Osa II

Johdanto C++ -kieleen

Oppitunnit

Osa

II

6. oppitunti

Perusluokat

Luokat ovat C++ -kielen ominaisuus, joka tukee monimutkaisten reaalimaailman ilmiöiden ja ongelmien esittämistä ja ratkaisemista. Tässä luvussa käsittelemme seuraavia asioita:

- Mitä luokat ja oliot ovat
- Kuinka uusi luokka määritellään ja kuinka luodaan tuon luokan olioita

Uusien tyyppien luominen

Olet jo saanut tietoa useista eri muuttujatyypeistä kuten kokonaisluvuista ja merkeistä. Muuttujan tyyppi kertoo paljon itse muuttujasta. Esimerkiksi, jos muuttujat Height ja Width ovat tyyppiä unsigned short integer, niihin voidaan tallentaa lukuarvo väliltä 0 - 65535 (olettaen, että tietotyypin koko on 2 tavua).

Itse asiassa se juuri onkin unsigned short integer -tyypin määrittely. Koon lisäksi tietotyyppi kertoo myös kohteen kyvyistä. Kaksi etumerkitöntä kokonaislukua voidaan esimerkiksi laskea yhteen. Siten muuttujien Height ja Width esitteleminen etumerkittöminä kokonaislukuina kertoo samalla,

että Height on mahdollista lisätä Width-muuttujaan ja tuollainen luku voidaan sijoittaa toiseen vastaavaan lukuun.

Tällaisten perusmuuttujien tyyppi kertoo:

- Niiden koon muistissa
- Millaista tietoa niihin voidaan tallentaa
- Mitä toimintoja niillä voidaan toteuttaa

Mikä on tyyppi?

Tyyppi on ikään kuin luokitus. Luokitteleminen on juuri ihmiselle luonteenomainen piirre. Emme esimerkiksi näe savannilla satoja erilaisia hahmoja, vaan mahdollisesti eläimiä ja puita. Emme näe pelkästään yksittäisiä eläimiä vaan ehkäpä gaselleja, kirahveja, norsuja jne. Teemme erilaisia ryhmittelyjä ja luokitteluja ja näemme yhteisiä piirteitä eri kohteilla. Itse asiassa ajattelemme koko ajan eri tyyppisiä asioita.

Appelsiini on sitrushedelmä, joka on taas hedelmäpuu. Hedelmäpuu on taas puu ja puu on kasvi. Jokapäiväisiä tyyppejä ovat esimerkiksi auto, talo, henkilö ja muoto. C++ -kielessä tyyppi on olio, jolla on koko, tila ja joukko ominaisuuksia.

C++ -ohjelmoija voi luoda minkä tahansa tarvittavan tyypin ja kullakin uudella tyypillä voi olla kaikki sisäänrakennettujen tyyppien ominaisuudet ja toiminnallisuus.

Miksi luoda uusia tyyppejä?

Ohjelmia kirjoitetaan yleensä ratkaisemaan todellisia ongelmia, kuten pitämään kirjaa työntekijöistä tai simuloimaan lämmitysjärjestelmää. Vaikkakin on mahdollista ratkaista monimutkaisia ongelmia käyttämällä pelkästään kokonaislukuja ja merkkejä, on paljon helpompaa selvittää laajoja, monimutkaisia ongelmia, jos voidaan luoda niiden olioesityksiä. Toisin sanoen lämmitysjärjestelmän simulointi on helpompi toteuttaa, jos voidaan luoda muuttujia, jotka esittävät huonetta, antureita, termostaatteja ja boilereita. Mitä lähemmin nuo muuttujat vastaavat todellisuutta, sitä helpompaa on kirjoittaa ohjelmia.

Luokat ja jäsenet

Uusi käsite Uusi tyyppi luodaan esittelemällä luokka. Luokka on joukko (usein eri tyyppisiä) muuttujia yhdistettynä joukkoon vastaavia funktioita.

Esimerkiksi auton voimme ajatella olevan joukko pyöriä, ovia, istuimia, ikkunoita, jne. Toinen tapa nähdä auto on ajatella sen toimintaa: se voi liikkua, kiihtyä, hidastaa, pysähtyä, jne.

Uusi käsite Kapselointi on keino sitoa yhteen kaikki yksikön tieto, ominaisuudet ja kyvyt yhdeksi yksittäiseksi kohteeksi.

Kapseloimalla kaikki, mitä autosta tiedetään, yhteen luokkaan, saadaan paljon etua ohjelmoijalle. Kaikki on paikallaan, jolloin kohteeseen voidaan viitata, sitä voidaan kopioida ja tietoa voidaan käsitellä.

Uusi käsite Luokkasi asiakkaat ovat muita luokkia tai funktioita, jotka käyttävät luokkaasi. Kapselointi sallii asiakkaiden käyttää luokkaa ilman, että niiden täytyy tietää, kuinka luokka toimii. Voit ajaa autoa ymmärtämättä, kuinka polttomoottori toimii ja luokkasi asiakkaat voivat käyttää luokkaasi tietämättä, kuinka tietyt toiminnot on luokan sisällä toteutettu. Asiakkaiden on tiedettävä pelkästään, mitä luokka tekee, ei miten asiat on toteutettu.

Luokka voi koostua mistä tahansa muuttujatyyppien ja myös muiden luokkatyyppien yhdistelmästä. Luokan muuttujat ovat luokan jäsenmuuttujia tai tietojäseniä. Autoluokalla voi olla jäsenmuuttujia, jotka edustavat istuimia, radiota, renkaita jne.

Uusi käsite Jäsenmuuttujat eli tietojäsenet, ovat luokan muuttujia. Ne ovat osa luokkaa aivan kuin pyörät ja moottori ovat osa autoa.

Uusi käsite Luokan funktiot käsittelevät tyypillisesti jäsenmuuttujia. Niitä kutsutaan jäsenfunktioiksi tai metodeiksi. Autoluokan metodeita voivat olla Käynnistä() ja Jarruta(). Kissaluokan tietojäseniä saattavat olla ikä ja paino ja metodeja taas Nuku(), Nauku() ja HaeHiiri().

Jäsenfunktiot eli metodit ovat luokan funktioita. Jäsenfunktiot ovat osa luokkaa aivan kuin jäsenmuuttujatkin. Ne määrittävät, mitä luokan oliot voivat saada aikaan.

Luokan esittely

Luokka esitellään avainsanalla class, jonka jälkeen tulee aloittava aaltosulku ja luettelo tietojäsenistä sekä metodeista. Esittely päättyy lopettavaan aaltosulkuun ja puolipisteeseen. Seuraavassa on Cat-luokan esittely:

```
class Cat
{
  unsigned int itsAge;
  unsigned int itsWeight;
  Meow()
};
```

Luokan esittely ei vielä varaa muistia luokalle. Se vain kertoo kääntäjälle, millainen luokka on: mitä tietoja se sisältää (itsAge ja itsWeight, ikä ja paino) ja mitä se osaa tehdä (Meow(), naukuminen). Esittely kertoo vielä kääntäjälle, kuinka paljon tilaa kukin luotava Cat-luokan edustaja vie muistista. Tämän esimerkin mukaan tilaa tarvittaisiin 4 tavua, koska kokonaisluku vie (yleensä) 2 tavua ja luokan esittelyssä on kaksi kokonaislukumuuttujaa. Funktio Meow() ei vie lainkaan tilaa, koska jäsenfunktioille (metodeille) ei varata tilaa.

Hieman nimeämissäännöistä

Kaikki jäsenmuuttujat, jäsenfunktiot ja luokat on tietenkin nimettävä. Kuten luvussa 3 kerrottiin, tulee nimien olla helposti ymmärrettäviä ja merkityksellisiä. Cat, Rectangle ja Employee ovat hyviä luokan nimiä. Meow(), ChaseMice() ja StopEngine() ovat hyviä funktioiden nimiä, koska ne kertovat, mitä funktiot tekevät. Monet ohjelmoijat nimeävät jäsenmuuttujat etuliitteellä its, kuten itsAge, itsWeight ja itsSpeed. Tällainen menettely helpottaa jäsenmuuttujien erottamisen tavallisista muuttujista.

C++ tunnistaa isot ja pienet kirjaimet, mikä tulee muistaa luokan nimeämisessäkin eli on paras käyttää yhtenevää ja loogista nimeämistapaa. Kuinka muuten olisit varma, oliko luokan nimi esimerkiksi rectangle, Rectangle vai RECTANGLE. Jotkut ohjelmoijat sijoittavat aina tietyn merkin luokan nimen eteen, kuten esimerkiksi cCat tai cPerson, kun taas jotkut käyttävät joko pelkästään pieniä kirjaimia tai pelkästään suuria kirjaimia. Kirjassa käytän tapaa, jossa luokan nimet alkavat isolla.

Funktioiden nimet kirjoitetaan usein isolla etukirjaimella ja muuttujien nimet pienellä etukirjaimella. Sanat erotetaan yleensä alaviivalla (kuten sanassa Chase_Mice) tai käyttämällä isoa kirjainta erottimena (kuten sanoissa ChaseMice tai DrawCircle).

Tärkeintä on omaksua yksi nimeämiskäytäntö ja käyttää sitä sitten johdonmukaisesti. Myöhemmin voit kehittää ohjelmointityyliäsi lisää ottamalla mukaan sisennykset, aaltosulkujen asettelun ja kommentoinnin.

Olion määrittely

Uuden tyypin mukainen olio määritellään aivan kuin mikä tahansa tavallinen muuttuja:

```
Unsigned int GrossWeight;    // määrittele unsigned int -
muuttuja
Cat Frisky; // määrittele Cat-luokan olio eli Cat-tyyppiä oleva
muuttuja
```

Ensin määritellään muuttujat GrossWeight, joka on tyyppiä unsigned int. Seuraavalla rivillä määritellään muuttuja Frisky, joka on Cat-luokan (tyypin) mukainen olio.

Luokat ja oliot

Sinulla ei ole lemmikkinä kissan määrittely vaan yksilöllinen kissa. Voit erottaa ajatuksissasi käsitteen kissa siitä tietystä kissasta, joka makailee olohuoneessasi. Samalla lailla C++ erottaa luokan Cat, joka on käsite, ja kunkin yksilöllisen Cat-olion. Siten Frisky on Cat-luokan olio aivan samoin kuin GrossWeight on tiettyä tyyppiä oleva muuttuja.

Luokan jäsenien käsittely

Kun olet määritellyt todellisen Cat-olion (esimerkiksi Friskyn), voit käyttää pisteoperaattoria tuon olion jäsenten käsittelyyn. Sijoittaaksesi arvon 50 Friskyn Weight-jäseneen, kirjoitat siis:

```
Frisky.Weight = 50;
```

Samalla lailla kutsuaksesi Meow()-funktiota kirjoitat

```
Frisky.Meow();
```

Tee sijoitukset olioihin, ei luokkiin

C++ -kielessä ei arvoja sijoiteta tyyppeihin; niitä sijoitetaan muuttujiin. Et voisi kirjoittaa esimerkiksi

```
int = 5;        // väärin
```

Kääntäjä antaisi heti virheilmoituksen tuosta menettelystä. Sen sijaan sinun tulee määritellä kokonaislukumuuttuja ja sijoittaa arvo siihen:

```
int x;  // määrittele int-muuttuja, x
x = 5;  // sijoita x:ään arvo 5
```

Lauseet kertovat "Sijoita 5 muuttujaan x, joka on int-tyyppiä". Emme voisi myöskään kirjoittaa

```
Cat.age = 5;
```

Sen sijaan meidän on määriteltävä Cat-olio ja sijoitettava siihen arvo 5:

```
Cat Frisky;     // kuten int x;
Frisky.age = 5; // kuten x = 5;
```

Private ja public

Uusi käsite Luokan esittelyssä käytetään myös muita määreitä. Tärkeimmät noista lisämääreistä ovat public (julkinen) ja private (yksityinen).

Kaikki luokan jäsenet - tietojäsenet ja metodit - ovat yksityisiä oletusarvoltaan. Yksityisiä jäseniä voidaan käsitellä vain luokan omien metodien avulla. Julkisia jäseniä voidaan käsitellä minkä tahansa luokan olion kautta. Tämä eroavaisuus on sekä merkittävää että hieman vaikeaselkoista. Tehdäksemme asian selkeämmäksi pohtikaamme hieman aiempaa esimerkkiä:

```
class Cat
{
  unsigned int itsAge;
  unsigned int itsWeight;
  Meow();
};
```

Tässä esittelyssä ovat itsAge, itsWeight ja Meow() yksityisiä, koska yksityisyys on oletuksena. Ellet tee mitään lisämäärityksiä, on siis yksityisyys voimassa.

Kuitenkin, jos kirjoitat:

```
Cat Boots;
Boots.itsAge = 5; //virhe! Et voi käsitellä yksityistä tietoa!
```

Kääntäjä antaa tuon virheilmoituksen. Kääntäjä sallii käsitellä luokan jäseniä itsAge, itsWeight ja Meow() vain Cat-luokan funktioiden kautta. Esimerkissä on kuitenkin käsitelty tietoa Cat-metodin ulkopuolelta. Se, että Boots on Cat-luokan olio, ei vielä riitä private-tyyppisten Boots-jäsenten käsittelyyn.

Jotta voisit käsitellä tietojäseniä on Cat-esittelyssä esiteltävä public-tyyppinen osa:

```
class Cat
{
public:
  unsigned int itsAge;
  unsigned int itsWeight;
  Meow();
};
```

Nyt kaikki jäsenet ovat julkisia eli public-tyyppisiä ja lause Boots.itsAge = 5 on täysin laillinen.

Jäsentiedon tekeminen private-tyyppiseksi

Uusi käsite Yleissääntönä tulisi jäsentieto pitää yksityisenä. Sen sijaan tulee luoda public-tyyppisiä funktioita, jotka tunnetaan nimellä käsittelymetodit, yksityisten jäsenmuuttujien käsittelyyn. Nuo käsittelymetodit ovat jäsenfunktioita, joita muut ohjelman osat kutsuvat päästäkseen käsittelemään yksityisiä jäsenmuuttujia.

Käsittelyfunktiot mahdollistavat tiedon tallennuksen yksityiskohtien erottamisen tiedon käyttämisestä. Tällöin voit muuttaa tiedon tallennustapaa tarvitsematta kirjoittaa funktioita uudelleen.

Luokan sisälle voidaan siis esitellä sekä public- että private-tyyppisiä osia lisämääreiden public ja private avulla. Määreet ovat voimassa esittelyssä aina seuraavaan määreeseen saakka tai luokan esittelyn loppuun saakka, jos määre on viimeisenä esittelyssä. Luokan esittely päättyy sulkevaan aaltosulkuun ja puolipisteeseen.

Katsokaamme seuraavaa esimerkkiä:

```
class Cat
{
public:
  unsigned int Age;
  unsigned int Weight;
  Meow();
};

Cat Frisky;
Frisky.Age = 8;
Frisky.Weight = 18;
Frisky.Meow();
```

Seuraavassa on toinen esimerkki:

```
class Car
{
public:     //seuraavat 5 ovat julkisia
  void Start();
  void Accelerate();
  void Brake();
  void SetYear(int Year);
  int GetYear();
private:        //loput ovat yksityisiä
  int Year;
  char Model[255];
};      //luokan esittely päättyy
```

```
Car OldFaithful;        //luokan Car olio
Int bought;         // paikallinen int-muuttuja
OldFaithful.SetYear(84);        //vuosi on 84
bought = OldFaithful.GetYear(); //bought on 84
OldFaithful.Start();            //kutsutaan start-metodia
```

Luokan metodien toteuttaminen

Jokainen esitelty luokan metodi tulee myös määritellä.

Uusi käsite Jäsenfunktion määrittely alkaa aina luokan nimellä, jota seuraa kaksi kaksoispistettä, funktion nimi sekä parametrit. Listaus 6.1 esittelee yksinkertaisen Cat-luokan ja sen käsittelyfunktion toteutuksen sekä yhden yleisen jäsenfunktion toteutuksen.

Listaus 6.1. Luokan metodien toteuttaminen

```
1:  // Esittelee luokan esittelyn
2:  // ja metodien määrittelyn
3:
4:  #include <iostream.h>      // laskentaan
5:
6:  class Cat                   // aloitetaan esittely
7:  {
8:  public:                   // julkinen osa
9:    int GetAge();           // käyttöfunktiot
10:   void SetAge (int age);
11:   void Meow();            // yleinen funktio
12: private:                  // private-osa
13:   int itsAge;             // jäsenmuuttuja
14: };
15:
16: // GetAge, palauttaa
17: // itsAge-jäsenen arvon
18: int Cat::GetAge()
19: {
20:   return itsAge;
21: }
22:
23: // SetAge-määrittely, se
24: // asettaa
25: // itsAge-arvon
26: void Cat::SetAge(int age)
27: {
28:   // asetetaan itsAge-arvoksi
29:   // age-muuttujassa viety arvo
30:   itsAge = age;
31: }
32:
33: // Meow-määrittely
34: // palauttaa: void
35: // parametrit: ei
36: // toiminta: tulostaa "meow" näytölle
37: void Cat::Meow()
38: {
```

```
39:    cout << "Meow.\n";
40:  }
41:
42:  // luodaan Cat-olio, asetetaan sen ikä ja
43:  // tulostetaan "meow", kerrotaan ikä ja tulostetaan "meow" uudelleen.
44:  int main()
45:  {
46:    Cat Frisky;
47:    Frisky.SetAge(5);
48:    Frisky.Meow();
49:    cout << "Frisky is a cat who is " ;
50:    cout << Frisky.GetAge() << " years old.\n";
51:    Frisky.Meow();
52:    return 0;
53:  }
```

Tulostus

```
Meow.
Frisky is a cat who is 5 years old.
Meow.
```

Analyysi

Rivit 6-14 sisältävät Cat-luokan määrittelyn. Rivi 8 sisältää avainsanan public, joka kertoo kääntäjälle, että seuraavassa on joukko julkisia jäseniä. Rivillä 9 on julkisen käsittelyjäsenen GetAge() esittely. Sen avulla päästään käsittelemään yksityistä itsAge-jäsentä, joka on esitelty rivillä 13. Rivillä 10 julkinen käsittelyfunktio nimeltä setAge(), joka ottaa kokonaislukuargumentin ja sijoittaa sen itsAge-jäsenmuuttujaan.

Rivi 12 aloittaa private-osan, joka sisältää itsAge-muuttujan esittelyn rivillä 13. Luokan esittely päättyy sulkevaan aaltosulkuun ja puolipisteeseen rivillä 14.

Rivit 18-21 sisältävät jäsenfunktion GetAge() määrittelyn. Metodi ei ota parametreja ja se palauttaa kokonaisluvun. Huomaa, että luokan metodit sisältävät luokan nimen, jota seuraa kaksi kaksoispistettä sekä funktion nimi (katso riviä 18). Tuo kirjoitustapa kertoo kääntäjälle, että määriteltävä GetAge()-funktio on Cat-luokassa esitelty funktio. Otsikko-osaa lukuunottamatta GetAge()-funktio luodaan aivan samoin kuin mikä tahansa muu funktio.

GetAge()-funktio on yhdellä rivillä; se palauttaa itsAge-arvon. Huomaa, että main() ei voi käsitellä itsAge-muuttujaa, koska se on private-tyyppinen Cat-luokassa. Sen sijaan main() voi kutsua julkista metodia GetAge(). Koska GetAge() on Cat-luokan jäsenfunktio, se voi käsitellä itsAge-muuttujaa. Näin GetAge() voi palauttaa itsAge-arvon main()-funktioon.

Rivi 26 sisältää SetAge()-jäsenfunktion määrittelyn. Se ottaa kokonaislukuparametrin ja asettaa sen itsAge-arvoksi rivillä 30. Koska se on Cat-luokan jäsen, se voi käsitellä itsAge-muuttujaa.

Rivi 37 aloittaa Cat-luokan Meow()-metodin määrittelyn eli toteutuksen. Se on yhden rivin funktio ja tulostaa sanan Meow ja rivinvaihdon näytölle (muistathan, että \n merkitsee rivinvaihtoa).

Rivi 44 aloittaa ohjelman main()-funktiolla. Tässä tapauksessa se ei ota argumentteja ja palauttaa void-arvon. Rivillä 46 esittelee main() Cat-olion nimeltä Frisky. Rivillä 47 sijoitetaan arvo 5 itsAge-jäsenmuuttujaan metodin SetAge() avulla. Huomaa, että metodia kutsutaan käyttämällä luokan nimeä (Frisky) sekä jäsenoperaattoria (pisteoperaattoria .) sekä metodin nimeä (SetAge()). Samalla lailla voidaan kutsua mitä tahansa muita luokan metodeita.

Rivi 48 kutsuu Meow()-jäsenfunktiota ja rivi 49 tulostaa viestin käyttäen GetAge()-metodia. Rivi 51 kutsuu Meow()-metodia uudelleen.

Muodostajat ja tuhoajat

Kokonaislukumuuttuja voidaan määritellä kahdella tavalla. Voit määritellä muuttujan ja sijoittaa siihen arvon myöhemmin ohjelmassa. Esimerkiksi:

```
int Weight;     //määrittele muuttuja
…       // muu koodi
Weight = 7; // sijoita muuttujaan arvo
```

Voit myös määritellä kokonaislukumuuttujan ja alustaa sen samalla:

```
int Weight = 7; //määrittele ja alusta
```

Alustaminen voidaan tehdä määrittelyn yhteydessä. Mikään ei tietenkään estä muuttamasta arvoa myöhemmin. Alustaminen takaa sen, ettei muuttujassa ole koskaan merkityksetöntä arvoa.

Kuinka sitten voidaan alustaa luokan jäsenmuuttujia? Luokilla on erikoisfunktioita nimeltä muodostimet. Muodostin voi ottaa parametreja, mutta se ei koskaan palauta arvoa - ei edes void-arvoa. Muodostin on luokan metodi, jolla on sama nimi kuin luokalla itsellään.

Aina, kun esittelet muodostimen, sinun on esiteltävä myös tuhoaja. Aivan samoin kuin muodostin luo ja alustaa luokan jäseniä, tuhoaja puhdistaa paikat ja vapauttaa varatun muistin. Tuhoajalla on aina luokan nimi, jota edeltää tilde-merkki (~). Tuhoaja ei ota argumentteja eikä palauta arvoa. Siksi Cat-esittelyssä on rivi

```
~Cat();
```

Oletusmuodostin

Uusi käsite Muodostin, joka ei ota parametreja on oletusmuodostin.

Kun kirjoitat

```
Cat Frisky(5);
```

Käytät Cat-muodostinta, joka ottaa yhden parametrin (tässä tapauksessa arvon 5). Jos kirjoitat kuitenkin

```
Cat Frisky;
```

Kääntäjä sallii sulkumerkkien poisjättämisen ja kutsuu oletusmuodostinta, muodostinta, joka ei ota lainkaan parametreja.

Kääntäjän tarjoamat muodostimet

Jos et esittele lainkaan muodostimia, kääntäjä luo oletusmuodostimen (muista, että oletusmuodostin on muodostin, joka ei ota parametreja).

Kääntäjän tarjoama oletusmuodostin ei tee mitään toimintoja; se on ikään kuin muodostin, joka ei ota lainkaan parametreja ja jonka runko on tyhjä.

Muodostinten suhteen tulee muistaa kaksi tärkeää seikkaa:

Oletusmuodostin on mikä tahansa muodostin, joka ei ota parametreja, olipa se itse esittelemäsi tai kääntäjän tarjoama.
Jos esittelet minkä tahansa muodostimen (parametrien kanssa tai ilman), kääntäjä ei tarjoa oletusmuodostinta sinulle. Siinä tapauksessa on sinun esiteltävä itse oletusmuodostin, jos aiot käyttää sitä.

Jos et onnistu esittelemään tuhoajaa, kääntäjä tarjoaa myös sen käyttöösi. Sekin on rungoltaan tyhjä, eikä tee mitään.

Muista aina, että kun esittelet muodostimen, esittele myös tuhoaja, vaikkei se tekisikään mitään. Vaikka oletustuhoaja toimiikin varmuudella aivan oikein, ei oman esitteleminen aiheuta vahinkoa, mutta se selventää koodia.

Listaus 6.2 sisältää Cat-luokan uuden esittelyn. Nyt muodostin alustaa Cat-olion asettaen sen iän halutuksi. Listauksessa esitellään myös tuhoajan käyttöä.

Listaus 6.2. Muodostimien ja tuhoajien käyttö.

```
1:  // Esittelee muodostimen ja tuhoajan esittelyn
2:  // Cat-luokassa
3:
4:  #include <iostream.h>      // laskentaan
5:
6:  class Cat                   // luokan esittely
7:  {
8:  public:                    // julkinen osa
9:    Cat(int initialAge);     // muodostin
10:   ~Cat();                  // tuhoaja
11:   int GetAge();            // käsittelymetodi
12:   void SetAge(int age);    // käsittelymetodi
13:   void Meow();
14: private:                   // private-osa
15:   int itsAge;              // jäsenmuuttuja
16: };
17:
18: // Muodostimen määrittely
19: Cat::Cat(int initialAge)
20: {
21:   itsAge = initialAge;
22: }
23:
24: Cat::~Cat()                // tuhoaja, ei tee mitään
25: {
26: }
27:
28: // GetAge, palauttaa
29: // itsAge-arvon
30: int Cat::GetAge()
31: {
32:   return itsAge;
33: }
34:
35: // SetAge, asettaa
36: // itsAge-arvon
37:
38: void Cat::SetAge(int age)
39: {
40:   // asetetaan arvoksi
41:   // age-arvo
42:   itsAge = age;
43: }
44:
45: // Meow-määrittely
46: // palauttaa: void
47: // parametrit: ei
48: // toiminta: tulostaa "meow" näytölle
49: void Cat::Meow()
50: {
51:   cout << "Meow.\n";
52: }
53:
54: // luodaan Cat-olio ja asetetaan itsAge
55: // tulostetaan "meow", kerrotaan itsAge-arvo ja tulostetaan "Meow".
56: int main()
57: {
58:   Cat Frisky(5);
59:   Frisky.Meow();
60:   cout << "Frisky is a cat who is " ;
```

```
61:    cout << Frisky.GetAge() << " years old.\n";
62:    Frisky.Meow();
63:    Frisky.SetAge(7);
64:    cout << "Now Frisky is " ;
65:    cout << Frisky.GetAge() << " years old.\n";
66:    return 0;
67: }
```

Tulostus

```
Meow.
Frisky is a cat who is 5 years old.
Meow.
Now Frisky is 7 years old.
Meow.
```

Analyysi

Listaus 6.2 on samanlainen kuin 6.1 lukuunottamatta riviä 9, jossa on esitelty muodostin, joka ottaa kokonaisluvun. Rivi 10 esittelee tuhoajan, joka ei ota lainkaan parametreja. Tuhoajat eivät ota koskaan parametreja. Muodostimet ja tuhoajat eivät kumpikaan palauta arvoa.

Rivit 19-22 esittävät muodostimen toteutuksen. Se on samanlainen kuin jäsenfunktion SetAge() toteutus.

Rivit 24-26 esittävät tuhoajan toteutuksen. Funktio ei tee mitään, mutta määrittely on oltava mukana, jos esittely on aiemmin tehty luokan esittelyssä.

Rivi 58 sisältää Cat-olion, Frisky, toteutuksen. Arvo 5 viedään Friskyn muodostimelle. Ei ole mitään tarvetta kutsua SetAge()-funktiota, koska Friskyn luomisen yhteydessä sai muuttuja itsAge arvon 5, kuten rivi 61 esittää. Rivillä 63 Friskyn itsAge-muuttujaan sijoitetaan arvo 7. Rivi 65 tulostaa uuden arvon.

Yhteenveto

Tässä luvussa opit luomaan uusia tietotyyppejä, luokkia. Opit määrittelemään myös tuota uutta tyyppiä olevia muuttujia eli olioita.

Luokassa on jäsenmuuttujia, jotka ovat eri tyyppiä olevia muuttujia. Luokka sisältää myös jäsenfunktioita eli metodeita. Näitä jäsenfunktioita käytetään jäsenmuuttujien käsittelyyn sekä muihin tehtäviin.

Luokan jäsenet - sekä muuttujat että funktiot - voivat olla joko yksityisiä tai julkisia. Julkisia jäseniä voidaan käsitellä mistä tahansa osasta ohjelmaa. Yksityisiä jäseniä voidaan käsitellä vain luokan funktioiden kautta.

Kysymyksiä ja Vastauksia

K
Kuinka suuri on luokan olio?
V
Olion koko muistissa määräytyy jäsenmuuttujien mukaan. Luokan metodit eivät aiheuta tilan varaamista.
Jotkut kääntäjät varaavat muistia siten, että 2 tavun muuttujat vievät tilaa hieman enemmän kuin 2 tavua. Tarkista asia kääntäjäsi manuaalista, mutta on turha mennä tällä erää yksityiskohtiin.

K
Miksi en voisi tehdä kaikista jäsenmuuttujista julkisia?
V
Kun jäsenmuuttuja on yksityinen, voi luokan asiakas käsitellä muuttujaa tarvitsematta tietää tallennustapaa. Jos esimerkiksi Cat-luokalla on metodi GetAge(), voivat Cat-luokan asiakkaat kysyä kissan ikää tarvitsematta pohtia, millaisessa tietorakenteessa ikää säilytetään tai lasketaanko ikä jollakin tavalla jne.