Seminář Java
I
2005/2006

Radek Kočí

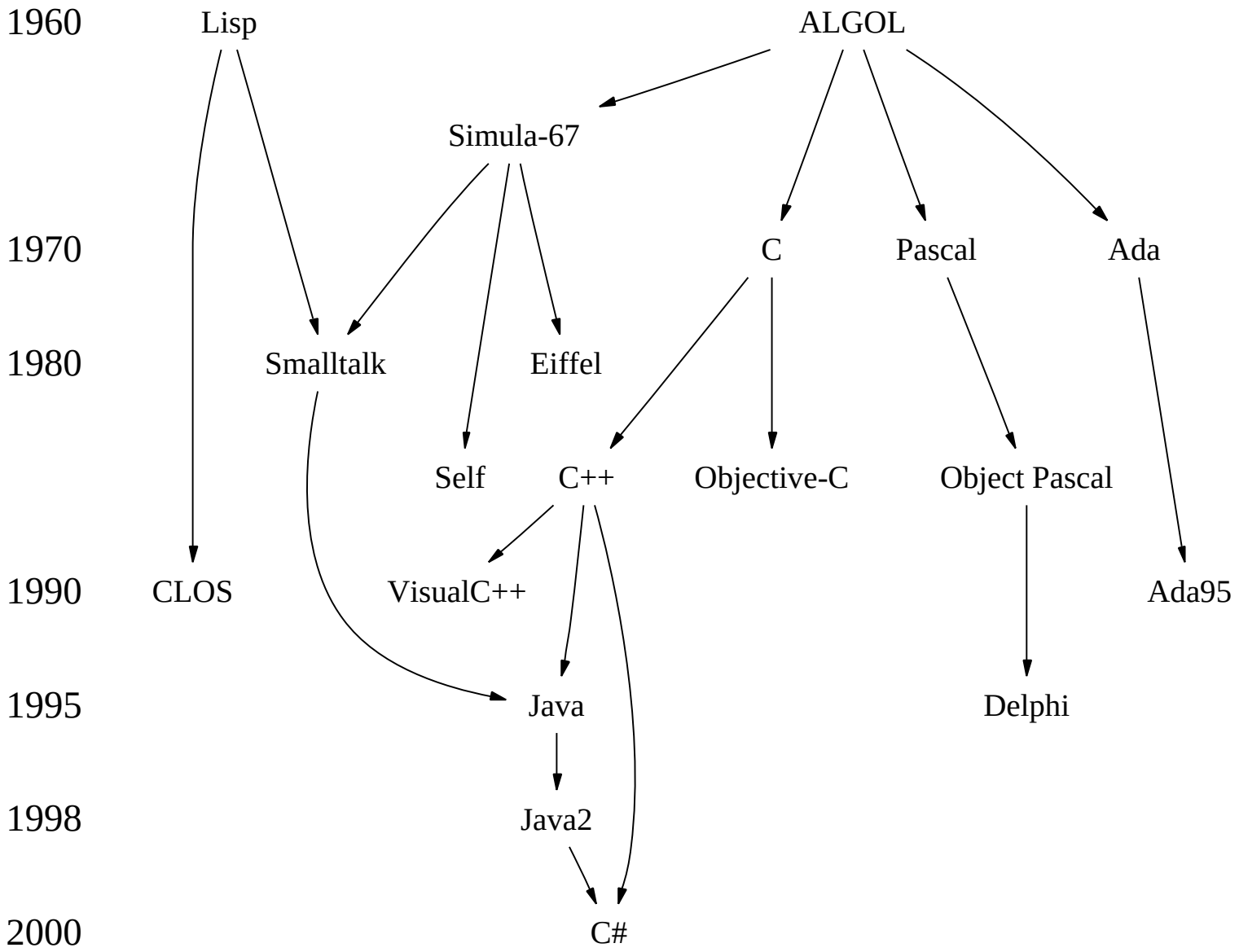
Téma přednášky

- Organizace semináře
- Úvod do programovacího jazyka Java
- Distribuce
- Základní principy OOP

Informace, studijní materiály

- Stránky předmětu
  - `http://www.fit.vutbr.cz/study/courses/IJA/`
  - zadání úkolů, informace
  - konzultace
  - studijní materiály

Přehled jazyků



Programovací jazyk Java

Základní charakteristika

- univerzální (není určen výhradně pro specifickou aplikační oblast)
- objektově-orientovaný
- statická typová kontrola
- jednodušší než C++ (méně syntaktických konstrukcí, méně nejednoznačností v návrhu)
- v průměru vyšší produktivita programátorské práce v Javě než v C++
- Java Virtual Machine – JVM (program v Javě je meziplatformně přenositelný na úrovni zdrojového i přeloženého kódu)
- automatické odklizení nepoužitelných objektů (automatic garbage collection)

Programovací jazyk Java

Základní charakteristika

- zdarma dostupné nezměrné množství knihoven pro různorodé aplikační oblasti, např. na SourceForge a tisících dalších místech
- k dispozici je řada kvalitních vývojových prostředí (i zdarma) - NetBeans, JBuilder, Visual Age for Java, Eclipse, IDEA
- reálným soupeřem je (Microsoft) C# (zatím převážně na platf. Windows)

Srovnání (názory)

- Java vs. C++ (`http://c2.com/cgi/wiki?JavaVsCpp`)
- Java vs. Smalltalk (`http://c2.com/cgi/wiki?JavaVsSmalltalk`)

Programovací jazyk Java

Využití Javy

- vícevláknové aplikace (multithreaded applications)
- škálovatelné výkonné aplikace běžící na serverech (Java Enterprise Edition)
- aplikace na přenosných a vestavěných zařízeních (Java Micro Edition)
- webové aplikace (servlety, JSP) - konkurence proprietárním ASP, SSI, CGI
- zpracování semistrukturovaných dat (XML)
- přenositelné aplikace s GUI
- aplikace distribuované po síti (applety nebo Java Web Start)

Programovací jazyk Java

Typy aplikací

- Konzolové aplikace
  - jednoduchá textová konzole
- GUI aplikace
- Applety
  - běží v HTML prohlížečích
  - mají silná bezpečnostní omezení

Java – platforma

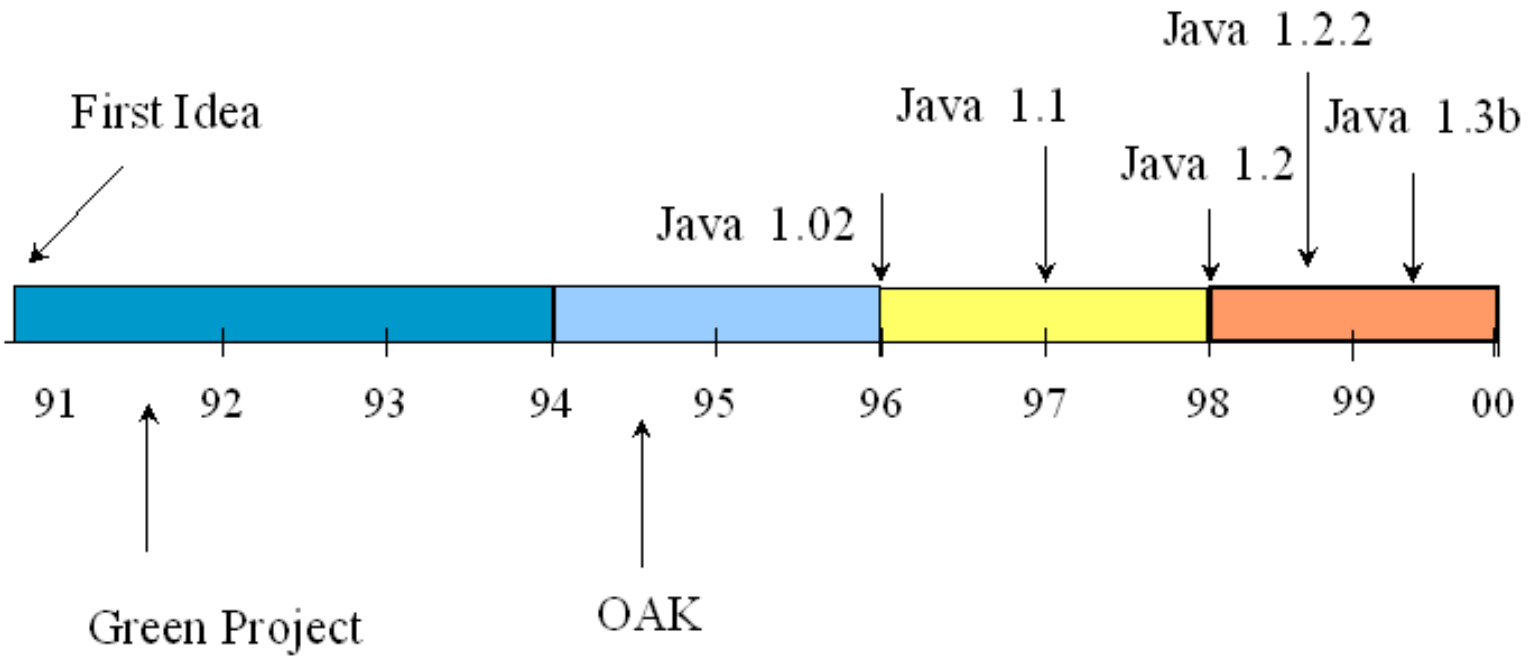
Java platformu tvoří:

- Java Virtual Machine (JVM)
- překladač a další vývojové nástroje
- Java Core API (základní knihovna tříd)

Java je tedy dána...

- definicí jazyka (Java Language Definition) - syntaxe a sémantika jazyka
- popisem chování JVM
- popisem Java Core API

Java – vývoj


First Idea
Java 1.02
Java 1.1
Java 1.2
Java 1.2.2
Java 1.3b
91
92
93
94
95
96
97
98
99
00
Green Project
OAK

Specifikace a implementace Javy

- Specifikace Javy
  - např. Java 2 Standard Edition, v1.4
  - např. Java 2 Enterprise Edition, v1.4

- Implementace Javy
  - např. Java 2 Software Development Kit, v1.4.2 - obsahuje vývojové nástroje
  - např. Java 2 Runtime Enviroment, v1.4 - obsahuje jen běhové prostředí pro spouštění hotových přeložených pg.

Verze Javy

Hrubé členění

- verze Java (před Java 2)
- verze Java 2

Číslování verzí:

- major číslo (např. Java 2, v1.4)
  - při změně major čísla se může měnit Core API a někdy i jazyk
- minor číslo (např. Java 2, v1.4.2)
  - změnu minor (třetího) čísla doprovází jen odstraňování chyb
- ke změně prvního čísla zatím nedošlo ... (?)

Aktuální verze

- Java 2 Standard Edition v1.5.0 (We have changed the version of this release from 1.5.0 to 5.0 to better reflect the level of maturity, stability, scalability and security built into J2SE.)
- aktuálně vždy na webu `http://java.sun.com`

Verze Javy

version	*code name*	*release date*
JDK 1.1.4	Sparkler	Sept 12, 1997
JDK 1.1.5	Pumpkin	Dec 3, 1997
JDK 1.1.6	Abigail	April 24, 1998
JDK 1.1.7	Brutus	Sept 28, 1998
JDK 1.1.8	Chelsea	April 8, 1999
J2SE 1.2	Playground	Dec 4, 1998
J2SE 1.2.1	*(none)*	March 30, 1999
J2SE 1.2.2	Cricket	July 8, 1999
J2SE 1.3	Kestrel	May 8, 2000
J2SE 1.3.1	Ladybird	May 17, 2001
J2SE 1.4.0	Merlin	Feb 13, 2002
J2SE 1.4.1	Hopper	Sept 16, 2002
J2SE 1.4.2	Mantis	June 26, 2003
J2SE 5.0 (1.5.0)	Tiger	Sept 29, 2004

Java Forum 2005
Java Technology is 10

Distribuce Javy

Podmínky získání a používání

- používání Javy pro běžný vývoj (i komerční) je zdarma
- redistribuce javového vývojového prostředí je dovolena pouze s licencí od Sunu
- redistribuce javového běhového prostředí je možná zdarma
- distribuce vyvíjí Sun Microsystems Inc. (Javasoft) i další výrobci (např. IBM) a tvůrci Open Source

Stažení distribuce Sun

- `http://java.sun.com` (pro Windows, Solaris, Linux)
- dokumentace se stahuje z téhož místa, ale samostatně (nebo lze číst z WWW)
- celkově vývojové prostředí J2SDK 1.4.2 vč. dokumentace zabere cca 220 MB na disku
- velikost operační paměti - doporučeno 128 MB (a více :-))

Obsah vývojové distribuce Javy

Obsah adresářů

- `bin` – vývojové nástroje (Development Tools) určené k vývoji, spouštění, ladění a dokumentování programů v Javě.
- `jre` – běhové prostředí Javy (Java Runtime Environment); obsahuje Java Virtual Machine (JVM), knihovnu tříd Java Core API a další soubory potřebné pro běh programů v Javě
- `lib` – přídavné knihovny (Additional libraries) jsou další knihovny nutné pro běh vývojových nástrojů
- `demo` – ukázkové applety a aplikace (Demo Applets and Applications); příklady zahrnují i zdrojový kód

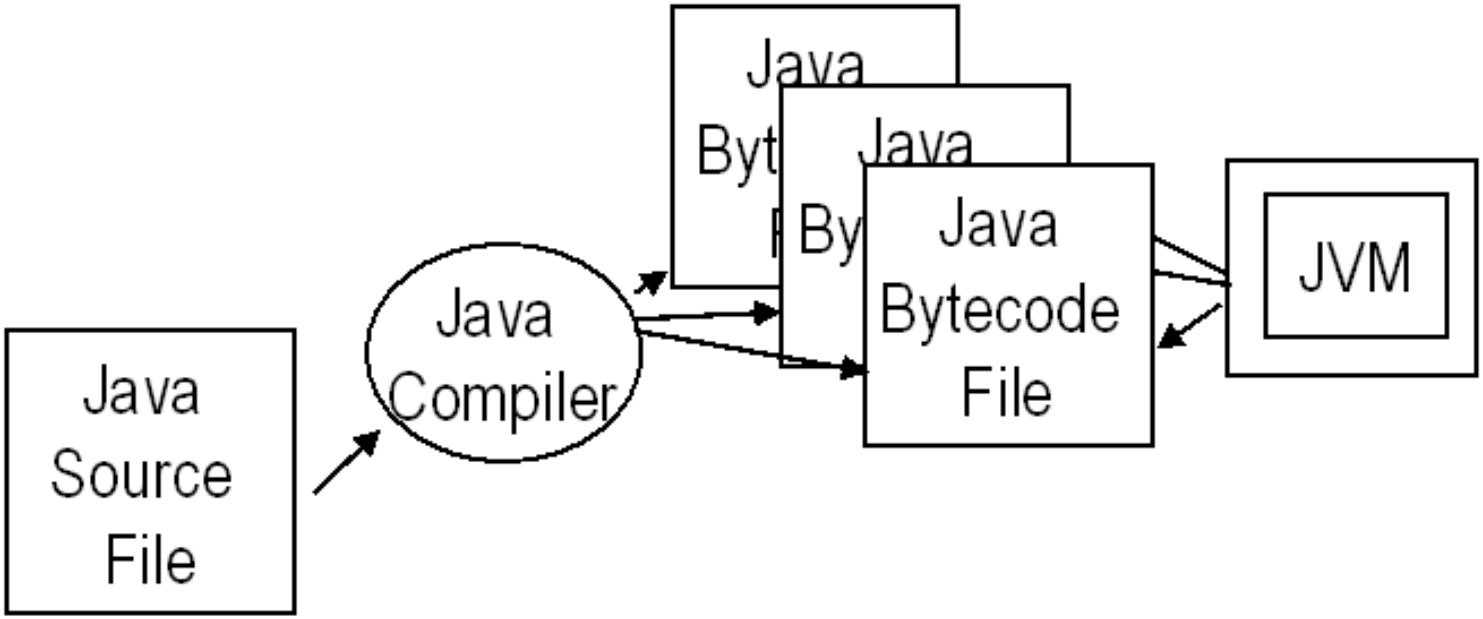
Nástroje ve vývojové distribuci

Pod Windows jsou to .exe soubory umístěné v podadresáři bin

- `java` – spouštěč (přeloženého bajtkódu)
- `javac` – překladač (.java -> .class)
- `javadoc` – generátor dokumentace API
- `jar` – správce archivů JAR (sbalení, rozbalení, výpis)
- `jdb` – debugger
- `appletviewer` – referenční prostředí pro spouštění appletů

Java Virtual Machine

- Překladač generuje byte-kód pro JVM
- JVM interpretuje byte-kód
- Optimalizace (JIT)



Distribuce Javy na FIT

- `sun00.fit.vutbr.cz` – `sun11.fit.vutbr.cz`
  - J2SE 1.4.2

- `merlin.fit.vutbr.cz`
  - J2SE 5.0 (1.5.0)

Praktické informace

Co je nutné udělat

- Cesty ke spustitelným programům (`PATH`) musejí obsahovat i adresář `$JAVA_HOME/bin`

Co je vhodné udělat

Systémové proměnné by měly obsahovat:

- `JAVA_HOME` = kořenový adresář instalace Javy, např. `JAVA_HOME=/usr/local/j2sdk1.4.2`
- `CLASSPATH` = cesty ke třídám (podobně jako v `PATH` jsou cesty ke spustitelným souborům), např. `CLASSPATH=$HOME/java`

Proces Objektově orientované tvorby

- Objektově orientovaná analýza
  - Porozumění řešené doméně
- Objektově orientovaný návrh
  - Návrh řešení, model domény (struktura, aktivity)
- Objektově orientované programování
  - Implementace řešení
- Dobrý návrh tvoří 2/3 práce ...
- Je to proces, ne vodopád ...
- OOA je jazykově nezávislá

Základy objektové orientace

- Objektově orientovaný přístup k modelování a vývoji systémů
  - kolekce vzájemně komunikujících objektů
  - objekt = abstrakce doménově specifických entit
  - objekt = sloučení dat a funkcionality do uzavřené jednotky
  - vykazuje vyšší stabilitu navrhovaných prvků z pohledu měnících se požadavků
  - soubor objektově orientovaných prostředků (objekty, třídy, UML, . . .) a metodologie (např. RUP)
  - *Objektový návrh nutně neimplikuje objektovou implementaci!*

- Vlastnosti objektové orientace
  - Abstrakce (abstraction)
  - Zapouzdření (encapsulation)
  - Polymorfismus (polymorphism)
  - Dědičnost (inheritance) – Hierarchie (hierarchy)

Základní pojmy – Atributy objektu

Atribut vs. proměnná objektu

- reprezentují data zapouzdřená v objektu
- *Proměnná objektu*
  - implementační pohled
  - získání/nastavení atributu $\Rightarrow$ lze (teoreticky) přímo (nedoporučuje se)
- *Atribut objektu*
  - pohled z vyšší úrovně
  - atribut je vlastnost objektu
  - atribut není proměnná (i když je tak většinou realizován)
  - atribut datum (dd/mm/rr) $\Rightarrow$ počet sekund od LP 1970
  - atribut objem $\Rightarrow$ součin tří hodnot
  - získání/nastavení atributu $\Rightarrow$ operace

Základní pojmy – Stav a identita objektu

Stav objektu

- stavová množina je reprezentována množinou hodnot atributů objektu
- aktuální hodnoty všech atributů představují aktuální stav
- v každém okamžiku je objekt v definovatelném stavu

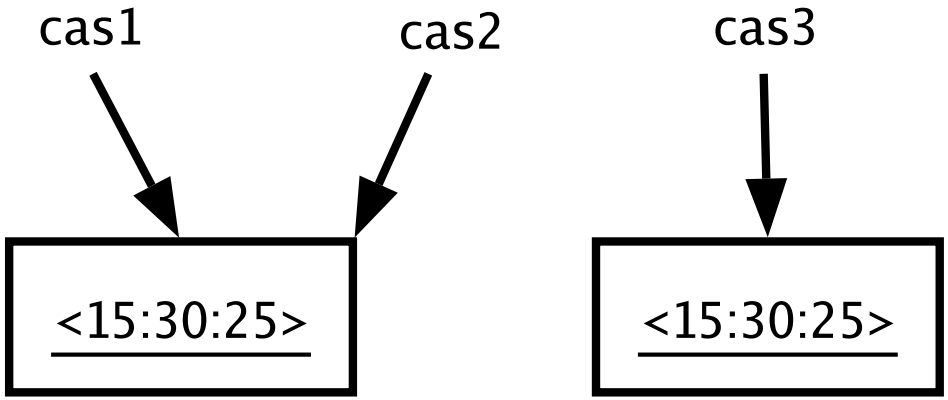
Identita objektu

- každý objekt je jedinečný bez ohledu na stav

Shodnost objektů

- shodnost je vázána na stavy objektů
- objekty, které nejsou identické, mohou být shodné

Základní pojmy – Identita objektu



	Java	Smalltalk	výsledek testu
shodnost	$cas1.equals(cas2)$	$cas1 = cas2$	true
	$cas2.equals(cas3)$	$cas2 = cas3$	true
identita	$cas1 == cas2$	$cas1 == cas2$	true
	$cas2 == cas3$	$cas2 == cas3$	false

Základní pojmy – Rozhraní objektu

Operace vs. metoda

- množina operací reprezentuje chování objektu
- metoda implementuje operaci
- jaký je rozdíl mezi operací a metodou?

Rozhraní objektu

- množina operací, které objekt nabízí
- pouze definuje *co* objekt umí (nabízí), nedefinuje *jak*
- způsob provedení operace závisí na implementaci metody
  - stejné rozhraní může být implementováno různými objekty
  - stejné operace mohou mít různé implementace

Základní pojmy – Komunikace objektů

Komunikace objektů

- objekty spolu komunikují zasíláním zpráv
- příjemce chápe zprávu jako požadavek na provedení služby (operace)
- zpráva obsahuje
  - identifikátor příjemce
  - název operace
  - argumenty
- obsluha zprávy (vykonání metody) reaguje podle stavu / modifikuje stav objektu
- po ukončení obsluhy může metoda vracet výsledek

Ukázka zaslání zprávy

Java: `obj.pridej(100);`

Smalltalk: `obj pridej: 100.`

Komunikace objektů – objekty ve zprávách

Čistá objektově orientovaná prostředí (např. Smalltalk) mají pouze objekty, které hrají jednu z těchto rolí:

- je odesílatel zprávy
- je cíl zprávy
- je odkazován proměnnou v jiném objektu
- je odkazován argumentem zprávy

V hybridních prostředích existují kromě objektů i (jiné) datové typy

- $Java \Rightarrow$ primitivní datové typy
- $C++ \Rightarrow$ strukturované datové typy

Základní pojmy – Abstrakce

- vytvářený systém objektů je abstrakcí řešeného problému
- analýza problému $\Rightarrow$ klasifikace do abstraktních struktur $\Rightarrow$ objekty
- klasifikace je založena na rozpoznávání podobností v řešené problematice
- zjednodušený pohled na systém bez ztráty jeho významu
- objekt je abstrakcí části řešené domény, má definovanou zodpovědnost za řešení části problému

Základní pojmy – Zapouzdření

- Seskupení souvisejících idejí do jedné jednotky, na kterou se lze následně odkazovat jediným názvem (objekt).
- Objektově orientované zapouzdření je seskupení operací a atributů (reprezentujících stav) do jednoho typu objektu. Stav je pak dostupný či modifikovatelný pouze prostřednictvím rozhraní (operace, metody).
- Omezení externí viditelnosti informací nebo implementačních detailů.
- Zaručené rozhraní.

Základní pojmy – Polymorfismus

Polymorfismus

- mnohotvarost, schopnost výskuty v mnoha formách
- logický vztah podobných operací (aplikace operací na podobné, ale technicky různé situace)
- výskyt různých typů chování na základě stejné zprávy
  - možnost vícenásobné definice operace s jedním názvem, která tak může nabývat více implementací (implementuje různé chování).
- mj. umožňuje proměnné objektu odkazovat objekty různých typů v různých okamžicích.

Základní pojmy – Polymorfismus

Časná vazba

- implementace operace (metoda) je vybrána v době kompilace

Pozdní vazba (dynamická vazba)

- je technika dosažení polymorfismu
- implementace operace (metoda) se vybere za běhu podle skutečně dosazeného objektu

Základní pojmy – Dědičnost

Dědičnost

- vyjadřuje hierarchický vztah mezi objekty
- definuje a vytváří objekty na základě již existujících objektů
  - možnost sdílení chování bez nutnosti reimplementace
  - možnost rozšíření chování
- $\Rightarrow$ organizuje a usnadňuje polymorfismus a zapouzdření objektů

Způsob vyjádření dědičnosti závisí na typu jazyku

- třídně orientované jazyky
- prototypově orientované jazyky

Hierarchie

- klasifikace pořadí abstrakcí
- dědičnost x skládání

Co je objekt?

Objekt je abstraktní struktura reprezentující část řešené domény mající:

- chování
- stav
- atributy
- identitu

Objekty

- nabízejí rozhraní
- komunikují zasíláním zpráv podle prezentovaného rozhraní

Třídně orientované jazyky

Představují takový styl OO přístupu, který definuje *třídy* objektů

- nalezené objekty jsou klasifikovány do tříd
- třída je generická definice pro množinu podobných objektů (šablona)
- třída je množina objektů, které mají stejné chování a stejnou množinu atributů

Třída

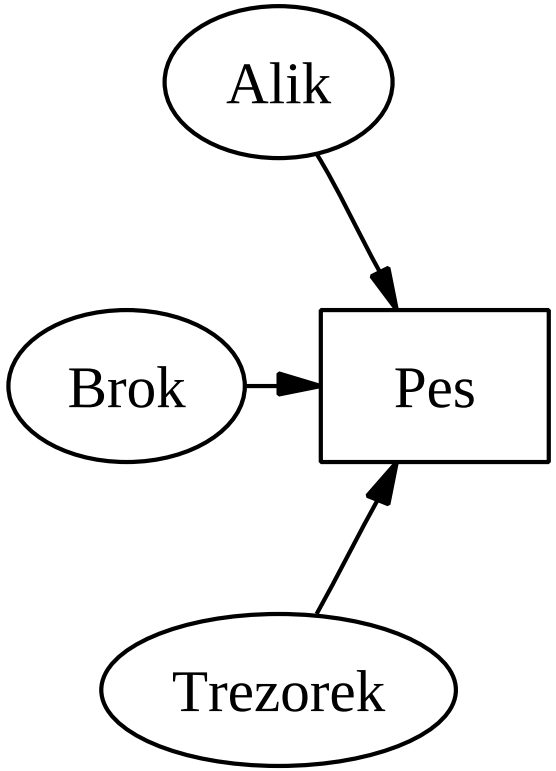
- třída definuje atributy a chování objektu (metody)
- objekt je instance třídy
- objekty stejné třídy sdílejí chování (metody), atributy má každý objekt vlastní
- třída může definovat *třídní atributy* $\Rightarrow$ jsou sdíleny všemi instancemi třídy

Jazyky

- Smalltalk, Java, C++, C#, . . .

Třídně orientované jazyky

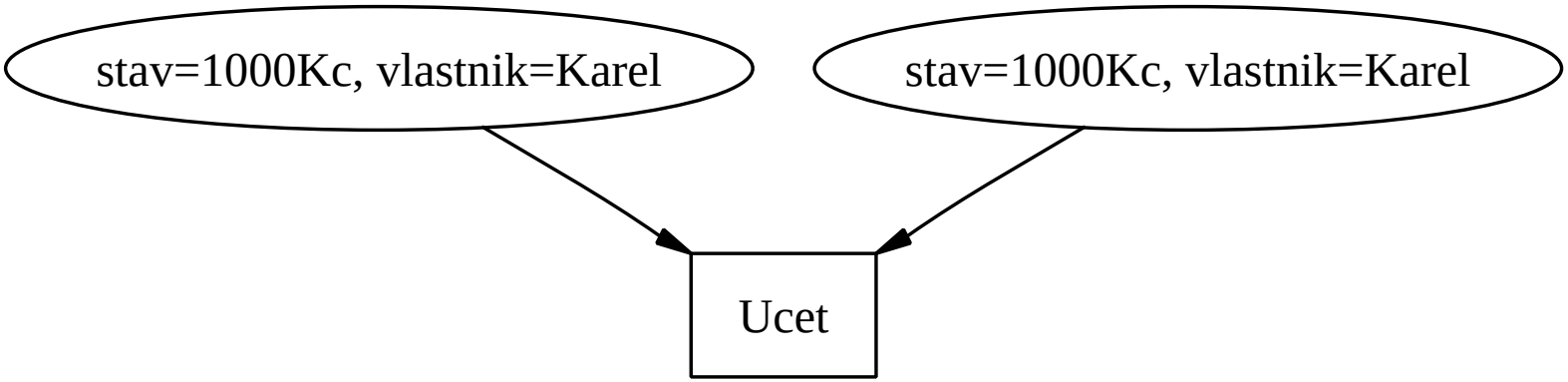
Ukázka třídy a instancí třídy



Třídně orientované jazyky – identita objektu

Identita objektu je nezávislá na stavu a třídě objektu

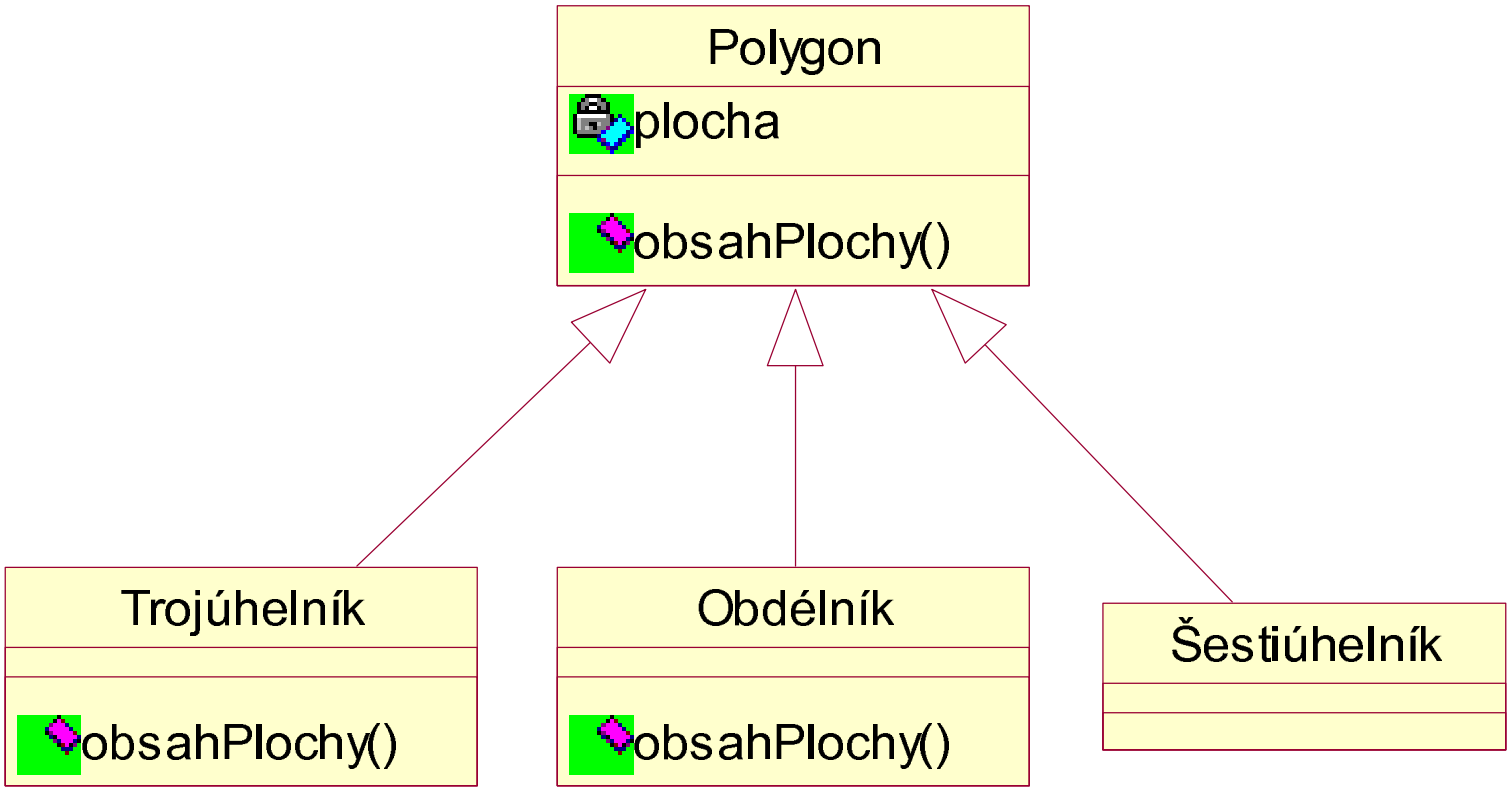
- objekty téže třídy jsou různé
- identita je vlastnost, podle které lze každý objekt identifikovat bez ohledu na jeho třídu nebo aktuální stav.
- většina OO jazyků vytváří jedinečné OID (např. adresa objektu)



Třídně orientované jazyky – dědičnost

Dědičnost

- vyjádřena prostřednictvím dědičnosti tříd
- vztah generalizace/specializace



Třídně orientované jazyky – dědičnost

- přepisování (overriding) je změna definice metody zadané v třídě T v některé z podřízených tříd
- přetěžování (overloading) je technika vícenásobné definice operace v jedné třídě.

Přetěžování metod (Java):

```
prevedNa(Ucet u, int castka);
prevedNa(Ucet u);
```

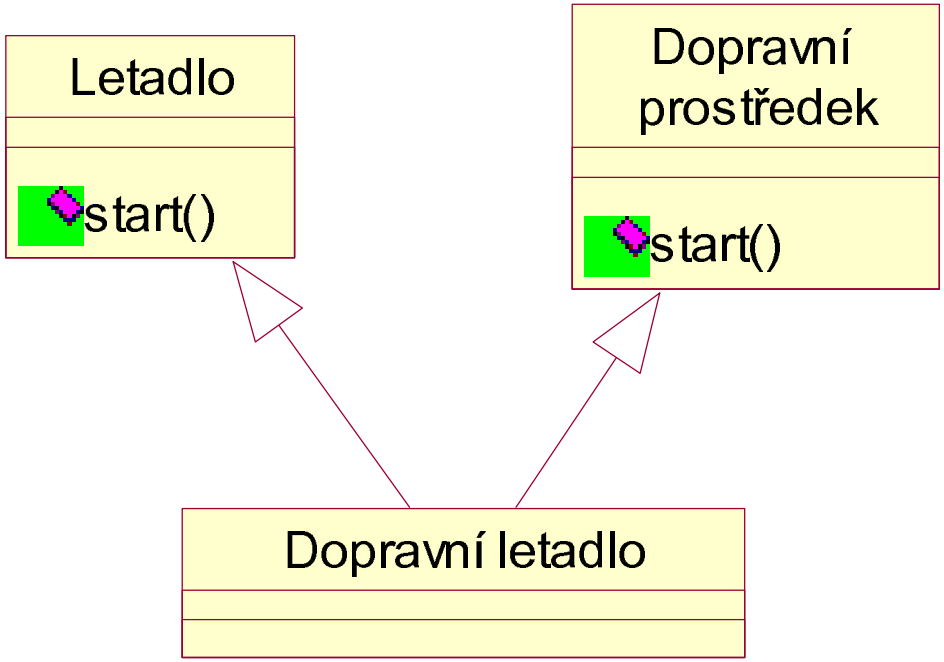
Smalltalk nezná přetěžování:

```
preved: castka na: u.
prevedNa: u.
```

Třídně orientované jazyky – dědičnost

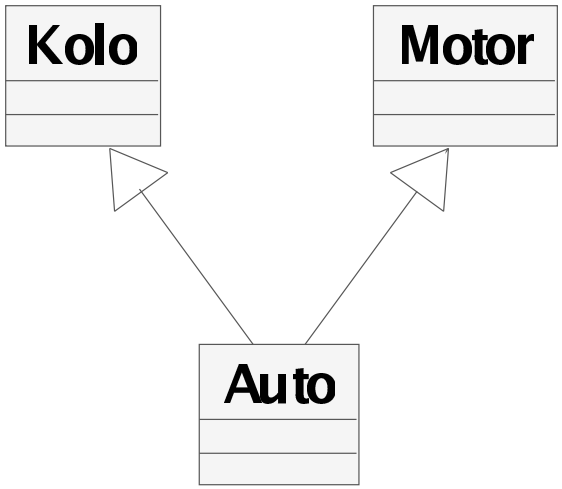
Vícenásobná dědičnost

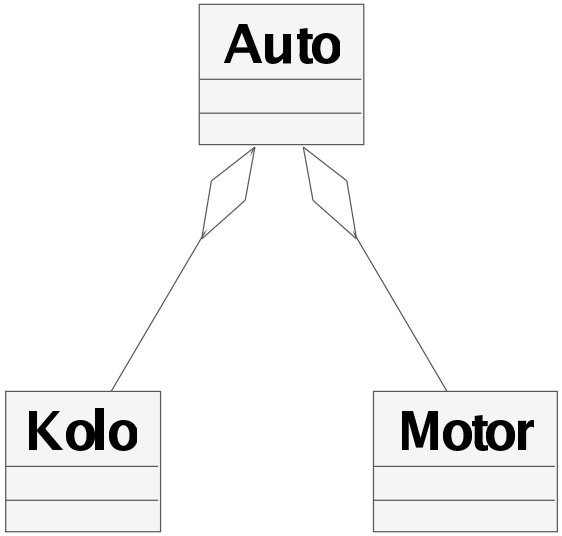
- komplikuje návrh (čitelnost)
- problém nejednoznačnosti
- dá se obejít (skládání objektů)
- existují případy, kdy má vícenásobná dědičnost význam



Třídně orientované jazyky – dědičnost

Vícenásobná dědičnost – ukázka nesprávně použité vícenásobné dědičnosti a její řešení pomocí skládání





Vztah třídy a rozhraní

Objektové rozhraní

- definuje *typ* v objektově orientovaném prostředí
- objekt je typu *A*, pokud implementuje rozhraní *A*
- objekt může mít více typů
- rozhraní může dědit jiná rozhraní
- objekty různých tříd mohou být stejného typu

Třída

- implementuje objekt (resp. chování objektu)
- objekt je instancí své třídy
- třída může dědit jiné třídy
- abstraktní třída
  - odkládá implementaci metod na své podtřídy
  - definuje společné rozhraní pro své podtřídy

Vztah třídy a rozhraní

Dědičnost

- tříd $\Rightarrow$ implementace objektu pomocí implementace jiného objektu (sdílení)
- rozhraní $\Rightarrow$ popisuje typovou zaměnitelnost různých objektů

Jazyky

- C++
  - třída $=$ specifikace typu objektu a implementace objektu
  - dědičnost rozhraní $\Rightarrow$ dědičnost od čistě abstraktní třídy
- Java
  - speciální definice rozhraní
- Smalltalk
  - podtřída $=$ podtyp

Prototypově orientované jazyky

Představují takový styl OO přístupu, který pracuje *pouze* s objekty

- nové objekty se vytvářejí klonováním již existujících objektů
- vždy existuje alespoň jeden počáteční objekt (prototyp)
- třída je množina objektů, které mají stejné chování a stejnou množinu atributů

Dědičnost (delegování)

- dědičnost objektů je vyjádřena delegováním
- objekt může určit množinu jiných objektů, na které deleguje zprávy, kterým sám nerozumí $\Rightarrow$ sdílení chování s jinými objekty
- více "nadřazených" objektů $\Rightarrow$ problém nejednoznačnosti $\Rightarrow$ priorita "nadřazených" objektů

Jazyky

- Self, JavaScript, . . .

Typy, kontrola typů

Význam typování

- určit sémantický význam elementům (hodnoty v paměti)
- pokud má paměťová hodnota přiřazený typ, můžeme s ní pracovat na vyšší úrovni – víme jaké operace je možné provést, můžeme provádět kontrolu typové konzistence atp.

Staticky typované jazyky

- k typové kontrole dochází v době kompilace
- jazyky C++, Java, . . .

Dynamicky typované jazyky

- k typové kontrole dochází v době běhu programu
- jazyky Smalltalk, Self, Python, Lisp . . .

Typy, kontrola typů

I. Ukázka chování staticky a dynamicky typovaných systémů

```
var x;          // (1)
x := 5;         // (2)
x := "hi";      // (3)
```

- staticky typované: řádek č. 3 je ilegální
- dynamicky typované: řádek č. 3 je OK (není požadovaná typová konzistence pro proměnnou x)

II. Ukázka chování staticky a dynamicky typovaných systémů

```
var x;          // (1)
x := 5;         // (2)
5 / "hi";       // (3)
```

- staticky typované: řádek č. 3 je ilegální
- dynamicky typované: řádek č. 3 vyvolá chybu za běhu programu

Typy, kontrola typů

Dynamická kontrola

- probíhá u všech jazyků
- jako dynamicky typované se označují ty, které nemají statickou kontrolu
- některé staticky typované jazyky ($C++$, $Java$) umožňují dynamické přetypování, čímž částečně obcházejí statickou typovou kontrolu

Silně a slabě typované jazyky

- lze se setkat s tímto rozdělením
- avšak význam těchto pojmů není jednoznačný
- viz např. `http://en.wikipedia.org`

Další vlastnosti OOP

- Souběžnost
  - objekty mohou konat ve stejném čase
  - procesy, vlákna

- Perzistence
  - Uložení stavu / dat během evoluce
  - Serializace