  }
};
```

3. Objasnite gdje se u prikazanom primjeru koristi statički a gdje dinamički polimorfizam. Pojasnite. Nacrtajte pojednostavljeni dijagram razreda (OMT, kao u predavanju).

```
typedef std::vector<Zapis*> MyVector;
class Imenik {
public:
  MyVector zapisi;
  void obrada() {
    for(int i=0; i<zapisi.size(); i++){
      zapisi[i]->poruka();
    }
  }
};
```

```
class Zapis {
  public:
    virtual void poruka() { ... }
};

class VazanZapis : public Zapis {
  public:
    virtual void poruka() { ... }
};
```

4. Program za generiranje i sortiranje brojeva prikazan je u nastavku.

```
class Radnik {
public:
  void radi(int generator, int nacinSorta) {
    int * niz;
    switch(generator) {
      case 1: niz = generirajNizBrojeva1(); break;
      case 2: niz = generirajNizBrojeva2(); break;
      case 3: niz = generirajNizBrojeva3(); break;
    };
    switch(nacinSorta) {
      case 1: sortiraj1(niz); break;
      case 2: sortiraj2(niz); break;
      case 3: sortiraj3(niz); break;
    };
  }
};
```

   Metoda `Radnik::radi()` omogućava odabir i) postupka generiranja niza brojeva (slijedno iz raspona, uporabom brzog generatora slučajnih brojeva, uporabom sporijeg ali kvalitetnijeg generatora slučajnih brojeva, itd.), te ii) postupka njihovog sortiranja (primjerice, radix-sort, merge-sort, heap-sort, bubble-sort, quick-sort). Predloženu izvedbu potrebno je modificirati u skladu s prikladnim obrascem oblikovanja. Skicirajte modificiranu izvedbu te prikažite pojednostavljeni dijagram razreda (OMT) s naznačenim sudionicima obrasca.

5. Na raspolaganju je komponenta `Informator` preko koje je moguće dobivati informacije o stanju računala poput temperature procesora, temperature kućišta, opterećenosti jezgara i slično. Objasite kako biste implementirali komponentu `Informator` tako da omogućite jednostavnu izgradnju niza različitih komponenata grafičkog korisničkog sučelja koje korisniku prikazuju te podatke na različite načine (npr. trenutno stanje, kretanje kroz posljednjih pola sata, prosječne vrijednosti, itd). Vaše rješenje mora omogućiti jednostavno dodavanje novih komponenata te mora podržavati istovremenu uporabu odabranih komponenata. Objasnite koji je oblikovni obrazac za to prikladan i naznačite sudionike obrasca u vašem rješenju.

6. Definiran je apstraktni razred `Messenger` koji služi za slanje poruka. Napravljene su dvije implementacije tog razreda: `MailMessenger` (koji poruku šalje na preddefinirani e-mail) te `FileMessenger` (koji poruku dodaje u preddefiniranu datoteku).

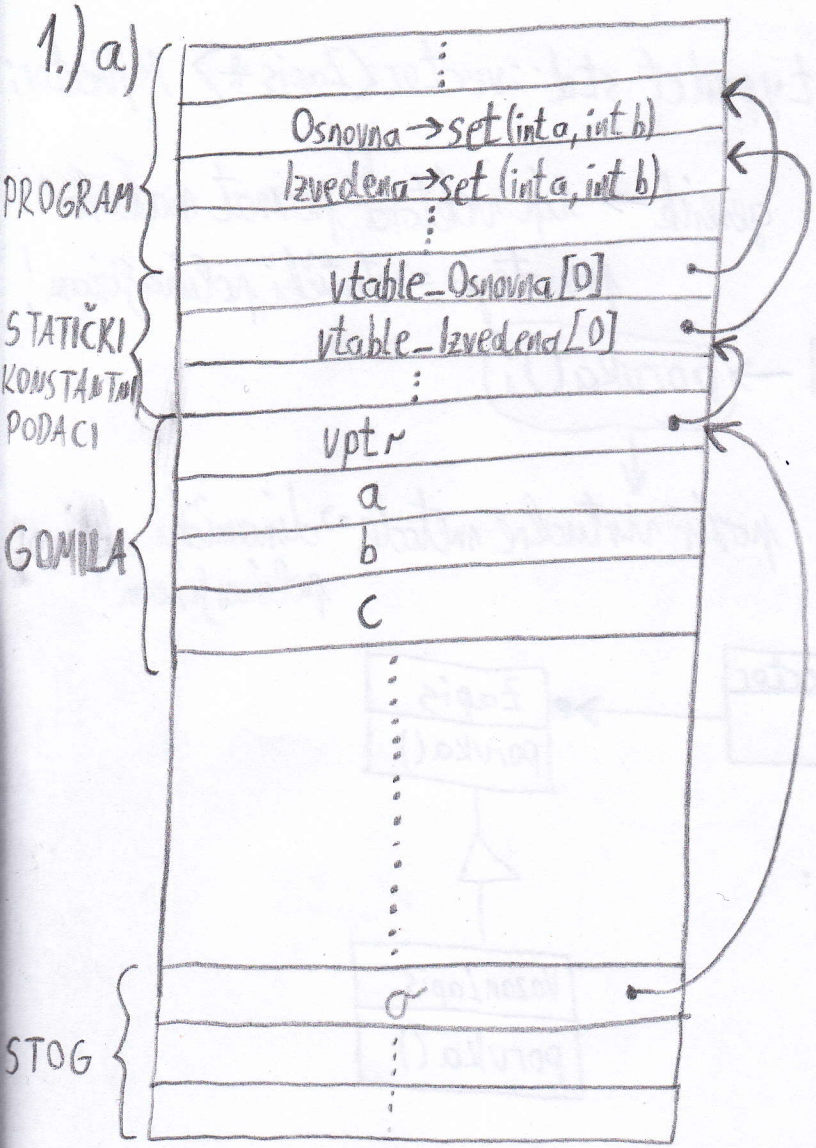
```
class Messenger {
public:
  virtual void sendMessage(char *message) = 0;
};
```

   Osmislite i predložite nadogradnju prikazanog rješenja koje će omogućiti da korisnik uz predanu poruku zapisuje još i različite druge informacije (primjerice, naziv dretve koja je poslala poruku, vrijeme kada je poruka poslana, datum kada je poruka poslana, korisnik koji je poruku poslao i još niz drugih podataka). Rješenje mora biti takvo da omogućava lagan razvoj koda koji će dodavati i druge informacije. Konkretnu selekciju koje će se to informacije zapisivati i kojim redoslijedom korisnik mora moći lagano definirati.

7. Pretpostavimo da objekte tipa `SuperDog` koriste dvije grupe klijenata: policajci i umirovljenici (radi se o policijskim psima koji tijekom ljeta rade u hotelu). Policajcima trebaju metode `searchGrass(SearchObject&)` i `attack(SuspectedCriminal&)`, dok umirovljenicima trebaju `bringSocks()` i `helpCrossStreet(Street&)`. Pored toga, poznao je da svi vlasnici pasa trebaju redovno pozivati metode `eat()` i `drink()`. Predložite izvedbu razreda `SuperDog` koja bi bila u skladu s oblikovnim načelima: nacrtajte OMT dijagram te skicirajte izvorni kod.

1. MI 2012.

1.) a)



b) sizeof(Osnovna) = 12 B
sizeof(Izvedena) = 16 B

c) σ→vptr[0](σ, 5, 12); ("this" ↑ σ)

d) Postavljanje pokazivača na tablicu virtualnih funkcija razreda Osnovna.

this→vptr = &vtable_Osnovna[0];

2.) Prekršeno je Liskovino načelo supstitucije (načelo nadomjestivosti osnovnih razreda).
Za razred Modul preduvjet za x je interval $x \in [4, 20]$, dok je za razred NoviModul preduvjet za x interval $x \in [6, 15]$.
Prema tome, vidimo da razred NoviModul ne poštuje ugovore osnovnog razreda jer „puca" za ulazne vrijednosti $x \in [4, 6\rangle$ i $x \in \langle 15, 20]$ koje OsnovniModul podržava ⇒ NoviModul ne može nadomjestiti Modul!

3.) Statički polimorfizam: zapisi[i] → poruka();

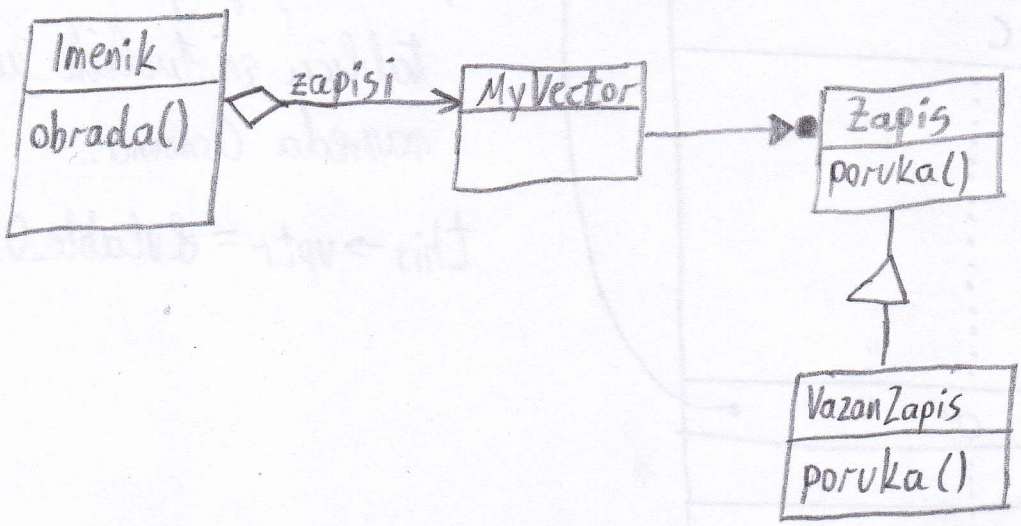
statički zbog typedef std::vector<Zapis*> MyVector;

generik → tip vektora poznat nakon prevođenja => statički polimorfizam!

Dinamički polimorfizam: zapisi[i] → poruka();

poziv virtualne metode => dinamički polimorfizam!

OMT:



4.) Oblikovni obrazac Strategija

```
// Radnik.hpp
class Radnik{
  public:
    Radnik(Generator& gen, Sort& sort);
    void radi();
  private:
    Generator& gen_;
    Sort&      sort_;
};
```

```
// Radnik.cpp
void Radnik::radi(){
    int* niz;
    niz = gen_.generirajNizBrojeva();
    sort_.Sortiraj(niz);
}
```

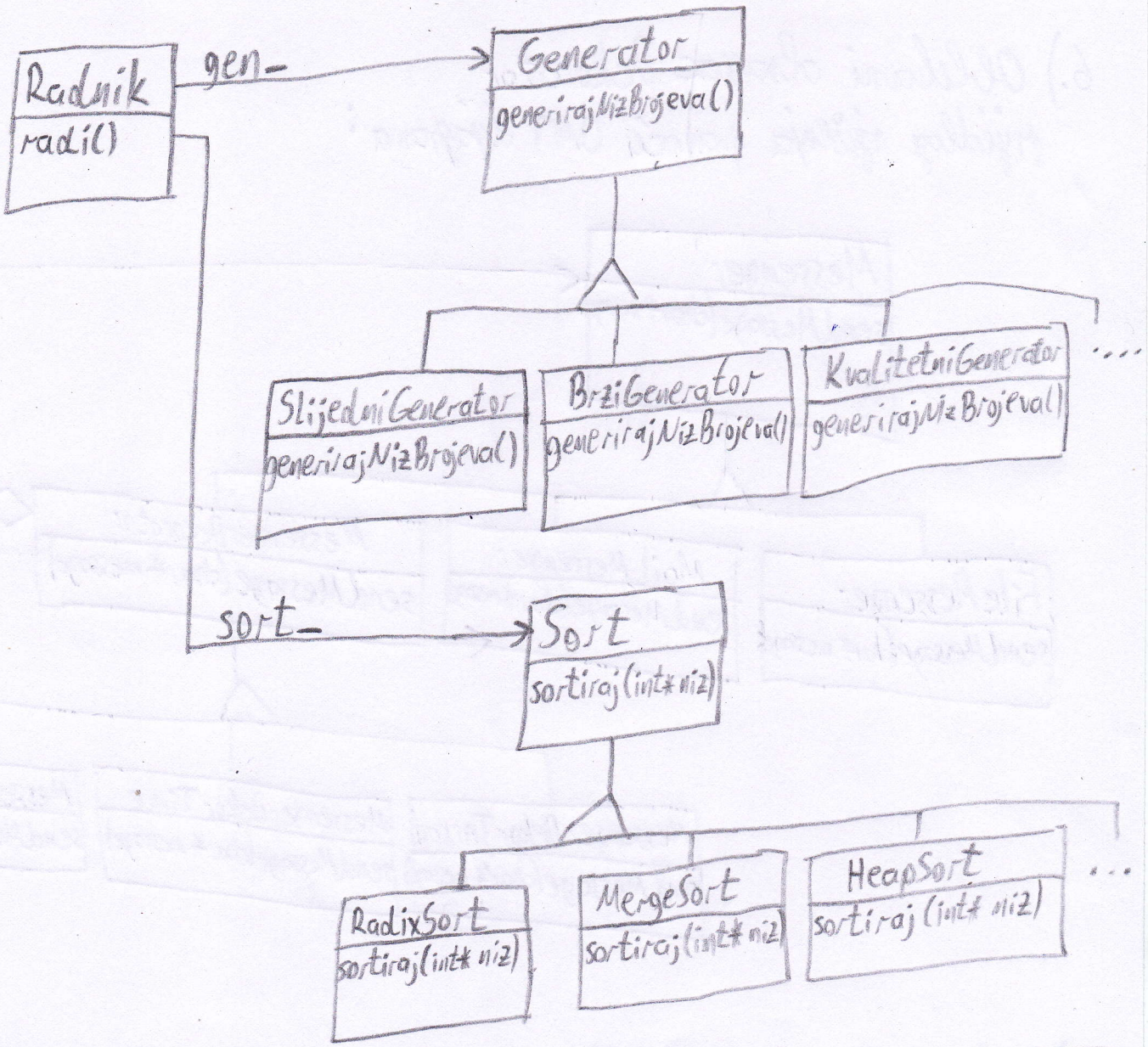
```
Radnik::Radnik(Generator& gen, Sort& sort):
    gen_(gen), sort_(sort){
}
```

```
// Generator.hpp
class Generator{
  virtual int* generirajNizBrojeva()=0;
}
```

```
// Sort.hpp
class Sort{
  virtual void sortiraj(int* niz)=0;
```

- raditi konkretne razrede koji nasljeđuju Generator i Sort i implementiraju željene funkcionalnosti

OMT:

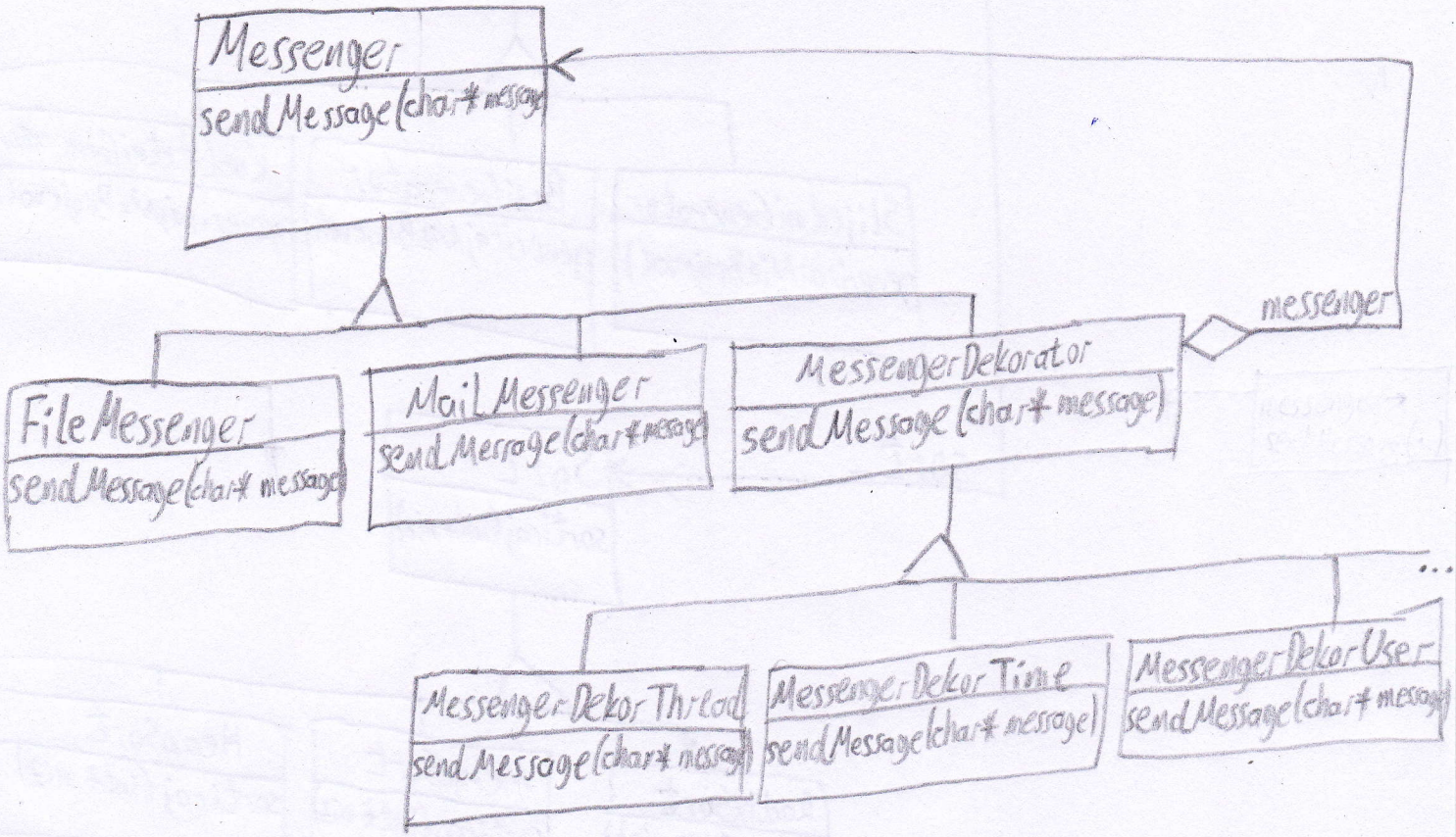


5.) Oblikovni obrazac Promatrač

{sudionici podcrtani}

Komponenta Informator imala bi ulogu subjekta. Komponente grafičkog sučelja bile bi konkretni promatrači koji implementiraju sučelje apstraktni promatrač preko kojeg ih subjekt (Informator) može obavještavati o promijenjenom stanju. Informator bi imao listu pokazivača na apstraktnog promatrača u koju bi se konkretni promatrači nadodavali metodom attach i izbacivali metodom dettach. Pri svakoj promjeni stanja Informator bi pozvao metodu notify() u kojoj se kreće po listi promatrača i poziva njihove metode update().

6.) Oblikuj obrazac Dekorator
prijedlog rješenja pomoću OMT dijagrama:



7.) OMT: