Popis programu

Program implementuje třídu Faktura a další pomocné třídy, které spolu tvoří kompletní reprezentaci faktury. Využil jsem princip kompozice, kdy třída Faktura obsahuje reference na další objekty (Prijemce, CastkyDokladu,

Sazba). Toto je správný objektově orientovaný přístup - každá třída má svou jasně definovanou zodpovědnost.

# Hlavní vlastnosti implementace

Nemutovatelnost objektů - Všechny třídy jsou implementovány jako immutable (neměnitelné). Proto mají všechny atributy modifikátor final a nejsou k dispozici žádné settery, což zajišťuje, že údaje na faktuře nelze měnit po vytvoření, jak bylo požadováno v zadání.

Zapouzdření - Všechny atributy jsou privátní a přístup k nim je umožněn pouze přes gettery.

Kompozice - Třída Faktura obsahuje objekty jiných tříd (Prijemce, CastkyDokladu), což je příklad kompozice objektů.

Automatický výpočet - Třída CastkyDokladu automaticky vypočítá cenu s DPH na základě ceny bez DPH a sazby.

Formátování - Metoda toString() je přepsána ve všech třídách pro pěkný a přehledný výpis.

# Testovací případy

V testovací třídě FakturaTest jsou implementovány tři základní testy:

Vytvoření standardní faktury - Testuje vytvoření a výpis faktury se základní sazbou DPH.

Faktura s nulovou hodnotou - Testuje, zda funguje korektně faktura s nulovou částkou.

Faktura se sníženou sazbou DPH - Testuje použití jiné sazby DPH.

Testy vypíšou výsledné faktury na standardní výstup, takže můžete zkontrolovat jejich správnost

pohledem.

Část 2

Vysvětlení úprav v implementaci

Původní implementace již využívala kompozici objektů (třída Faktura obsahovala objekty tříd Prijemce a CastkyDokladu),

ale nyní jsem přidal následující funkce k třídě Faktura:

Zprostředkování dílčích hodnot - Do třídy Faktura jsem přidal metody, které delegují volání na metody kompozitních objektů. Například metoda getJmenoPrijemce() volá metodu getJmeno() na objektu prijemce.

Tímto způsobem třída Faktura zprostředkovává přístup k dílčím hodnotám svých komponent.

Aktualizace testů - V testovací třídě jsem přidal nový test testDilciHodnoty(), který demonstruje přímý přístup k dílčím hodnotám přes delegované metody třídy Faktura.

UML diagram - Vytvořil jsem UML diagram, který znázorňuje vztahy mezi třídami v našem řešení. Kompozice je

znázorněna plnou čárou s diamantem na straně celku.

# Změny v implementaci testů

S přidáním delegovaných metod do třídy Faktura se mění i způsob, jakým můžeme testy implementovat. Nyní můžeme testovat dílčí hodnoty přímo přes rozhraní třídy Faktura, aniž bychom museli pracovat s jejími vnitřními

objekty.

# Hlavní rozdíly

Původní přístup (bez delegace):

javaFaktura faktura = new Faktura(...); String jmeno = faktura.getPrijemce().getJmeno(); double cenaBezDph = faktura.getCastkyDokladu().getCenaBezDph();

Nový přístup (s delegací):

javaFaktura faktura = new Faktura(...); String jmeno = faktura.getJmenoPrijemce(); double cenaBezDph = faktura.getCenaBezDph();

Tento nový přístup má několik výhod:

Jednodušší API - Klient může přistupovat k dílčím hodnotám přímo přes rozhraní třídy Faktura, což je jednodušší a přímější.

Abstrakce implementace - Klient nemusí znát vnitřní strukturu objektu Faktura.

Flexibilita změn - Pokud se v budoucnu změní implementace vnitřních objektů, klientský kód nemusí být měněn.