ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
*Одсек за СИ* Београд 05.12.2014.

## Други колоквијум из
# Објектно оријентисаног програмирања I

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

**а)** Да ли је могуће написати операторску методу класе `X` за израчунавање резултата израза: (1) `i+x`, (2) `x+i`, где су: `X x; int i;` и зашто?

**б)** Како се могу пренети стварни аргументи конструктору основне класе приликом конструисања објекта изведене класе?

**в)** Да ли деструктор може бити виртуелна функција и зашто?

**2)** *(укупно 70 поена)* Написати на језику *C++* следеће класе (класе опремити оним конструкторима, деструктором и операторима доделе, који су потребни за безбедно и ефикасно коришћење класа):

- *(20 поена)* ***Супстанца*** има назив (`string`) који се задаје приликом стварања и може да се дохвати. Може да упише свој назив у излазни ток (`it<<supstanca`).
- Једноћелијски ***организам*** има стање `ZIV` или `MRTAV` (подразумевано `ZIV`). Може да направи свој клон, да реагује са задатом супстанцом и да у излазни ток упише назив свог стања (`it<<organizam`). Не може да се копира ни премешта, ни иницијализацијом ни доделом.
- *(20 поена)* ***Бактерија*** је једноћелијски организам који има задате назив (`string`), назив супстанце која јој шкоди (`string`) и вероватноћу успешног клонирања (подразумевано 50%). Бактерија памти број успешних клонирања. Након реакције са супстанцом која јој шкоди, бактерија прелази у стање `MRTAV`, и тада се сваки покушај клонирања завршава неуспехом. Може да се упише у излазни ток (`it<<bakterija`) у облику *назив/бр.усп.клонирања*`(`*организам*`)`. Напомена: функција `rand()` генерише случајан цео број у опсегу од 0 до `RAND_MAX`.
- *(20 поена)* ***Колонија*** једноћелијских организама има јединствен, аутоматски генерисан, целобројан идентификатор. Садржи произвољан број организама. Ствара се задавањем једног организма који постаје члан колоније. Колонија може да се размножи, при чему се од сваког организма прави клон и додаје у колонију. Може да реагује са задатом супстанцом, при чему супстанца реагује са сваким организмом у колонији. Колонија не може да се копира ни премешта, ни иницијализацијом ни доделом. Може да се упише у излазни ток (`it<<kolonija`) у облику (*ид.број*)`{`*организам* `|` *организам* `| ... |` *организам*`}`.

*(10 поена)* Написати на језику *C++* програм који направи једну колонију бактерија осетљивих на цефалоспорин, затим неколико пута изврши размножавање колоније и испише је на главном излазу. Потом изврши реакцију колоније са цефалоспорином и поново испише колонију на главном излазу.

---

**НАПОМЕНЕ:** **а)** Колоквијум траје **120** 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 <string>
#include <iostream>
#include <cstdlib>
using namespace std;

inline double rnd()
 { return rand() / (RAND_MAX + 1.0); }

class Supstanca {
public:
  Supstanca(string n) : naziv(n) { }
  string dohNaziv() const { return naziv; }
  friend ostream &operator<<(ostream &os,
                       const Supstanca &s){
    return os << s.naziv;
  }
private:
  string naziv;
};
```

```
class Organizam {
public:
  enum Stanje { ZIV, MRTAV };
  Organizam() = default;
  virtual ~Organizam() { }
  virtual Organizam *kopija() = 0;
  virtual void reaguj(const Supstanca &s)=0;
  friend ostream &operator<<(ostream &os,
    const Organizam &o)
    {o.pisi(os); return os;}
  Organizam(const Organizam &o)=delete;
  void operator=(const Organizam &o)=delete;
protected:
  virtual void pisi(ostream &os) const {
    static char *naziv[]={ "ziv", "mrtav" };
    os << naziv[stanje];
  }
  Stanje stanje = ZIV;
};
```

```
class Bakterija : public Organizam {
public:
  Bakterija(string n, string stet,
  double pp=0.5) : naziv(n), stetna(stet),
                   p(pp), brUspKop(0) {  }
  Bakterija *kopija() override {
    if (stanje==ZIV && p>rnd()){
      brUspKop++;
      return new Bakterija(naziv,stetna,
                           rnd());
    } else return nullptr;
  }
  void reaguj(const Supstanca &s) override {
    if (s.dohNaziv()==stetna) stanje=MRTAV;
  }
private:
  string naziv, stetna;
  double p;
  int brUspKop;
  void pisi(ostream &os) const override {
    os << naziv << '/' << brUspKop << '(';
    Organizam::pisi(os); os << ')';
  }
};
```

```
class Kolonija {
public:
  Kolonija(Organizam *o){prvi=new Elem(o);}
  Kolonija(const Kolonija &) = delete;
  void operator=(const Kolonija &) = delete;
  ~Kolonija();
  void razmnozi();
  void reaguj(const Supstanca &s);
  friend ostream &operator<<(ostream &os,
                       const Kolonija &k);
private:
  struct Elem {
    Elem *sled;  Organizam *org;
    Elem(Organizam *o, Elem *sl = nullptr)
    : org(o), sled(sl) { }
    ~Elem() { delete org; }
  };
  Elem *prvi = nullptr;
   static int poslId;
   int id = ++poslId;
};

int Kolonija::poslId;

Kolonija::~Kolonija() {
  while (prvi) {
    Elem *stari=prvi;
    prvi = prvi->sled;
    delete stari;
  }
}

void Kolonija::razmnozi() { Elem *tek =
prvi;
  while (tek) {
    Organizam *o=tek->org->kopija();
    if (o) prvi = new Elem(o, prvi);
    tek = tek->sled;
  }
}

void Kolonija::reaguj(const Supstanca &s) {
  Elem *tek = prvi;
  while (tek)
  { tek->org->reaguj(s); tek = tek->sled; }
}

ostream &operator<<(ostream &os,
                    const Kolonija &k) {
  os << '(' << k.id << "){";
  Kolonija::Elem *tek = k.prvi;
  while (tek) {
     os << *tek->org;
     if (tek->sled) os << " | ";
     tek = tek->sled;
  }
  return os << '}';
}
```

```
int main() {
  Kolonija k(new Bakterija("streptococcus",
                           "cefalosporin"));

  for (int i = 0; i < 2; i++) k.razmnozi();
  cout << k << endl;
  Supstanca s("cefalosporin");
  cout << "Reakcija sa supstancom: " << s
       << endl;
  k.reaguj(s);
  cout << k << endl;

  return 0;
}
```

```
(1){streptococcus/0(ziv) |
streptococcus/1(ziv) | streptococcus/1(ziv)}
Reakcija sa supstancom: cefalosporin
(1){streptococcus/0(mrtav) |
streptococcus/1(mrtav) |
streptococcus/1(mrtav)}
```