Sovelto

Viikko 2, Oliot ja periytyminen

Tuote- ja varastohallinta

ACADEMY
A PART OF ACADEMIC WORK

Olio- ja periytymisharjoitukset sekä vielä kokoelmista

- Harjoituksessa tehdään ensin luokat Tuote ja Varasto. Kun nämä toimii ja on testattu niin sitten tehdään vielä lisäksi Tuote-luokasta johdetut luokat Alennustuote ja Tilaustuote.
- Harjoitukset tavoitteet:
  - luokan, ominaisuuksien ja konstruktorien tekeminen
  - ToString() -metodin toteutus omaan luokkaan
  - kokoelmien käyttö (lisäys, poisto ja käsittely silmukoissa)
  - periytyminen ja johdettujen luokkien (aliluokkien) konstruktorit
  - virtuaalimetodien toiminta
  - lisätehtävässä Enum-määrittelyn tekeminen ja käyttäminen
  - hieman tyyppimuunnoksia ja kertausta moneen perusasiaan eli koodausrutiinin hankkiminen jatkuu

Testiprojektin tekeminen

- Luo uusi projekti jonka tyyppi on Console Application, pidetään käyttöliittymä siis mahdollisimman yksinkertaisena
- Solutionin nimi olkoot OlioharjoitusViikko2 ja projektin nimeksi Testeri
- Talleta Solution sopivaan paikkaa josta löydät sen jatkossa
- Tähän projektiin lisätään seuraavissa harjoituksen vaiheissa luokat
  - Tuote
  - Varasto
  - Alennustuote
  - Tilaustuote
  - Verokanta (ei luokka vaan Enum-määrittely)

Tuote-luokka

- Toteuta esimerkin mukainen luokka
- Ihan kaikki koodi ei ole valmiina, lisättävää koodia ei kuitenkaan ole kovin paljon
- Testaa ensin toiminta ja sitten vasta lisäätarkistukset ominaisuuksiin :
  - Nimi, pituus vähintään 5 merkki
  - Hinta > 0
  - heitä poikkeus jos virhe, esimerkiksi: throw new ArgumentException("Nimi liian lyhyt");
- Huomaa että joudut tekemään erikseen kentät jotta pystyt tekemään ominaisuuksiin tarkistukset!

```
class Tuote
{
    public int Tuotenumero { get; set; }
    public string Nimi { get; set; }
    public decimal Hinta { get; set; }
    public Tuote() { }
    public Tuote(int tnro, string nimi, decimal hinta)
    // lisää koodi

    public virtual decimal LaskeHinta()
        // lisää koodi

    public override string ToString()
        // palaute tuotenumero, nimi ja hinta
}
```

Varasto-luokka

```
class Varasto
{
    public List<Tuote> Tuotteet { get; private set; }
    public string  VarastonNimi { get; set; }
    public Varasto(string nimi)
    {
        VarastonNimi = nimi;
        Tuotteet = new List<Tuote>();
    }

    public void LisääTuote(Tuote t)    {
        // toteutus: lisää t Tuotteet kokoelmaan Add-metodilla
    }

    public bool PoistaTuote(int tnro)    {
        // etsi parametrin mukainen tuote ja poista se kokoelmasta vaikka  RemoveAt-metodilla ja palauta true
        // jos ei löydy poistettavaa niin palauta false
        return false;
    }
    public bool PoistaTuote(Tuote t)    {
        // etsi parametrin mukainen tuote ja poista se kokoelmasta vaikka RemoveAt-metodilla tai vaikka Remove-metodilla
        // paluuarvo kuten toisessa PoistaTuote-metodissa
        return false;
    }

    public decimal VarastonArvo()
    {
        return 0M; // Laske kaikkien tuotteiden hinta yhteen ja palauta se
    }
```

Testeri eli Main-metodi

1. Luo ensin Varasto-luokka (nimi vaikka "Keskusvarasto")
2. Tee kolme tuotetta kolmella erilaisella alustustavalla
   1. oletuskonstruktori ja ominaisuuksien asetus
   2. konstruktorilla jolla on kolme parametria
   3. oletuskonstruktori ja alustus 'Object Initializer' –tekniikalla (jotenkin näin: new Tuote() { Nimi = "Koe")
3. Lisää nämä tuotteet Varastoon
4. Tulosta kaikki tuotteet
5. Tulosta varaston arvo
6. Testaa
7. Poista Hiiri ja tulosta uudelleen tuotteet ja varaston arvo

```
Tuotteiden tiedot:
tnro     nimi              hinta
1001     Jatkojohto        100
1020     Hiiri             9.99
3010     Mukiteline        19.99
```

Alennustuote

1. Lisää projektiin uusi luokka nimeltään Alennustuote
2. Lisää luokkaan ominaisuus Alennusprosentti (float-tyyppinen)
3. Lisää tarvittavat/sopivat konstruktorit, kantaluokka (Tuote) siis tallettaa hinnan tuotenumero ja nimen
4. Tee uusi toteutus Alennustuote-luokkaan LaskeHinta-metodista (override!) jossa lasket tuotteen alennetun hinnan (prosenttilaskua siis tarvitaan, palautetaan kouluajat mieleen)
5. toteuta ToString()-metodi ja mieti mitä pitäisi tulostaa
6. Siirry testerin puolelle luo Alennustuote tiedoilla (2345, "Tamagotchi", 100.0, 80.0) ja lisää se varastoon
7. suorita testeri ja tarkista että kaikki tiedot tulostuu oikein, myös se alennettu hinta

Tilaustuote

1. Lisää projektiin uusi luokka nimeltään Tilaustuote
2. Lisää luokkaan ominaisuus Tilauskulut (decimal) niin että jos ei erikseen määritellä tilauskulun arvoa niin se on 25€. Tee tätä varten vakio
   private const decimal VakioTilausMaksu = 25M;
   ja käytä sitä koodissa
3. Toteuta:
   1. konstruktori(t)
   2. LaskeHinta
   3. ToString
4. Luo testerissä kaksi uutta tilaustuotetta niin että toisessa et erikseen määrittele tilauskulua vaan koodin pitäisi silloin käyttää oletusarvoa
5. Lisää tuotteet varastoon, suorita ja tarkista että hinta sekä varaston kokonaisarvo tulostuvat oikein

Enum

- Tähän saakka on laskettu hinnat ilman ALV:tä
- Suomessa on käytössä viisi verokantaa (https://www.veronmaksajat.fi/luvut/Tilastot/Kulutusverot/Arvonlisavero/)
- Tee näitä vastaavat enum-määrittelyt: Yleinen24, ALV_14, ALV_10, ALV_0 ja Vapautus pääset helpoimmalla kun lisäät enum-määrittely Tuote-luokan tiedostoon
- Lisää Tuote-luokkaan ominaisuus jonka tyyppi on Verokanta ja nimi ALV
- Oletuksena olkoot Yleinen24 jos ei erikseen tuotteelle määritellä verokantaa
- Korjaa LaskeHinta-metodia niin että se huomioi tuotteelle määritellyn verokannan hinnan laskennassa eli lisää hintaa ALV-prosentin

```
enum Verokanta
{
    Vapautus,
    ALV_0,
    ALV_10,
    ALV_14,
    Yleinen24,
}
```

Lisätehtävä - Dictionary

- Tämä on tekninen harjoitus Dictionary-kokoelman käytöstä, oikeassa elämässä tämän harjoituksen toiminnallisuus toteutettaisiin toisella tavalla
- Lisää Varasto-luokkaan:
  - kokoelma private Dictionary<string, Tuote> luokitellut;
  - metodi PäivitäPerusluokitus joka etsii kalleimman ja halvimman tuotteen ja lisää ne Dictionaryyn avaimilla "Kallein" ja "Huokein"
  - metodi public Tuote LuokiteltuTuote(string luokitus) joka palauttaa parametrina annetun 'luokituksen' mukaisen tuotteen ja jos ei löydy ko. tuotetta niin silloin palauttaa null-arvon
- Onko PäivitäPerusluokitus –pakollinen? No ei, jos aina kun lisäät tai poistat tuotteita varastosta tarkistat ja päivität luokittelutilanteen
- Muuta testerin koodia niin että saat testattua tämän toiminnon
- Toimiiko vielä oikein jos poistat varastosta kaikki tuotteet?

Rajapinta - Interface

- Lisää projektiin rajapinta IAlennus, ja siihen kaksi metodia:
  - void LisäAlennus(float prosenttia);
  - void MuutaAlennus(float alkuperäinen, float uusiAlennus);
- Toteuta tämä rajapinta Alennustuote-luokkaan niin että LisäAlennus laskee uuden alennetun hinnan (taas harmittavasti prosenttilaskua, jos se kiusaa niin voit pelkästään lisätä uuden prosentin alkuperäiseen alennukseen. MuutaAlennus muuttaa alennusprosentin uudeksi jos tuotteen alkuperäinen alennusprosentti on sama kuin parametrin alkuperäinen arvo
- Lisää Varasto-luokkaan metodi LaskeHintoja(float prosenttia) ja sen toteutuksessa käy läpi kaikki tuotteet ja jos tuotteella on IAlennus-rajapinta niin päivitä alennusprosentti
- Miten voisit testata vielä tuon toisen metodin rajapinnassa?