7 Jazyk UML (Unified Modeling Language)

7.1 Základní charakteristika jazyka

- **Motivace**
  - **vznik řady OO metod a metodologií (konec 80. let a první polovina 90.let) → podobné notace vyjadřující totéž, komplikující rozdíl**
  - **snaha o standardizaci týmem OMG**
  - **1995:**
    - **Booch + Rumbaugh (Rational Software): Unified Method v.0.8**
    - **Rational Software kupuje Objectory (Jacobson)**
  - **1996: UM → UML (Booch, Rumbaugh, Jacobson)**
  - **1997: UML v.1.1 standardem OMG (základem UML 1.0 od Rational)**
  - **UML v.1.2**
  - **1999: UML v.1.3**
  - **2001: UML v.1.4**
  - **snaha povzbudit vývojáře k modelování systémů před jejich vytvářením a umožnit univerzálnost nástrojů vizuálního modelování (interoperability)**
- **Úloha modelování při vývoji programů**
  - **Proč modelujeme?**

- lepší pochopení vyvíjeného systému

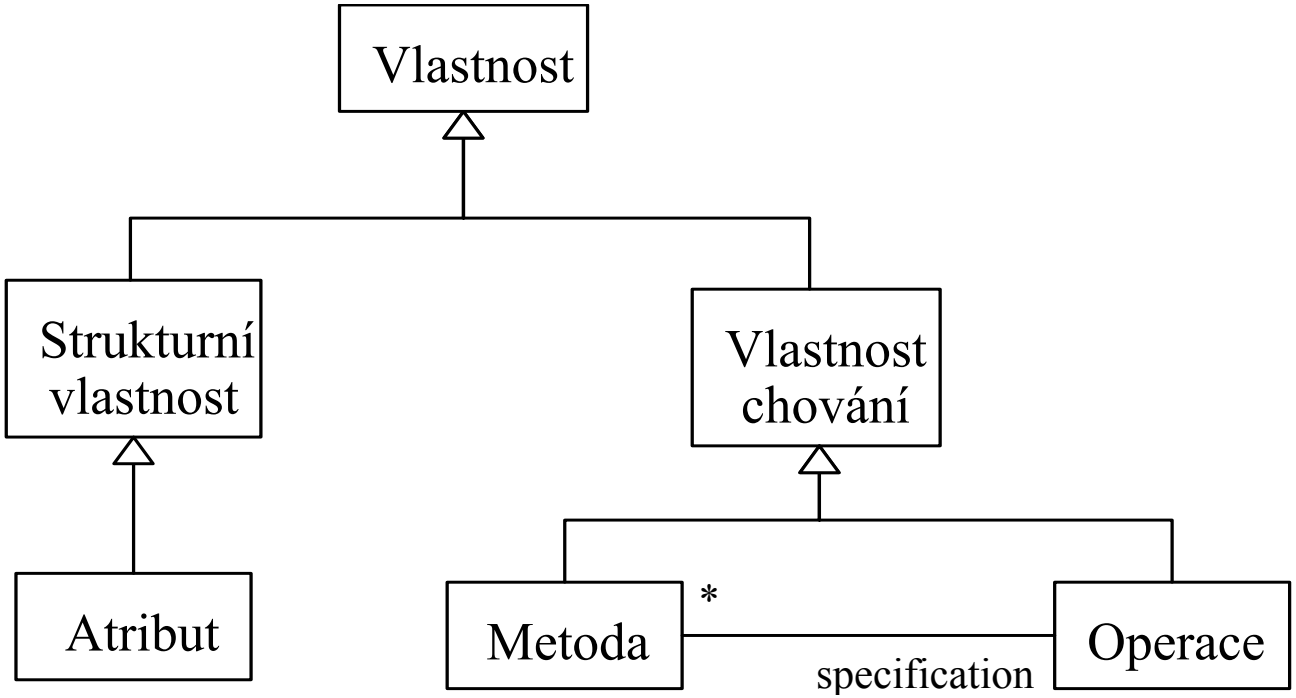
- Cíle modelování?
  - vizualizace (jaký je nebo má být)
  - specifikace struktury a chování systému
  - „šablona“ pro konstrukci systému
  - dokumentuje provedená rozhodnutí
- Zásady modelování:
  - volba modelů má vliv na zvládnutí problému a podobu řešení
  - každý model lze vyjádřit na různé úrovni podrobností
  - nejlepší modely jsou ty, které jsou spojeny s realitou
  - většinou nestačí jediný model, ale je třeba vytvořit několik „nezávislých“ modelů

UML je jednotný jazyk (grafický) pro specifikaci, vizualizaci, konstrukci a dokumentaci při OO analýze a návrhu (OOAaD) a pro modelování organizace (business modelling).

- Složky standardu UML
  - notace (syntax jazyka) pro jednotlivé modely OOAaD
  - metamodel – formální model definující sémantiku notace modelů
  - jazyk OCL (Object Constraint Language) – formální jazyk pro specifikaci omezení

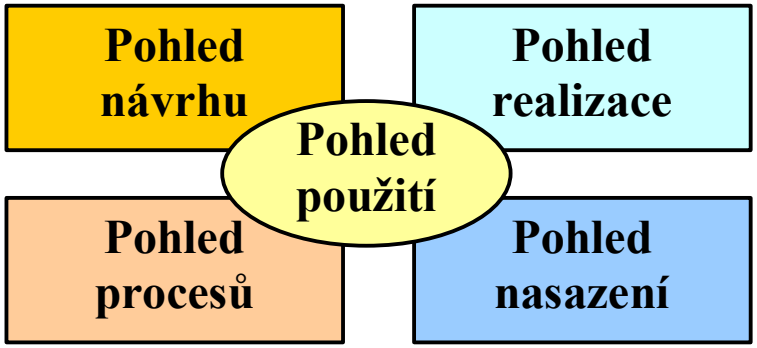
- specifikace rozhraní (v jazyce CORBA IDL a XML DTD) pro výměnu uživatelských modelů mezi nástroji a systémy OOAaD

Vrstva	Příklad
meta-metamodel	*metatřída, metaatribut, metaoperace*
Metamodel	*třída, atribut, operace, komponenta*
Model	*Student, jméno, foto, zobrFoto, srvStudent*
uživatelské objekty	*<Student_100>, "Novák", "....."* *<srvStudent_1234>*



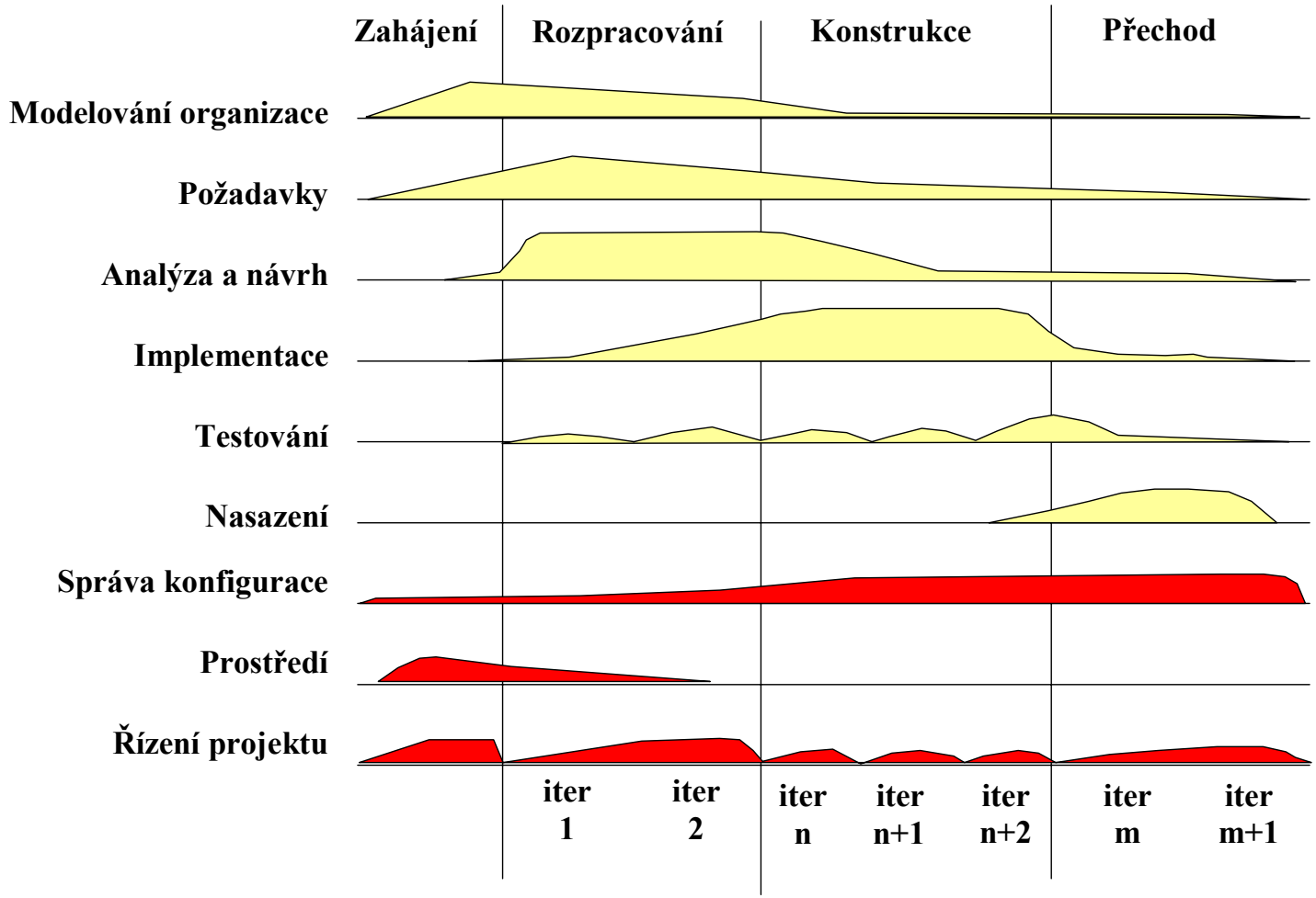
- **Stavební bloky UML**
  1. **prvky – abstrakce:**
     - **strukturní – třída, případ použití, komponenta, …**
     - **chování – interakce, stav**
     - **seskupování – modul, balíček, podsystém (package)**
     - **komentáře - poznámka**
  2. **vztahy:**
     - **závislost**
     - **asociace**
     - **generalizace**
     - **realizace**
  3. **diagramy:**
     - **diagram tříd**
     - **diagram objektů**
     - **diagram případů použití**
     - **diagram interakce – d. sekvence, d. spolupráce**
     - **stavový diagram**
     - **diagram aktivit**
     - **diagram komponent**
     - **diagram nasazení (rozmístění) (deployment)**

- **Modelování architektury systému**



Pohled	Statické aspekty	Dynamické aspekty
použití	diagramy použití	diagramy interakce, stavové diagramy, diagramy aktivit
návrhu	diagramy tříd, diagramy objektů	diagramy interakce, stavové diagramy, diagramy aktivit
procesů	diagramy tříd, diagramy objektů	diagramy interakce, stavové diagramy, diagramy aktivit
realizace	diagramy komponent	diagramy interakce, stavové diagramy, diagramy aktivit
nasazení	diagramy nasazení	diagramy interakce, stavové diagramy, diagramy aktivit

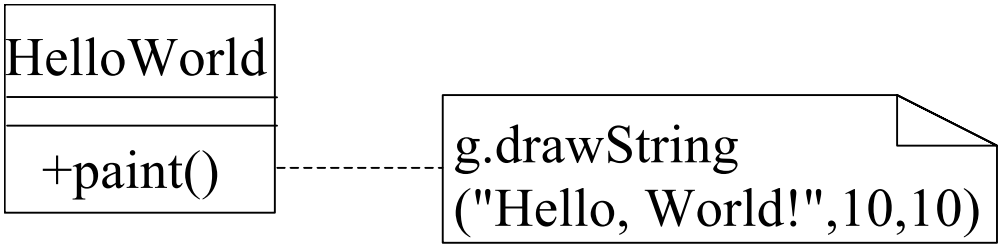
- **Životní cyklus vývoje**
  - **UML je nezávislý na procesu, největší užitek pro proces, který je:**
    - **řízený případy použití (use case driven),**
    - **orientovaný na architekturu systému (architecture-centric)**
    - **iterativní a inkrementální**

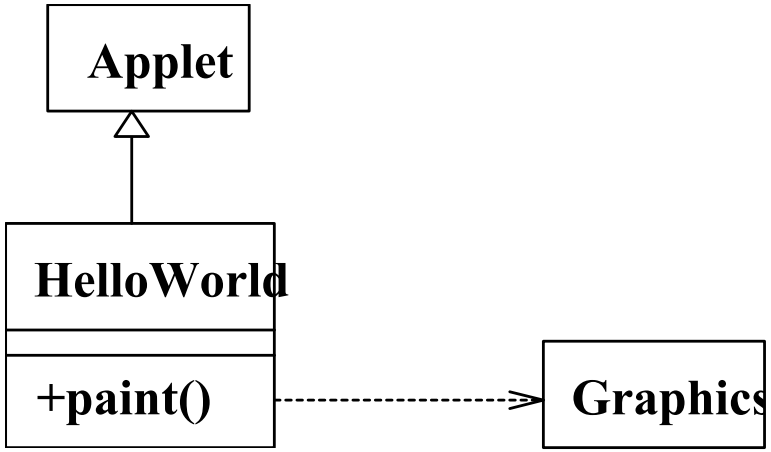


Př)

```
import java.awt.Graphics;
class HelloWord extends java.applet.Applet {
   public void paint (Graphics g) {
      g.drawstring ("Hello, World!", 10, 10);
   }
}
```

Diagramy tříd:





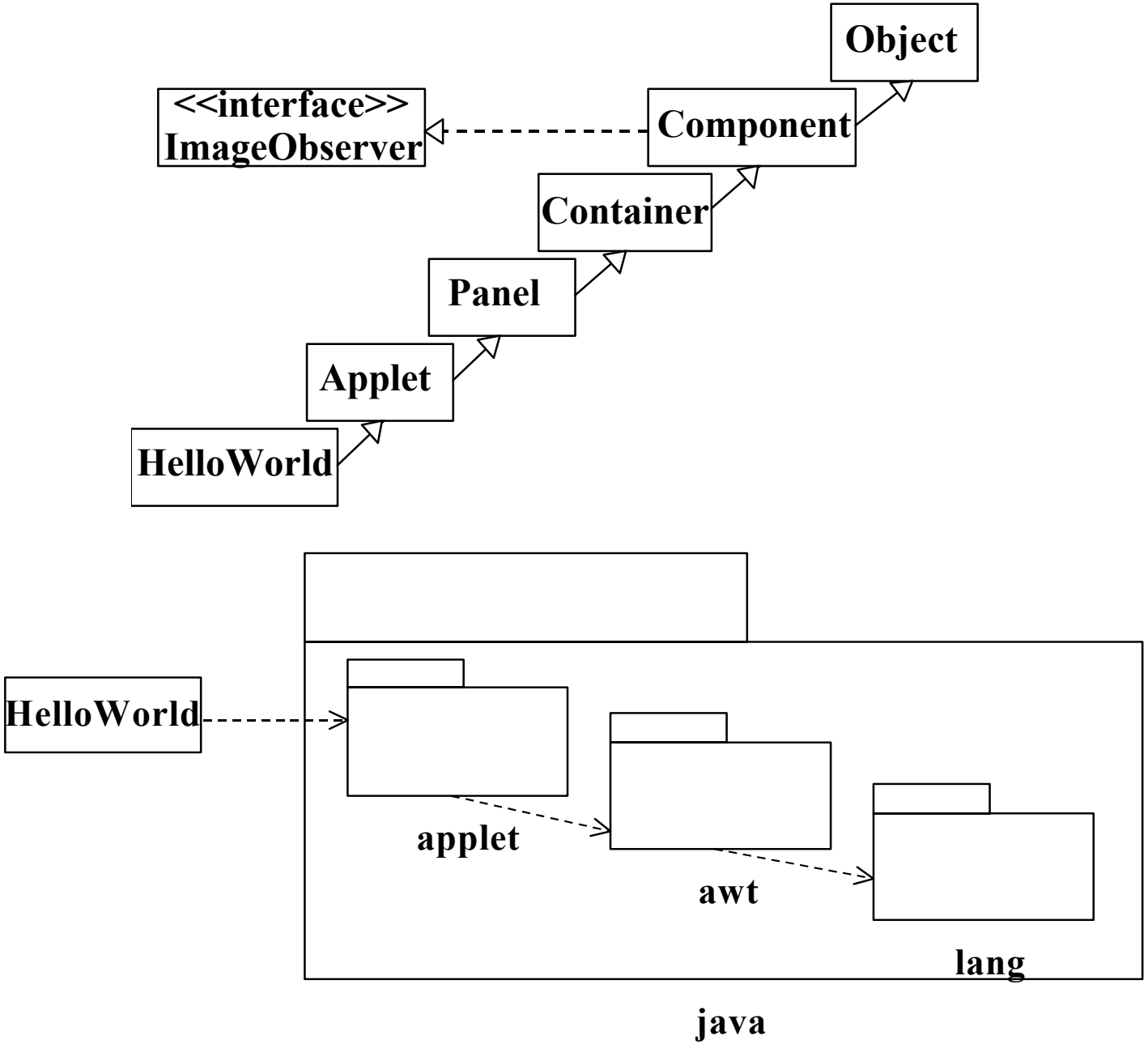

Object
<<interface>>
ImageObserver
Component
Container
Panel
Applet
HelloWorld
HelloWorld
applet
awt
lang
java

Diagram sekvence:

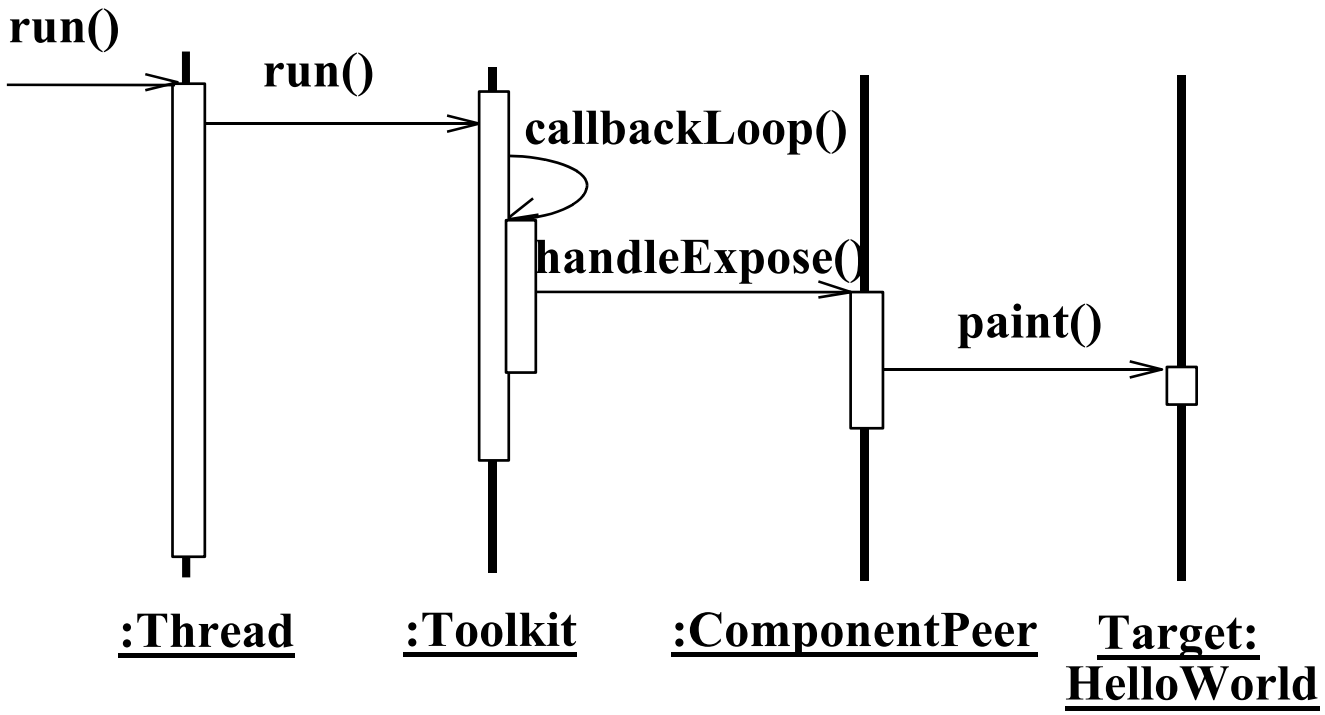


Diagram komponent: