OOP – BI-WSI-SI-08

Čisté objektové paradigma — klíčové pojmy, abstrakce, chování.

Obsah

1 Základní pojmy

- **Třída** – Vzor pro objekt. Definuje vlastnosti a chování objektů vytvářených podle tohoto vzoru.
- **Objekt** – Instance třídy. Uchovává svůj vlastní vnitřní stav. Má vlastní identifikátor, kterým se odlišuje od ostatních objektů stejné třídy (objekt je vždy jedinečný v rámci programu).

  Chování objektu (*behaviour*) je definované jeho třídou. Objekt reaguje na zasílané zprávy (*messages*) tak, že provede příslušnou metodu definovanou pro jeho třídu (**method lookup**).

  Objekt má zodpovědnost: sadu problémů, které řeší. (viz. **distribution of responsibility**, **separation of concerns**).

  - **atributy** – udržují vnitřní stav objektu
  - **metody** – využívají nebo mění vnitřní stav objektu
- **Zpráva (Message)** – *What to do?*

  Zpráva odeslaná cílovému objektu, aby něco provedl. Různí příjemci mohou reagovat na stejnou zprávu různě. Příjemce je známý až při chodu programu (**late binding**).
- **Metoda (Method)** – *How to do it?*

  Sekvence instrukcí pro vyřešení problému. Definice, jak odpovědět na zprávu. Jakou metodu vybrat při dané zprávě se určuje dynamicky za běhu programu (**late binding**) podle procesu **method lookup**.

2 OOP foundations

Základy pro objektově orientované paradigma.

2.1 Abstrakce

Objekt vytváří určitou abstrakci, která schovává interní implementaci a detaily před ostatními objekty. *Ostatní objekty mohou s konkrétním objektem komunikovat pouze zasíláním zpráv a nesmějí vědět, jak konkrétní objekt funguje uvnitř.*

2.2 Encapsulation

information hiding, zapouzdření

Vnitřní stav objektu není přístupný zvenčí. Stav objektu je udržován pomocí jeho atributů. Ty lze měnit a číst pouze pomocí metod. Tím má objekt větší kontrolu nad změnami svého stavu.

2.3 Composition

Třída může být závislá na jednom či více jiných objektech. To má za následek:

- delegování problémů na jiné objekty pro lepší znovupoužitelnost kódu,
- výměna implementace dílčího objektu nijak neovlivní chování objektu, který s ním pracuje.

2.4 Distribution of responsibility

separation of concerns

Rozdělení zodpovědnosti (funkcionality nebo chování) mezi objekty tak, aby se co nejméně překrývaly (aby implementace zabrala co nejméně kódu, tedy znovupoužitelnost kódu byla co nejvyšší).

2.5 Message passing

delegating responsibility

Delegování zodpovědoností na dílčí objekty. Vyhodnocování metody objektu (*receiver*) v kontextu jiného objektu (*sender*).

- **explicit** – Předání *sender* objektu do *receiver* objektu.
- **implicit** – *Receiver* objekt provede *method lookup* podle zaslané zprávy (Pharo).

2.6 Inheritance

Zakládání vlastností jedné třídy (*child class*, *derived class* nebo *subclass*) na jiné nadřazené třídě (*parent class*, *base class* nebo *super class*). Mezi třídami pak vzniká stromová hierarchie. *Sub class* získává všechny vlastnosti svojí *super class* a přidává nové.

Polymorphism Jednotné rozhraní pro různé datové typy objektů. *K objektům různých typů lze přistupovat pomocí jednoho rozhraní.* Dynamický polymorfismus: Jaký konkrétní objekt se použije se vybere až za běhu programu (**late binding**).

3 OOP foundations ideas

Uniform reference Všechno je objekt. Každá entita OO programu je objekt.

Uniform access Všechno se provádí pouze pomocí zasílání zpráv mezi objekty.