9. travnja 2010.

Zavod za elektroniku, mikroelektroniku
računalne i inteligentne sustave

Oblikovni obrasci u programiranju

prvi međuispit

1. Zadan je razred koji implementira dohvat, obradu i pohranu web stranica kako slijedi:

```
class WebPageProcessor{
public:
  void processUrl(std::string url);
private:
  std::string getPage(std::string url);
  std::string extractTextFromHTML(std::string html);
  void storeTextToDatabase(std::string text);
};

void ProcessWebPage::processUrl(std::string url){
  std::string html = getPage(url);
  std::string text = extractTextFromHTML(html);
  storeTextToDatabase(text);
}
```

Na početku razvoja bilo je važno što prije isporučiti verziju koja ispravno radi za protokol HTTP, stranice u HTML-u, te bazu MySQL. Međutim, sada je potrebno podržati i mogućnost spremanja u bazu Oracle.

Pokušaj poboljšati organizaciju koda kako bi se potrebno proširenje postiglo u skladu s načelima oblikovanja koja su uvedena na predavanjima. Objasni koja načela oblikovanja su zadovoljena u tvom rješenju i kako.

2. Zadan je razred Animal:

```
class Animal {
public:
  virtual ~Animal(){};
  virtual void run()=0;
  virtual void swim()=0;
  virtual void getName()=0;
}
```

Razred Dog izvodi razred Animal, a njegova tablica virtualnih funkcija sadrži pokazivače na destruktor, te izvedbe metoda run i swim.

Razred MyDog izvodi razred Dog:

```
class MyDog : Dog {
public:
  virtual char const* getName(){return "Rex";};
}
```

(a) Skiciraj definiciju razreda Dog.

(b) Skiciraj tablicu virtualnih funkcija za razred MyDog. Za svaki element tablice navedi na što se odnosi.

(c) Navedi pristupe memoriji koji se trebaju dogoditi prilikom izvršavanja funkcije:

```
bool xyzzy(Animal *x){
  return x->getName()!=0;
}
```

(d) Koja vrsta polimorfizma je prisutna u navedenom primjeru?

Zadana su sučelja dvaju komponenata iz hipotetskog programa za vektorsku grafiku. Komponenta **Shape** modelira pojedinačni element crteža, npr. tekst ili kvadrat. Komponenta **ShapeGroup** modelira grupu osnovnih elemenata kojoj se se može pristupati i preko sučelja razreda **Shape**. Metoda **addToGroup** dodaje oblik zadanoj grupi, dok metoda **makeGroup** od oblika stvara **ShapeGroup** s jednim elementom.

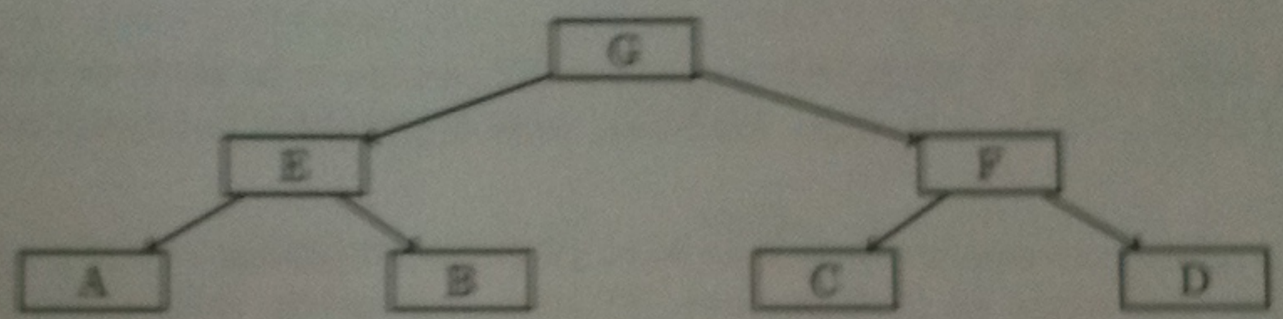
```
// component Shape
class Shape{
public:
  virtual int getX()=0;
  virtual int getY()=0;
public:
  void addToGroup(ShapeGroup& group);
  ShapeGroup makeGroup();
};
```

```
// component ShapeGroup
class ShapeGroup: public Shape{
  // ...
private:
  std::list<Shape*> shapes_;
}
```

Koje su loše strane prikazane organizacije komponenata **Shape** i **ShapeGroup**? Predloži promjene uz pomoć kojih bi se organizacija poboljšala uz zadržavanje iste funkcionalnosti.

Apstraktni osnovni razred **S** definira virtualnu funkciju **drive**. **S** nasljeđuju konkretni razredi **X**, **Y** i **Z**. Razred **C** prima pokazivač na **S** u konstruktoru, sprema ga kao podatkovni član, te ga koristi u metodi **navigate**. Nacrtaj dijagram razreda, te skiciraj implementaciju u C++-u.

Neka je struktura ovisnosti među komponentama A - G programskog sustava zadana kako slijedi:



Pokaži kako bismo sustav mogli preoblikovati na način da u skladu s načelom inverzije ovisnosti smanjimo utjecaj promjena u konkretnim komponentama na glavnu komponentu sustava.

Zadano je sučelje komponente **Database**. Konstruktor otvara komunikaciju s bazom, metoda **getAttribute** dohvaća zadani atribut prema zadanom ključu, dok metoda **query** zadaje SQL upit, te rezultat smješta u polje **queryResult**.

```
class Database{
public:
  Database(std::string path);
  std::string getAttribute(
    std::string key,
    std::string attribute);
  void query(std::string STLquery);
public:
  std::vector<std::string> queryResult;
  std::vector<std::string> keys;
  std::vector<std::string> attributes;
};
```

Predloži promjene koje bi klijentima komponente **Database** olakšale nadogradnju bez promjene.

7. Koje načelo oblikovanja je prekršeno u prikazanom odsječku? Kako bismo mogli poboljšati (navedi barem dva načina)?

```
class Animal{
  virtual char const* sing()=0;
};
class RescueDog: public Animal{
  virtual char const* sing(){
    return "woof";
  }
  virtual void train(){...}
};
```

```
class PetOwner{
  virtual void adopt(Animal* a)=0;
};
class RescueDogOwner: public PetOwner{
  virtual void adopt(Animal* a){
    //PRECONDITION: a is a RescueDog
    ((RescueDog*) a)->train();
  }
};
```