ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за РТИ и Одсек за СИ Београд 23.01.2015.

Испит из Објектно оријентисаног програмирања I

1) *(30 поена)* Одговорити концизно (по једна или две реченице) и прецизно на следећа питања:

а) Да ли се дефиниција објекта класе сме појавити (1) пре декларације и дефиниције дате класе, (2) после декларације, а пре дефиниције дате класе, (3) после дефиниције дате класе?

б) Да ли је механизам операторских функција у језику C++ више сличан механизму преклапања имена (*name overloading*) функција или полиморфног редефинисања (надјачавања, *overriding*) виртуелних функција?

в) Којим редоследом ће се извршити конструктори при конструисању објекта `Y y;` ако постоје следеће дефиниције класа:
```
class X{}; class A: public virtual X{};
class B: public X{}; class C: public virtual X{};
class Y: public A, public virtual B, public C{};
```

2) *(укупно 70 поена)* Реализовати на језику *C++* следећи систем класа (класе опремити оним конструкторима, деструктором и операторима за доделу вредности, који су потребни за безбедно и ефикасно коришћење класа; грешке пријављивати изузецима типа једноставних класа које су опремљене писањем текста поруке):

- *(20 поена)* ***Датум*** садржи целобројне месец и годину, који могу да се дохвате. Може да се утврди једнакост (==) или различитост (!=) два датума. Може да се упише у излазни ток (<<) у облику *месец*`/`*година*.
- Апстрактна ***карта*** садржи аутоматски генерисан јединствен целобројан идентификатор. Може да јој се провери ваљаност на основу задате цене вожње и датума. Не може да се копира ни на који начин. Може да се упише у излазни ток (<<), када се уписује идентификатор.
- *(20 поена)* ***Месечна карта*** има задато име власника (`string`), годину и месец важења. Важење може да се продужи за задати месец и годину. Месечна карта је ваљана ако је важећа задатог датума. Може да се упише у излазни ток (<<) у облику *власник*`(`*карта*`)`*датум*.
- ***Појединачна карта*** има задат уплаћен износ (реалан број), подразумевано 50. Износ на карти може да се допуни задатим износом. Карта је ваљана ако је износ на карти већи од цене вожње задате приликом провере. Тада се износ на карти умањује за задату цену. Може да се упише у излазни ток (<<) у облику *износ*`(`*карта*`)`.
- *(20 поена)* ***Збирка*** садржи неки број показивача на неки тип. Ствара се празна, након чега се подаци додају (`z<<pok`) и узимају (`z>>pok`) појединачно, по адреси. Збирка функционише по принципу кружног (*FIFO*) бафера. Грешка је ако се покуша додавање у пуну збирку или узимање из празне. Збирка може да се испразни и да се одреди колико места је заузето. При копирању се не копирају показани подаци.
- ***Апарат*** за проверу карата садржи збирку од неког броја карата, подразумевано 10. Може да се дода једна карта (`a+=&k`) и да се провери ваљаност свих карата у садржаној збирци, задавањем цене карте и датума. Провера се врши тако што се свака карта проверава, а затим на стандардном излазу испише `Karta` *карта ваљаност*, где *ваљаност* може имати вредност `"valjana"` или `"nije valjana"`. Апарат не може да се копира на било који начин.

(10 поена) Написати на језику *C++* **програм** који створи један апарат за проверу и у њега дода неколико карата, а затим изврши провера. Користити фиксне параметре – није потребно ништа учитати с главног улаза.

НАПОМЕНЕ: **а)** Испит траје **180** минута.

б) Рад се предаје искључиво у вежбанци за испите (-5 поена за неадекватну вежбанку). Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.

в) Водити рачуна о уредности. Нечитки делови текста ће бити третирани као непостојећи. Решења задатака навести по горњем редоследу (-1 поен за лош редослед). Препоручује се рад обичном графитном оловком.

г) Решење задатка не треба раздвајати у датотеке. Довољно је за сваку класу навести дефиницију класе и одмах иза ње евентуалне дефиниције метода које нису дефинисане у самој класи.

д) Резултати колоквијума биће објављени на *Web*-у на адреси:
`http://rti.etf.bg.ac.rs/rti/ir2oo1/index.html`

```
#include <iostream>
#include <string>
using namespace std;

class Datum {
public:
   Datum(int _god, int _mes)
   : god(_god), mes(_mes) { }

   int dohMes() const { return mes; }
   int dohGod() const { return god; }

   friend bool operator==(const Datum &d1,
                          const Datum &d2) {
      return d1.god == d2.god &&
            d1.mes == d2.mes;
   }

   friend bool operator!=(const Datum &d1,
                          const Datum &d2)
   {     return !(d1 == d2);    }

   friend ostream &operator<<(ostream &os,
                          const Datum &d)
   {     os << d.mes << '/' << d.god; }

private:
   int god, mes;
};
```

```
class Karta {
public:
   Karta() = default;
   virtual ~Karta() { }
   virtual bool validiraj(double iznos,
const Datum &d) = 0;
   friend ostream & operator<<(ostream &os,
const Karta &k)
   {     k.pisi(os);     return os;   }
   Karta(const Karta &) = delete;
   Karta(Karta &&) = delete;
   void operator=(const Karta &) = delete;
   void operator=(Karta &&) = delete;

protected:
   virtual void pisi(ostream &os) const
    { os << id; }
private:
   static int posId;
   int id = ++posId;
};
int Karta::posId = 0;
```

```
class Mesecna : public Karta {
public:
   Mesecna(string _ime, int _god, int _mes)
   : ime(_ime), god(_god), mes(_mes) { }
   void produzi(int _god, int _mes)
   { god = _god; mes = _mes; }
   bool validiraj(double iznos,
                  const Datum &d) override {
      return Datum(god,mes) == d;
   }
private:
   void pisi(ostream &os) const override {
      os << ime << '('; Karta::pisi(os);
      os << ") " << god << '/' << mes;
   }
   string ime;
   int god, mes;
};
```

```
class Pojedinacna : public Karta {
public:
   Pojedinacna(double i=50) { dopuna(i);  }
   double dopuna(double i)  {
      iznos += i;
      return iznos;
   }

   bool validiraj(double i, const Datum &d)
override {
      if (i > iznos) return false;
      iznos -= i;
      return true;
   }
private:
   void pisi(ostream &os) const override {
      os << iznos << '('; Karta::pisi(os);
      os << ')';
   }
   double iznos = 0;
};
```

```
class GPuna { };
ostream &operator<<(ostream &os, const
GPuna &m)
{  return os << "Zbirka prepunjena!"; }
```

```
class GPrazna { };
ostream &operator<<(ostream &os, const
GPrazna &m)
{  return os << "Van opsega!"; }
```

```
template<typename T, int K>
class Zbirka
{
public:
   Zbirka() { isprazni(); }
   Zbirka &operator<<(T *t) {
      if (pop == K) throw GPuna();
      mesta[posl] = t;
      pop++;
      posl = (posl + 1) % K;
      return *this;
   }

   void isprazni() {
      prvi = posl = pop = 0;
   }

   Zbirka &operator>>(T *&t) {
      if (pop == 0) throw GPrazna();
      t = mesta[prvi];
      prvi = (prvi + 1) % K;
      pop--;
      return *this;
   }

   int zauzeto() const { return pop; }

private:
   T  *mesta[K];
   int pop, prvi, posl;
};
```

```
template<int K=10>
class Aparat
{
public:
   Aparat() = default;
   Aparat(const Aparat &) = delete;
   Aparat(Aparat &&) = delete;
   void operator=(const Aparat &) = delete;
   void operator=(Aparat &&) = delete;

   Aparat & operator+=(Karta *k) {
      karte << k; return *this;
   }

   void testiraj(double iznos,
                 const Datum &d);

private:
   Zbirka<Karta, K> karte;
};
```

```
template<int K>
void Aparat<K>::testiraj(double iznos,
const Datum &d) {
   while (karte.zauzeto() > 0)  {
      Karta *k;
      karte >> k;
      bool val = k->validiraj(iznos, d);
      cout << "Karta " << *k << " ";
      if (!val)
         cout << "nije valjana.";
      else
         cout << "valjana.";
      cout << endl;
   }
}
```

```
int main() {
   try {
      Aparat<> a;
      Mesecna m1("Marko", 2015, 1);
      Mesecna m2("Petar", 2014, 12);
      Pojedinacna p1(100), p2;
      a += &m1; a += &m2;
      a += &p1; a += &p2;
      a.testiraj(75, Datum(2015, 1));
   }
   catch (GPuna &g) { cout << g << endl; }
   catch (GPrazna &g) { cout << g << endl;
}
}
```

```
Karta Marko(1) 2015/1 valjana.
Karta Petar(2) 2014/12 nije valjana.
Karta 25(3) valjana.
Karta 50(4) nije valjana.
```