1. První přednáška (01-IPP\_intro)
   * Programming process
   * Co je programovací jazyk (What a Programming Language Is)
   * Co je program (What a Computer Program Is)
   * Stručná historie – symbolická jména, subrutiny, zvyšování abstrakce
   * Budoucnost
   * Specifikace programovacích jazyků (Programming languages specification)
     1. Syntaxe
     2. Sémantika – statická, dynamická
   * Formální prostředky definice sémantiky (Formal Semantics Definition)
   * Deklarace vs. definice
   * Vazby (Binding)
   * Druhy vazeb (A Kind of Binding)
     1. Statická
     2. Dynamická
   * Vazby u proměnných (Example – variable)
   * Proměnné
     1. Vlastnosti
     2. Názvy
     3. Umístění, adresa a hodnoty
     4. Typy
     5. Rozsah platnosti
   * Další termíny
   * *K zapamatování*:
     1. Syntaxe, sémantika
     2. Vazby
     3. Abstrakce
     4. Deklarace vs. definice
2. Druhá prezentace: Principy OOP
   * Motivace – modelování, abstrakce
   * Historický vývoj abstrakcí
   * Objektová orientace
   * Modely OOP
     1. Statické
     2. Dynamické
   * Model výpočtu (minimální)
   * Formální báze OOP
   * Popis syntaxe a sémantiky složitějších OOJ
   * OO Paradigma – Koncepty (obecný popis – beztřídní i třídní OOJ)
   * Objekt
     1. Datová povaha objektu
   * Zapouzdření
   * Mnohotvárnost (polymorfismus)
   * Dědičnost (obecně)
   * Vytváření nových objektů
   * Třída (Class)
     1. Instanciace, instance
   * Čistý třídní model OOJ Smalltalk
   * **Klasifikace OOJ**
   * Klasifikace nejen OOJ
   * Historie OOJ

*--- konec druhé přednášky ---*

* + **OO návrh (Design)**
  + Principy OO návrhu
  + **UML**
    1. Diagram tříd/objektů
    2. Diagram tříd: Dědičnost
    3. Násobnost vztahů (Association)
    4. Druhy vztahů mezi entitami
    5. Diagram tříd: Agregace/Kompozice
    6. Sekvenční diagram
    7. Formální návrh v UML
  + Výhody/Nevýhody OOP
  + *K zapamatování*:
    1. OO paradigma – koncepty, pojmy
    2. Formální modely OOP – UML
    3. Klasifikace OOJ

1. Druhá prezentace: **Třídní** objektově orientované jazyky a jejich implementace
   * Třída
   * Třídní dědičnost
   * Hierarchie třídní dědičnosti
     1. Relace dědičnosti *is-a*
   * Definice/Deklarace
   * Operace nad instancemi
   * Operace nad třídami
   * *Případová studie: Python*
   * Invokace metody – kontext
   * Typy v OOJ: 2 přístupy
     1. Čistý
     2. Hybridní
   * **Statický OO model**
   * Implementace třídního OOJ
   * Implementační problémy
   * **Uložení instance v paměti**
   * Statické (třídní) a instanční položky
   * Přístup k položkám instance
   * Zodpovědnost za správu zásobníku volní
   * Optimalizace uložení instance
   * Vytváření instancí (instanciace)
   * Rušení instancí/objektů
   * Řízení toku programu (flow control)
     1. Polymorfismus, pozdní vazba (late binding)
   * Třída jako typ
     1. Zahrnutí typu (subsumption)

*--- konec třetí přednášky ---*

* + Vývojová prostředí
  + *Případová studie: Smalltalk*
  + Běhová prostředí OOJ
  + Reflektivita (*Reflection*)
    1. Reflektivita – příklad ve Smalltalku
  + Činnost překladače
  + Lexikální a syntaktická analýza
  + Sémantická analýza
  + Činnost interpretu
  + Doporučení na závěr
  + *K zapamatování:*
    1. Třída, třídní dědičnost a další pojmy
    2. Způsoby vytváření a rušení objektů
    3. Reprezentace objektů v paměti
    4. Tabulka virtuálních metod
    5. Brzká a pozdní vazba

1. Druhá prezentace: Typový systém objektově orientovaných jazyků
   * Typový systém
     1. Kategorie dle určení typu v kódu
     2. Kategorie dle doby typové kontroly – staticky/dynamicky typované OOJ
   * Typy vs. třídy
   * Přístupy pro určení typu
   * Typ daný jménem
     1. Podtřída vs. podtyp
     2. Vyžádaná dědičnost
   * Typ daný výčtem položek
     1. Skutečné (pod)typy – duck typing
   * Implementace skutečných podtypů
   * Vícenásobná třídní dědičnost
   * Problémy vícenásobné dědičnosti
     1. Metody
     2. Atributy
     3. Inicializace instancí (konstruktory)
     4. Uložení instancí v paměti
   * Rozhraní (Interfaces)
   * Vícenásobná dědičnost rozhraní
   * Rozhraní a polymorfismus
   * Implementace rozhraní
     1. Návrh
   * Rozhraní objektu vs. protokol
   * Systémy s rolemi
   * Operace s vícetypovými objekty
   * Perzistence objektů
     1. Implementace
   * *K zapamatování:*
     1. Vyžádaná dědičnost, skutečné podtypy
     2. Problémy vícenásobné dědičnosti
     3. Implementace rozhraní
     4. Systémy s rolemi – vícetypové objekty
     5. Perzistence
2. Druhá prezentace: Beztřídní objektově orientované jazyky
   * Charakteristika beztřídních OOJ
   * *Případová studie: SELF*
   * SELF: „třídní“ model
     1. Ukázka
   * *Případová studie: JavaScript*
3. Druhá prezentace – vynecháno:
   * Teoretický OOJ: ς-kalkul, minimální OO model
   * Ostatní nejen OO koncepty
     1. Výjimky
     2. Šablony – generický polymorfismus
4. Třetí prezentace (06-IPP\_nonstruct): Language Classification
   * Co znamená programování (What *Programming* Language)
   * **Úrovně datové abstrakce** (Data Abstraction Level)
   * Machine-oriented languages
   * Rané jazyky vyšší úrovně
   * Jazyk PL/I
   * Strukturované jazyky
   * Jazyky s abstraktními datovými typy
   * OO jazyky
   * Další paradigmata
   * **Abstrakce řízení – imperativní jazyky**
   * Abstrakce na úrovni příkazů
     1. Strojové jazyky
   * Abstrakce na úrovni programového kódu
     1. Podprogramy, bloky, koprogramy, paralelní programy, odložené zpracování
   * Abstrakce řízení – deklarativní jazyky
   * Abstrakce na úrovni programového kódu
   * Order selection – strategie volání podprogramů
     1. Volání hodnotou (call-by-value, strict)
     2. Volání jménem (call-by-name, non-strict)
     3. Volání v případě potřeby (call-by-need, lazy)
5. Třetí prezentace: Nestrukturované jazyky
   * Typičtí představitelé – Fortran, Basic
   * Features we face even today
   * Formální báze nestrukturovaných jazyků
   * Popis syntaxe
   * Popis sémantiky
   * Datové abstrakce
   * Způsoby manipulace s daty (Data/Control Manipulation)
   * Otevřené podprogramy (Open Subroutines)
   * Vkládání zdrojového textu (Insertion)
   * Vlastnosti otevřených podprogramů (Features of Such Subroutines)
   * Návrh programu (Program Design)
   * Nevýhody
   * Typy (Types and Their Processing)
     1. Reprezentace položek v paměti – špatný a lepší návrh
     2. Vyhodnocování během chodu (Evaluation During Run Time)
     3. Pole
   * Řízení toku (Flow Control)
   * Formální verifikace a validace (Formal V&V)
   * Možnosti využití metodik, postupů softwarového inženýrství (Suitability for SE Methodologies)
   * Překlad (Compiler Tasks)
   * Sémantická analýza
   * Kontextová analýza
   * Generování kódu
   * Interpretování (Interpreter Tasks)