Souvislosti

Academic Honesty
A Guide for Students: Avoid the Need to Cheat!

Student Judicial Affairs, University of California, Berkeley

This guide is intended to provide students with an overview of academic honesty and dishonesty and the process by which dishonest acts will be adjudicated by the campus. It does not constitute campus or University policy. The details of the procedures referenced here can be found, unedited, in the Code of Student Conduct, Rev. July 1998, available from Student Judicial Affairs, 326 Sproul Hall #2432, (510) 643-9069.

The University of California, Berkeley considers academic honesty to be fundamental for each student's intellectual development. To help you derive the full benefit of the educational opportunity provided by Berkeley, this brochure defines academic dishonesty, gives examples of actions that the campus considers dishonest, provides some tips and resources to help you to do your work without resorting to cheating, and explains the student conduct process and possible outcomes.

Definition of Academic Dishonesty

Academic dishonesty is any action or attempted action that may result in creating an unfair academic advantage for yourself or an unfair academic advantage or disadvantage for any other member or members of the academic community.

Academic dishonesty can take the following forms:

Cheating
Cheating is defined as fraud, deceit, or dishonesty in an academic assignment, or using or attempting to use materials, or assisting others in using materials which are prohibited or inappropriate in the context of the academic assignment in question. Here are some examples:

- Copying or attempting to copy from others during an exam or on an assignment.
- Communicating answers with another person during an exam.
- Preprogramming a calculator to contain answers or other unauthorized information for exams.
- Using unauthorized materials, prepared answers, written notes, or concealed information during an exam.
- Allowing others to do an assignment or portion of an assignment for you, including the use of a commercial term-paper service.
- Submitting the same assignment for more than one course without prior approval of all the instructors involved.
- Collaborating on an exam or assignment with any other person without prior approval from the instructor.
- Taking an exam for another person or having someone take an exam for you. ...

Preparing for an exam:

- Review and / or rewrite your notes after each class. Reading them soon after class will make remembering them easier.
- Try condensing your notes to one page. This exercise will help you to organize the main ideas and to select the most important concepts and facts.
- If you don't understand the material, see your professor or the teaching assistant(s) during office hours or make an appointment. The longer you wait, the less time you will have to prepare. ...

R. P. Feynman: To nemyslíte vážně, pane Feynmane!

http://cs.wikipedia.org/wiki/Richard_Feynman

... Byl jsem překvapený, že asi jenom osm studentů z osmdesáti mně odevzdalo první úlohu. Takže jsem jim udělal kázání, že si to musí skutečně *zkusit*, ne jenom sedět v lavicích a pozorovat jak to dělám *já*. ...

... Další věcí, k níž jsem je nedokázal přimět, bylo ptát se. Konečně mně to jeden student vysvětlil: „Když se vás při přednášce něco zeptám, tak mně potom všichni řeknou: ‚Co plýtváš naším časem během přednášky? Snažíme se něco se naučit. A ty ho zdržuješ vyptáváním.'" ...

str.187

... Takže jsem vešel a pod paží jsem nesl učebnici základů fyziky, kterou používali v prvním roce vysoké školy. Tuhle knížku považovali za obzvláště dobrou, protože v ní byly různé typy písma – tučné pro nejdůležitější věci k zapamatování, polotučné pro méně důležité věci a tak dále.

Někdo řekl okamžitě: „Nechcete o té knížce říkat nic špatného, že ne? Člověk, co ji napsal, je tady a všichni si myslí, že je to dobrá učebnice.“ ...

str. 188

... „Když v té knize budu náhodně listovat a položím někde prst a přečtu vám pár vět na té straně, mohu vám ukázat, co mi na ní vadí – není to fyzika, ale *od začátku do konce jenom memorování!* ...

str. 189

... Pak jsem udělal analogii s badatelem o starém Řecku, ...

(badatel) Jde na zkoušku studenta, který promuje, a ptá se ho: „Jaké byly Sokratovy názory na vztah mezi pravdou a krásou?“ – student nedovede odpovědět. Pak se ho zeptá: „Co řekl Sokrates Platónovi na třetím sympoziu?! Student se rozzáří a: „brrrrrrr-ap“ – odvrčí v nádherné řečtině všechno slovo od slova, co Sokrates řekl.

str. 188

// tušíte pointu ?

… Ale to, o čem Sokrates hovořil na třetím sympoziu, byl vztah mezi pravdou a krásou. ...

str. 189

// konec Feynmana

Co byste měli udělat:

uvědomit napsat, říct, si vlastními slovy „o čem to bylo“, například:

- důležitá je má vlastní práce, zkušenost
- důležité je prohovořit věci s přednášejícím, cvičícím, kolegy (ne opisování)
- sám musím určit co je (pro mé poznání) důležité
- slovům musím rozumět, ne učit se jenom nazpaměť

Pověz mi a zapomenu; ukaž mi a já si vzpomenu; ale nech mne se zúčastnit a já pochopím.

Konfucius

Počítače a programování 2

Co je počítač?

Co je programování?

http://www.mapy.cz

Průvodce turistů (Tourist Guide)[1]

Rio de Janeiro je překrásné město. Je tu velké množství míst, které chcete navštívit, ale nevíte kam dřív. Naštěstí, váš přítel Bruno slíbil, že bude vaším průvodcem.

Bruno je však příšerný řidič. Za dopravní přestupky už zaplatil mnoho pokut a tudíž se chce vyvarovat dalšímu placení. Z tohoto důvodu by rád věděl, kde jsou všechny sledovací kamery umístěny, aby mohl řídit opatrněji, když projíždí místem s kamerou. Tyto kamery jsou strategicky rozmístěny po městě do míst, kterými řidič musí projet v případě, že jede z jedné městské čtvrti do druhé. V místě C bude umístěna kamera tehdy a jen tehdy, existují-li dvě místa A a B taková, že všechny silnice z místa A do místa B vedou přes C.

Například předpokládejme, že máme šest čtvrtí (A, B, C, D, E a F) se sedmi obousměrnými silnicemi B – C, A – B, C – A, D – C, D – E, E – F a F – C. Kamera musí být v místě C, protože z místa A do místa E musíte jet přes C. V tomto uspořádání je umístěna kamera pouze v C.

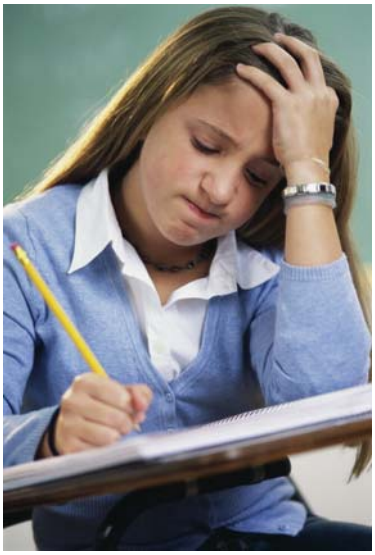
Vaším úkolem je pomoct Brunovi vyhnout se dalším pokutám při cestování s Vámi tím, že určíte podle zadané mapy města všechna místa, kde se kamery nalézají.

Od soumraku do úsvitu (From Dusk Till Dawn)

Vladimír má bílou pleť, velmi dlouhé zuby a je mu 600 let. Žádný div, Vladimír je totiž upír. Být upírem nečinilo nikdy Vladimírovi problém. Je znamenitým lékařem, který si zásadně bere noční služby, díky čemuž má mezi svými kolegy mnoho přátel. Ovládá velice působivý trik, který předvádí na večírcích: dokáže stanovit krevní skupinu pouhým ochutnáním krve. Vladimír miluje cestování, ale díky tomu, že je upír, se musí potýkat se třemi překážkami.

1. Může cestovat pouze vlakem, protože si sebou musí vzít svou rakev. Mimochodem, protože investoval značné množství peněz do dlouhodobých cenných papírů, může si dovolit jet první třídou.
2. Může cestovat jen od soumraku do úsvitu, konkrétně od 18:00 do 6:00. Během dne musí zůstat na nádraží v rakvi.
3. Musí si sebou vzít něco k snědku. Na každý den potřebuje jeden litr krve, který pije vždy v poledne (12:00) ve své rakvi.

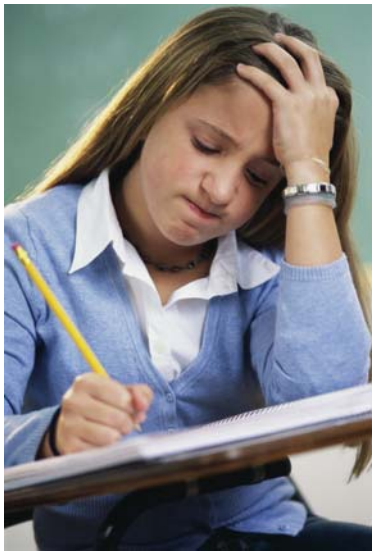
Pomozte Vladimírovi nalézt cestu z výchozího do cílového města tak, aby mohl cestovat s minimem krve (tj. časově nejkratší) a vyhnul se tak všetečným otázkám: "A na co vlastně potřebujete tolik krve?"



problém, úkol, ...

?

program + počítač



problém, úkol, ...

Algorithms + Data Structures = Programs
Niklaus Wirth – autor jazyka Pascal

program + počítač

Programovací jazyk

Fortran, Algol60, Pascal, C, C++, Java, C#

Datové typy

Příkazy

Program pro tentýž problém můžeme zapsat ve více jazycích!

Jazyk je (pouze) vyjadřovací prostředek!

Datový typ

je množina hodnot a soubor operací nad nimi

čísla, logické hodnoty, znaky, řetězce, …

Příkazy

řídí postup výpočtu

přiřazení
alternativy (if, ...)
opakování / cyklu (while, ...)

Mohu jakýkoliv algoritmus zapsat uvedenými příkazy ?

Počítač

procesor

vykonává instrukce nad daty
v registrech *8, 16, 32, 64, … bitů*

instrukce

přesunu *move*
aritmeticko-logické
řídící *skok – JMP (jump), …*

(hlavní) paměť
obsahuje instrukce + data programu

Jak se příkazy programu transformují na instrukce?

Nemáme instrukce *přiřazení, if, while, ...* !

přiřazení

instrukce MOV

a = a + 1;

```
MOV    R1, A   // přesun hodnoty do registru
INC    R1      // inkrementace
MOV    A, R1   // přesun do paměti
```

příkaz if

instrukce porovnání a podmíněného skoku

```
if (i < 10)
   ... ;

          CMP    I, 10     // porovnej - compare
          JGE    L         // je-li i větší nebo rovno 10
                           // skoč na L
          ...               // vykonej co je na …
L:
```

příkaz opakování

instrukce porovnání a podmíněného skoku

```
while (i < 10)
   ... ;

L1:     CMP I,10    // porovnej - compare
        JGE  L2     // je-li i větší nebo rovno 10
                    // skoč na L2
        ...         // vykonej co je na …
        JMP  L1
L2:
```

```
program opakuj10krat;
var i:integer;
begin
  i:=0;
  while i<10 do i:=i+1
end.
```

```
                        // i:=0;
     XOR     R1, R1     // operací XOR vynuluj registr R1
     MOV     I, R1      // na paměťové místo pro proměnnou I
                           // přesuň hodnotu v R1

                        // while i<10 do i:=i+1
L1:  CMP     I, 10      // porovnej hodnotu v I a 10
     JGE     L2         // byla-li hodnota I větší nebo rovná 10
                           // skoč na navěstí L2
     MOV     R1, I      // do registru R1 přesuň hodnotu
                           // z paměťového místa pro I
     INC     R1         // zvětši hodnotu registru R1 o 1
     MOV     I, R1      // na paměťové místo pro proměnnou I
                           // přesuň hodnotu v R1
     JMP     L1          // skoč na instrukci s návěstím L1
L2:
```

Jiný stroj – jiné registry, jiné instrukce

jiný překladač, pro tentýž jazyk

Nápad:

- **definujme „univerzální“ instrukce, do kterých budeme překládat náš program – virtuální stroj**
- **pro každý reálný stroj stačí napsat „jenom“ vykonání (interpretaci) těchto instrukcí jeho instrukcemi**

Pozn.: bytecode, JVM

Jaké registry ?

žádné – zásobníkový stroj

operace se vykonávají nad zásobníkovou pamětí

data jsou přístupná v opačném pořadí než byla zapsána

5*(6+4)

zapiš 5	**5**	
zapiš 6	**5 6**	
zapiš 4	**5 6 4**	
sečti	**5 10**	**// sčítance jsou 4 a 6**
		// sečti a zapiš
vynásob	**50**	**// součnitelé jsou 5 a 10**
		// vynásob a zapiš

```
class Opakuj10Krat {
  public static void main(String[] args) {
    int i=0;
    while (i<10)
       i++;
  }
}
```

```
                    // i:=0;
 0   iconst_0       // vlož do zásobníku konstantu 0 typu int
 1   istore_1       // ulož hodnotu typu int ze zásobníku do
                       // proměnné 1 (i)
 2   iload_1        // vlož do zásobníku hodnotu typu int
                       // z proměnné 1 (i)

                    // while (i<10) i++;
 3   bipush 10      // vlož do zásobníku hodnotu 10 typu int
 5   if_icmpge 14   // porovnej poslední dvě hodnoty vložené
                       // do zásobníku
                       // a je-li i větší nebo rovno 10 skoč na
                       // index 14
 8   iinc 1 1       // inkrementuj proměnnou 1 (i)
11   goto 2         // skoč na index 2
14   return         // návrat
```

problém

↓ ***vymyslet***

algoritmus + data

↓ ***zvolit jazyk a zapsat***

program

↓ ***přeložit***

počítač

Co je programování ?

datový typ v jazycích

hodnoty a operace odvozené z jednoduchých *matematických abstrakcí* – celé číslo, řetězec, vektor, matice

***život* je bohatší**

celky jsou opsány:

- **několika hodnotami**
- **akcemi, které nad nimi pracují – obecně**
- **více operací – procedura, funkce, metoda**

Pozn.: Volba hodnot a akcí je „programování“ a závisí na řešeném problému

celek: **auto**

hodnoty: **obsah nádrže, spotřeba, …**
akce: **ujetá vzdálenost, …**

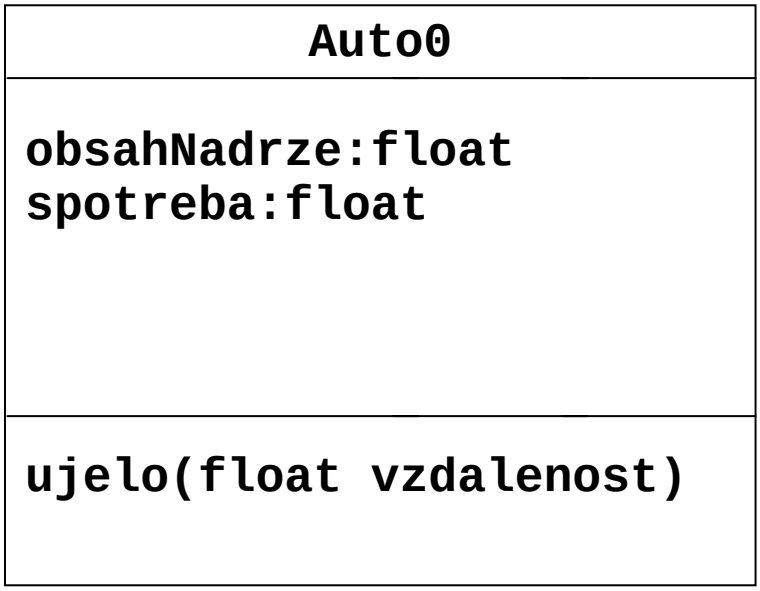
hodnoty a akce jsou definovány třídou

hodnoty - proměnné
akce - metody

```
class Auto0 {
  float obsahNadrze;
  float spotreba;

  void ujelo(float vzdalenost) {
    obsahNadrze = obsahNadrze - vzdalenost /
                    100.0F * spotreba;
  }
}
```

graficky:



opis auta – třída Auto0

konkrétním jednotlivým celkům, říkáme instance třídy / objekty

instanci třídy vytvoří operátor new

`new Auto0()`

vrátí referenci / odkaz na objekt

uložíme ji na (referenční) proměnnou na objekty třídy `Auto0`

```
Auto0 a;
a = new Auto0();
```

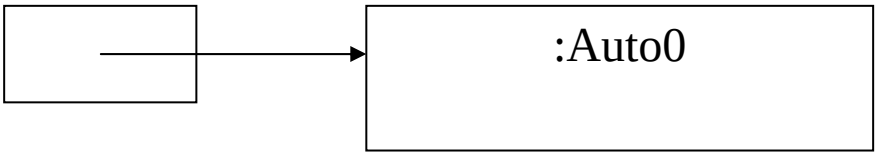
graficky

```
Auto0 a;
```

a

```
a = new Auto0();
```

a



také, jako obvykle

```
Auto0 a = new Auto0();
```

klientská třída / program

práce s instancí(emi) jiné třídy (tříd)

referenční proměnná.*xxx*

xxx proměnná
metoda

```
class MojeAuto {

  public static void main(String[] args) {

    Auto0 a;
    a = new Auto0();
    System.out.println("Spotreba a je
                        "+a.spotreba+" l/100km");
    System.out.println("Obsah nadrze a je
                          "+a.obsahNadrze+" l");

    a.spotreba = 5.0F;
    a.obsahNadrze = 20.0F;
    System.out.println("Spotreba a je
                        "+a.spotreba+" l/100km");
    System.out.println("Obsah nadrze a je
                          "+a.obsahNadrze+" l");

    a.ujelo(200.0F);
    System.out.println("Obsah nadrze a je
                          "+a.obsahNadrze+" l");
  }
}
```

imperativně (procedurálně) – NE !!!

data a metody (procedury) jsou odděleny

```
class Auto0 {
  float obsahNadrze;
  float spotreba;
}

class MojeAuto {

  static void ujelo(Auto0 a, float vzdalenost){
    a.obsahNadrze = a.obsahNadrze -
                  vzdalenost/100.0F*a.spotreba;
  }

  public static void main(String[] args) {
    Auto0 a = new Auto0();
    System.out.println("Spotreba a je "
                       +a.spotreba+" l/100km");
    System.out.println("Obsah nadrze a je "
                          +a.obsahNadrze+" l");

    a.spotreba = 5.0F;
    a.obsahNadrze = 20.0F;
    System.out.println("Spotreba a je "
                       +a.spotreba+" l/100km");
    System.out.println("Obsah nadrze a je "
                          +a.obsahNadrze+" l");

    ujelo(a,200.0F);
    System.out.println("Obsah nadrze a je "
                          +a.obsahNadrze+" l");
  }
}
```

konstruktor

metody, které mají stejný název jako třída, jsou konstruktory
umožňují inicializovat datové členy, členské proměnné

konstruktor je možné přetížit, což umožňuje inicializovat objekt různými způsoby

přetížené metody mají stejné jméno, ale liší se počtem nebo typem nebo pořadím formálních parametrů

Pozn.1: V předcházejícím příkladě se použil tzv. *implicitní* konstruktor `Auto0()`

Pozn.2: Má-li třída konstruktor s parametry, není vytvořen implicitní konstruktor

```
class Auto {
  final float kapacitaNadrze;
  float obsahNadrze;
  float spotreba;

  Auto(float kapacitaNadrze, float spotreba) {
    this.kapacitaNadrze = kapacitaNadrze;
    this.spotreba = spotreba;
  }

  Auto(Auto a) {
    kapacitaNadrze = a.kapacitaNadrze;
    spotreba = a.spotreba;
  }

  Auto() {
    kapacitaNadrze = 24.0F;
    spotreba = 8.0F;
  }

  float doplnNadrz() {
    float doplneni;
    doplneni = kapacitaNadrze - obsahNadrze;
    obsahNadrze = kapacitaNadrze;
    return doplneni;
  }

  float dojezd() {
    float dojede = obsahNadrze / spotreba *
                       100.0F;
    return dojede;
  }

  void ujelo(float vzdalenost) {
    obsahNadrze = obsahNadrze - vzdalenost /
                      100.0F * spotreba;
  }
}
```

```
class MojeAuta {

  public static void main(String[] args) {

    Auto a0 = new Auto();
    System.out.println("Kapacita nadrze a0 je"
                       +a0.kapacitaNadrze+" l");
    System.out.println("Spotreba a0 je "
                       +a0.spotreba+" l/100km");
    System.out.println("Obsah nadrze a0 je "
                       +a0.obsahNadrze+" l");

    Auto a1 = new Auto(35.0F, 5.5F);
    System.out.println("Kapacita nadrze a1 je "
                       +a1.kapacitaNadrze+" l");
    System.out.println("Spotreba a1 je "
                       +a1.spotreba+" l/100km");
    System.out.println("Obsah nadrze a1 je "
                       +a1.obsahNadrze+" l");

    /* promenna kapacitaNadrze je final a
       napriklad zvetsit ji nejde
       a1.kapacitaNadrze = 50.0F;
       System.out.println("Kapacita nadrze a1
       je "+a1.kapacitaNadrze+" l");
    */

    a1.spotreba = 6.0F;  // spotreba a1 stoupla
    System.out.println("Spotreba a1 je "
                       +a1.spotreba+" l/100km");

    Auto a2 = new Auto(45.0F, 10.5F);
    System.out.println("Kapacita nadrze a2 je "
                       +a2.kapacitaNadrze+" l");
    System.out.println("Spotreba a2 je "
                       +a2.spotreba+" l/100km");
    System.out.println("Obsah nadrze a2 je "
                          +a2.obsahNadrze+" l");
```

```
/* dalsi moje auto a3 ma nadrz jako a2 a
   spotrebu jako a1 */
Auto a3 = new Auto(a2.kapacitaNadrze,
                   a1.spotreba);
System.out.println("Kapacita nadrze a3 je "
                  +a3.kapacitaNadrze+" l");
System.out.println("Spotreba a3 je "
                  +a3.spotreba+" l/100km");


/* moje auto a4 je stejne jako a2 */
Auto a4 = new Auto(a2);
System.out.println("Kapacita nadrze a4 je "
                  +a4.kapacitaNadrze+" l");
System.out.println("Spotreba a4 je "
                  +a4.spotreba+" l/100km");

// autu, kteremu rikam a1, nekdy rikam a11
Auto a11 = a1;
System.out.println("Kapacita nadrze a11 je"
                 +a11.kapacitaNadrze+" l");
System.out.println("Spotreba a11 je "
                  +a11.spotreba+" l/100km");

/* zase mu vzrostla spotreba */
a1.spotreba = 7.0F;
System.out.println("Kapacita nadrze a1 je "
                  +a1.kapacitaNadrze+" l");
System.out.println("Spotreba a1 je "
                  +a1.spotreba+" l/100km");
System.out.println("Kapacita nadrze a11 je"
                  +a11.kapacitaNadrze+"l");
System.out.println("Spotreba a11 je "
                  +a11.spotreba+"l/100km");
```

```
    /* autu a1 doplnime nadrz */
    System.out.println("Nacerpali jsme "
                       +a1.doplnNadrz()+ " l");

    /* zjistime dojezd a1 */
    System.out.println("Dojezd je "
                       +a1.dojezd()+" km");

    /* auto a1 ujelo 200 km */
    a1.ujelo(200.0F);

    /* zjistime obsah nadrze */
    System.out.println("Obsah nadrze je "
                       +a1.obsahNadrze+" l");

    /* a muzeme jet kousek dal, doplnit nadrz,
       zmenit spotrebu, zjistit dojezd,... */

    /* prodal jsem auto a4 */
    a4 = null;

    /* prodal jsem auto a1 */
    a1 = null;
    a11 = null;
  }
}
```

objekt parametrem metody objektu téže třídy

```
class Auto {
  final float kapacitaNadrze;
  float obsahNadrze;
  float spotreba;

  Auto(float kapacitaNadrze, float spotreba) {
    this.kapacitaNadrze = kapacitaNadrze;
    this.spotreba = spotreba;
  }

  float doplnNadrz() {
    float doplneni;
    doplneni = kapacitaNadrze - obsahNadrze;
    obsahNadrze = kapacitaNadrze;
    return doplneni;
  }

  float dojezd() {
    float dojede = obsahNadrze / spotreba *
                                          100.0F;
    return dojede;
  }

  void ujelo(float vzdalenost) {
    obsahNadrze = obsahNadrze -
              vzdalenost/100.0F*spotreba;
  }

  float rozdilSpotreby (Auto a) {
    return this.spotreba - a.spotreba;
  }
}
```

```
class MojeAuta {

  public static void main(String[] args) {

    Auto a0 = new Auto(40.0F, 6.5F);
    System.out.println("Kapacita nadrze a0 je
           "+a0.kapacitaNadrze+" l");
    System.out.println("Spotreba a0 je
           "+a0.spotreba+" l/100km");
    System.out.println("Obsah nadrze a0 je
           "+a0.obsahNadrze+" l");

    Auto a1 = new Auto(35.0F, 5.5F);
    System.out.println("Kapacita nadrze a1 je
           "+a1.kapacitaNadrze+" l");
    System.out.println("Spotreba a1 je
           "+a1.spotreba+" l/100km");
    System.out.println("Obsah nadrze a1 je
           "+a1.obsahNadrze+" l");

    float rozdil = a0.rozdilSpotreby(a1);
    System.out.println("Rozdil spotreby a0 a a1
           je "+rozdil+" l");
  }
```

Doplňte si auto o počet míst a počet pasažérů, metody nástup a výstup !

co dál?

http://www.sigcse.org/cc2001/

CS112I. Data Abstraction

Continues the introduction of programming begun in CS111I, with a particular focus on the ideas of data abstraction and object-oriented programming. Topics include recursion, programming paradigms, principles of language design, virtual machines, object-oriented programming, fundamental data structures, and an introduction to language translation.

Prerequisites: CS111I; discrete mathematics at the level of CS105 is also desirable.

Syllabus:

- Review of elementary programming
- Recursion: The concept of recursion; recursive specification of mathematical functions (such as factorial and Fibonacci); simple recursive procedures (Towers of Hanoi, permutations, fractal patterns); divide-and-conquer strategies; recursive backtracking; implementation of recursion
- Introduction to computational complexity: Asymptotic analysis of upper and average complexity bounds; big-O notation; standard complexity classes; empirical measurements of performance
- Fundamental computing algorithms: O(N log N) sorting algorithms (Quicksort, heapsort, mergesort); hashing, including collision-avoidance strategies; binary search trees
- Programming languages: History of programming languages; brief survey of programming paradigms (procedural, object-oriented, functional)
- Fundamental issues in language design: General principles of language design, design goals, typing regimes, data structure models, control structure models, abstraction mechanisms
- Virtual machines: The concept of a virtual machine, hierarchy of virtual machines, intermediate languages

- Object-oriented programming: Object-oriented design; encapsulation and information-hiding; separation of behavior and implementation; classes, subclasses, and inheritance; polymorphism; class hierarchies; collection classes and iteration protocols; fundamental design patterns
- Fundamental data structures: Linked structures; implementation strategies for stacks, queues, hash tables, graphs, and trees; strategies for choosing data structures
- Introduction to language translation: Comparison of interpreters and compilers; language translation phases (lexical analysis, parsing, code generation, optimization); machine-dependent and machine-independent aspects of translation

Co bude v PPA2 ?

Jak na to ?

Zadání:

http://www.kiv.zcu.cz/~netrvalo/vyuka/ppa2/portal/cviceni/cv02.pdf