# Principy objektově orientovaného návrhu

## Cíle objektově orientovaného návrhu

- **Cílem objektově orientovaného návrhu** je vytvoření kvalitního programového vybavení (**SW** – *software*), který má následující vlastnosti:
  - **Robustnost** – cílem je vytvořit software, který je schopen správně reagovat i na neočekávané vstupy, které nejsou explicitně definovány pro danou aplikaci.
  - **Přizpůsobivost** (*adaptability*) – schopnost software běžet s minimálními úpravami i na jiné platformě (windows, unix, MacOs).
  - **Znovupoužitelnost** (*reusability*) – cílem je vytvořit software nebo jeho část tak, aby byla opětovně použitelná v jiných systémech. Vytváření programů se pak podobá stavebnici, kdy z hotových dílů vytváříme funkční celek.

## Principy objektově orientovaného návrhu

- Mezi nejdůležitější **principy objektově orientovaného návrhu**, které vedou ke splnění uvedených cílů patří:
  - **Abstrakce** – cílem je rozdělit složitý problém na základní části, tyto části popsat a implementovat.
    - Příkladem abstrakce mohou být **abstraktní datové typy** – matematický model datových struktur, který specifikuje v jakém datovém typu budou uložena data a jaké operace lze nad těmito daty provádět. Abstraktní datové typy jsou reprezentovány obvykle **třídami**.
    - **Objekty** jsou pak konkrétními výskyty (instancemi) tříd.
  - **Zapouzdření** – cílem je ukrýt implementační detaily které nejsou pro uživatele podstatné a poskytnout uživateli určité rozhraní pomocí kterého může přistupovat k datovým strukturám. Chrání datové struktury před neoprávněnou manipulací.
    - Implementační detaily jsou obvykle řešeny pomocí **metod** tříd.
  - **Modularita** znamená rozdělení softwarového systému do oddělených funkčních jednotek (modulů). Moduly mohou být navrženy tak aby byly využitelné v jiných systémech (princip znovupoužitelnosti).

- Při návrhu složitého software obvykle skládáme celek z jednodušších komponent. Pro „lepší“ využití jednotlivých komponent využívá objektově orientovaný návrh principů **dědičnosti** a **polymorfismu.**

- **Dědičnost** - umožňuje vytvořit tzv. generické třídy, ze kterých je možné odvozovat třídy další, přidávat do nich nové metody, popř. modifikovat metody existující. Dědičnost se využívá zejména proto abychom se vyhnuli nadbytečnému programovému kódu (popř. nadbytečným datovým strukturám).

- **Polymorfismus** (mnohotvarost) – umožňuje, aby stejné metody vykonávaly různou činnost v závislosti na objektu nad kterým pracují.

- Jednotlivé objekty mění své stavy prováděním metod a komunikují mezi sebou zasíláním **zpráv**.

  - **Provedení programu** v podstatě znamená změnu stavu objektů a jejich vzájemnou komunikaci