9. Objektově orientované paradigma

- kód je přidružen k datům (metody jsou zapouzdřeny v objektech), což umožňuje snadnější přenosnost kódu mezi různými programy, abstrakce, zapouzdření
- snaha modelovat řešení tak, aby principy odpovídaly reálnému světu

1. Základní pojmy - objekt, třída, dědičnost, polymorfismus

- **Objekt**: jednotlivé prvky modelované reality jsou v programu reprezentované entitami, nazývají se objekty; pamatují si svůj stav a navenek poskytují určité operace - metody, pamatuje si svůj stav (v podobě dat či atributů), zveřejněním svých operací vytváří rozhranní → zapouzdření
- **Třída**: jednotlivé instance (konkrétní případ realizace) objektů seskupujeme do tříd, je to celistvá entita
- **Rozhranní**: k dané třídě vytvoříme vytvoříme rozhranní s požadavky, které by daná třída měla implementovta, rozšiřujeme schopnosti, příklad třída Sekretářka bude mít rozhranní SchopenPsátNaPočítači
- **Zapouzdření**: zajišťuje konzistenci přístupů k jednotlivým objektům, každý objekt má přesně definováno k jakým objektům má přístup
- **Dědičnost**: objekty jsou organizovány v stromové struktuře, jednotlivé objekty, které jsou závislé na ostatních dědí určité vlastnosti, můžou k nim přidat svoje, myšlenka se dále implementuje rozdělením objektů do tříd, každý objekt je instancí jedné třídy a třída může samotná dědit z jiné, příklad zahradník, prodavačka jsou třídy, které dědí třídu člověk, ta má svoje rozhranní a metody, ty bude třída prodavačka umět
- **Polymorfismus**: objekt se chová podle toho jaké třídy je instancí, několik objektů poskytuje stejné rozhranní (funkce, protokoly, které může programátor využívat), vyskytuje se u dědičnosti, objektům odvozených z různých tříd umožňuje volat tutéž metodu se stejným významem, často přes určité rozhranní, jinými slovy polymorfismus je vlastnost, která umožňuje volání jedné metody se stejným jménem, ale jinou implementací, příklad stejná zpráva poslaná od různých objektů bude implementovaná stejně
- **Metoda**: překrývání (override) původní zděděné metody, přetěžování metody (overloading) rozšiřovat funkce děděné metody - je potom na výběr, kterou konkrétní implementaci zvolíme, naopak třeba funkce *jezdit(vozidlo)* není přetížená (stejný počet vstupů, tak výstupů)
- **Abstraktní třída**: jedná se o entitu, která neumožňuje vytvářet objekty, slouží jako předpis metod či atributů děděné třídě, metody abstraktní třídy mohou být implementovatelné (děděná třída může dané metody implementovat) či nikoliv, takže se jedná v podstatě o šablonu pro další třídy
- **Atomizaace**: dobře vytvořené třídy mají jednotlivé metody atomizované (každá metoda dělá jednu věc pořadně), rozebrané, takže lze poté jednoduše spravovat, rozšiřovat

2. Použití

- umožňuje velice efektivně řešit problémy z reálného života za pomocí implementace objektů

- popisuje způsob výroby v závislosti na uvažování nad daným objektem

3. SOLID

- princip, který by měl splňovat dobře napsaný program, jedná se o akronym
- **S**: SRP, single-responsibility principe, každá třída by měla dělat jednu věc, za kterou je zodpovědná
- **O**: open-close principe, program by měl být otevřený pro rozšíření, avšak zapouzdřený k modifikaci
- **L**: Liskov substitution principe, neboli že děděná třída by se měla chovat v rámci rodičovské třídy
- **I**: interface segregation principe, lépe je, aby bylo vícero rozhranní, než-li jedno pokrývající všechny třídy, s tím souvisí SRP
- **D**: dependency inversion principe: třídy by měly být závislé na abstrakcích, high-level třídy by neměli být závislé na low-level a abstrakce tříd by neměli záviset na detailech, detaily by měli záviset na konkrétní abstrakci