**Prednosti objektno usmerjenega programiranja**

Strukturirano in objektno usmerjeno programiranje sta dva različna načina za dosego istega cilja. Nobeden izmed obeh načinov ni "boljši" od drugega, to je v veliki meri odvisno od osebnega okusa programerja oziroma razvijalca programske kode. Prav tako lahko v programu svobodno mešate strukturiran ali objektno usmerjen slog programiranja, vendar to ni priporočljivo.

Obstaja nekaj prednosti, ki jih programerjem oziroma piscem kode ponuja uporaba objektno usmerjenega programiranja:

* **Enostavna preslikava situacij iz resničnega življenja:** Objekte je enostavno preslikati v "objekte" v resničnem življenju, kot so na primer osebe, različni predmeti, izdelki, nakupovalni vozički v spletnih trgovinah, objave v spletnem dnevniku itd. To precej olajša pisanje programov. , Vnaprej je precej jasna vloga vsakega objekta, podobno pa so jasni tudi odnosi oziroma povezave med posameznimi objekti.
* **Enostavno pisanje modularne kode:** Objektno usmerjen slog programiranja je zelo primeren za pisanje kode v ločenih modulih. Z ločevanjem kode v ločene module je koda bolj obvladljiva, saj so moduli namenski – rešujejo točno določene probleme oziroma naloge nad določeno vrsto podatkov. Posledičnoje tako kodo lažje razhroščevati in razširiti njene zmožnosti.

* **Enostavno pisanje kode, ki jo lahko uporabimo večkrat:** Pisanje kodo za večkratno uporabo lahko pri pisanju aplikacij prihrani veliko časa. Sčasoma lahko zgradimo celo knjižnico modulov kode za večkratno uporabo. Te module potem lahko uporabimo v različnih aplikacijah. Objektno usmerjeno programiranje omogoča relativno enostavno pisanje kode za večkratno uporabo, saj so podatkovne strukture in funkcije združene v enem objektu, primernem za večkratno uporabo. Prav tako je mogoče, da vzamete obstoječi objekt in ga razširite za uporabo v novi aplikaciji tako, da objektu dodate nove funkcionalnosti. Tudi tako lahko objekte na enostaven način uporabite večkrat.
* **Določanje novih podatkovnih tipov:** Na posamezne razrede lahko  gledamo kot na nove podatkovne tipe. Razrede lahko sestavimo tako, da so podatki, ki opisujejo določen objekt vedno smiselni, oziroma v določenem stanju. Na primer razred Ulomek lahko sestavimo tako, da bo imenovalec vedno pozitiven. Na ta način lahko poenostavimo delo s podatki, ki nam predstavljajo ulomke, saj ni treba paziti na to, da je imenovalec morda 0. Prav tako lahko ločimo način, kako podatke hranimo in kako so predstavljeni – npr. v objektu tipa Datum lahko datum hranimo v obliki števila (datum je predstavljen npr. kot število dni po 1. 1. 1800), ob prirejanju in izpisovanju pa lahko uporabimo "običajni" zapis kot "21/3/2018")