Szegedi Tudományegyetem

Informatikai Tanszékcsoport

Objektum orientált kiterjesztés A+ programozási nyelvhez

Diplomamunka terve

Készítette:

Bátori Csaba

programtervező matematikus hallgató

Témavezető:

Dr. Kiss Ákos

egyetemi adjunktus

Szeged

2010. november 16.

1. fejezet

Motiváció

SZTE TTIK Informatikai Tanszékcsoport Szoftverfejlesztés tanszéken egy fordítóprogramot készítünk A+ programozás nyelvhez. A+ egy tömb orientál programozási nyelv, amely igen gazdag függvény és operátor készlettel rendelkezik. Elsődlegesen gazdasági jellegű programok készítésére használják, számos kritikus alkalmazás készült A+-ban.

A fordítóprogram Dynamic Language Runtime fölé készül. DLR a Microsoft által készített .NET keretrendszer része, amely további szolgáltatásokat add a Commom Language Runtime-hoz. A DLR fejlesztésének célja, hogy segítségével könnyen készíthessünk dinamikus programozási nyelveket .NET futtatókörnyezethez. Több nyelvet megvalósítottak már DLR-ben, mint például IronRuby, IronPython.

A .NET keretrendszer egy nagy osztálykönyvtárral rendelkezik. Jelenlegi diplomadolgozat célja az lenne, hogy az A+ programozási nyelvet olyan objektum orientált képességekkel ruházzuk fel, amelyek segítségével képes lenne ezt az osztálykönyvtárat felhasználni.

2. fejezet

Objektum orientált kiterjesztés

A formális definíció helyett egy példán keresztül mutatjuk be a kiterjesztést. A bemutatás a komplex számokat reprezentáló osztály segítségével történik, amelyet C#-ban készítünk el.

Objektumok kezelését A+-ban egy függvény segítségével oldjuk meg. A kiterjesztés alapját a Pick nem skaláris függvény adja, aminek egyik esete az amikor egy slotfiller-ből elemeket veszünk ki. A mi objektum orientált függvényük is ezt az elvet próbálja követni, hogy adott nekünk egy objektumunk, és abból vegyünk ki elemeket. Ennek a pontos leírását az egyes alfejezetekben részletezzük.

2.1. Tagváltozók elérése/módosítása

A komplex számok ábrázolása egy valós és egy képzetes egység segítségével történik. Ezért vegyünk is fel a Complex osztályunkba két publikus tagváltozót, ezek reprezentálásra:

```
public partial class Complex
{
   public double real;
   public double imaginary;
}
```

Jelen esetben a két tagváltozó, azért publikus, hogy közvetlenül is el tudjuk érni őket.

Jelöljük az általunk újonnan bevezetett objektum orientált A+ függvényünket: ⊙-val. Továbbá tegyük fel, hogy A+-ban már rendelkezünk egy Complex osztály példánnyal, és most jelölje c változó ezt a példányt.

Akkor ha le szeretnénk kérdezni a c komplex szám valós részét, akkor ezt a következőképpen tehetjük meg:

`real ⊙ c

Ebben az esetben ha a függvény jobb oldalára egy objektumot, a bal oldalára pedig egy szimbolikus konstanst írunk, amelyik reprezentál egy tagváltozót, akkor a függvény hívás eredménye a tagváltozónak az értéke. A+-osan fogalmazva, úgy is mondhatjuk, hogy ”kipickeljük” az adott tagváltozót az adott objektumból.

Amennyiben például új értéket szeretnénk beállítani a képzetes egységnek, akkor ezt a következőképpen tehetjük meg:

(`imaginary ⊙ c) := 5

Most, hogy már tudunk tagváltozókat kezelni, térjünk rá a metódusok kezelésére.

2.2. Metódusok meghívása

Először is bővítsük a Complex osztályunkat egy olyan metódussal, amelyik képes két komplex számot összeadni:

```
public partial class Complex
{
   public Complex Add(Complex complex)
   {
      return new Complex
         (this.real + complex.real,
          this.imaginary + complex.imaginary);
   }
}
```

Továbbá tegyük fel, hogy van még egy komplex objektumunk, amelyet jelöljünk d-vel.

Akkor az Add metódust a következő két lépés során tudjuk meghívni:

f:= `Add ⊙ c

A fenti függvényhívás eredménye egy speciális függvény lesz, ugyanis ez a függvény magába foglalja azt az információt, hogy melyik objektumnak melyik metódusát kell meghívni. Jelen esetben ez a függvény hozzá lesz kötve a c objektumpéldány Add metódusához. Tehát ha meg szeretnénk hívni egy objektumnak egy metódusát, akkor az objektum orientált függvényünk jobb oldalára egy objektumot, a bal oldalára pedig egy szimbolikus konstanst kell írni, amely reprezentálja a metódus nevét.

Ezután nincs más dolgunk csak meghívni az újonnan konstruált függvényünket, amely eredményül visszaad egy új komplex számot:

f{d}

2.3. Statikus tagváltozók kezelése

Statikus változók értékeinek lekérdezése/beállítása az osztály nevének segítségével lehetséges. Bővítsük a Complex osztályunkat egy statikus változóval:

```
public partial class Complex
{
   public readonly static Complex ADDITIVE_IDENTITY =
      new Complex(0,0);
}
```

Ebben az esetben a fenti publikus statikus tagváltozónk csak olvasható, és az összeadás műveletének neutrális eleme. Mivel nincs szükségünk objektumra, hogy elérjük a neutrális elemet, így az objektum orientált függvényünk jobb oldalára egy szimbolikus konstansot írunk, amely reprezentálja az osztály nevét:

`ADDITIVE_IDENTITY ⊙ `Complex

Vizsgáljuk meg a következő alfejezetben a statikus metódusok kezelését.

2.4. Statikus metódusok meghívása

Adjunk egy statikus metódust a Complex osztályunkhoz:

```
public partial class Complex
{
   public static Complex Parse(List<char> complex)
   {
      //Új komplex szám készítése complex char listából.
   }
}
```

Hasonlóan a nem statikus metódusok meghívásához, először konstruálunk egy függvényt, amely magában hordozza azt az információt, hogy melyik osztálynak melyik statikus metódusát hívjuk meg. Ezután alkalmazzuk az eredményül kapott függvényt. A két lépés egy kifejezéssé összevonva a következőképpen néz ki:

(`Parse ⊙ `Complex){'4 + 22i'}

Jelen esetben a Complex osztálynak a Parse metódusát szeretnénk meghívni.Tehát ahogy a fenti zárójeles függvényhívás mutatja, az objektum orientált függvényünk jobb és bal oldalára egy szimbolikus konstanst írunk. A jobb oldali az osztály nevét, míg a baloldali a metódus nevét reprezentálja. Az objektum orientált függvény által konstruált függvény pedig létrehoz egy új komplex számot.

Eddigiek során feltettük, hogy már rendelkezünk objektumokkal, a következő alfejezetben arról beszélünk, hogyan lehetne objektumokat létrehozni A+-ban.

2.5. Objektumok létrehozása

Az objektum orientált függvény monadikus alakja felelős az objektumok konstruálásért. Az eddigi elveket követve először létrehozunk egy olyan függvényt, amely tudja, hogy melyik osztályból szeretnénk példányosítani, majd meghívjuk a megfelelő paraméterekkel. A két lépés egy kifejezésként leírva:

(⊙ `Complex){3;5}

Az objektum orientált függvényünk egy szimbolikus konstanst kap paraméterül, amely reprezentál egy osztály nevet. Jelenleg a Complex osztály nevét. A monadikus függvény eredménye egy ”konstruktor függvény” lesz, amely meghívása után eredményül visszaad egy új komplex számot.