Tehnici Avansate de Programare

Recapitulare și noțiuni de bază

Petru Rebeja, Marius Apetrii

27 Februarie 2020

Facultatea de Matematică
Universitatea Alexandru Ioan Cuza, Iași

Introducere

Ce am discutat data trecută

Ce am discutat data trecută

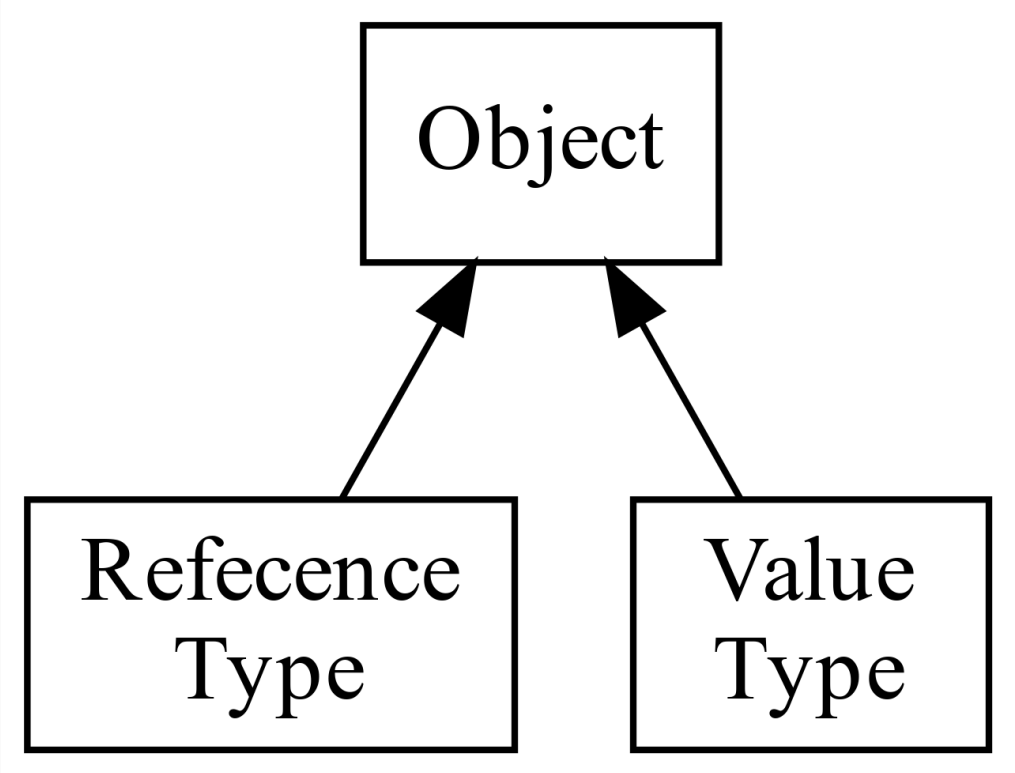
- Ciclul de dezvoltare al aplicațiilor
- Dezvoltarea în iterații
- Unelte de lucru
- Fluxul de lucru Git
- Bune practici

Ce vom discuta azi

- Tipuri de date în `.net`
- Principiile programării orientată-obiect
- Clase abstracte vs. interfețe.
- Acuplare și Coeziune.

Tipuri de date în `.net`

Tipuri referință și tipuri valoare



Tipuri referință și tipuri valoare

Exemple

Tip Referință	Tip Valoare
`string`	`int`
`System.Object`	`float`
`System.Action`	`bool`
`System.IO.MemoryStream`	Tipuri `struct`
`System.Collections.Generic.List<T>`	Tipuri `enum`

Tipuri `valoare`

- O variabilă a unui tip valoare conține o instanță a tipului respectiv.
- La crearea unei instanțe se alocă o singură zonă în memorie.
- Dimensiunea zonei alocate este dată de tip.

Tipuri `referință`

- O variabilă a unui tip referință conține adresa de memorie unde este alocată instanța propriu-zisă.
- La crearea unei instanțe se alocă două zone de memorie:
  - O zonă pentru adresă (stivă),
  - O zonă pentru instanță (memoria heap).
- Dimensiunea zonei alocate pentru instanță poate să varieze (ex. `List<T>`).

Cazul special: `string`

- Deși tipul `string` este un tip referință, acesta se comportă ca un tip valoare.
- Această proprietate se numește `imuabilitate`

Principiile Programării Orientate-Obiect

Recapitulare

- Încapsularea — îmbinarea datelor și metodelor care le procesează.
- Moștenire — proprietățile tipului părinte se păstrează și la copii.
- Polimorfism — același comportament manifestat de mai multe tipuri.

Exemplificare

Context

Pentru exemplificare vom modela o serie de operațiuni bancare.

Exemplificare

Cerință 1

În calitate de posesor al unui cont bancar vreau să pot depune și extrage diverse sume de bani din cont.

Încapsularea

În situația de față, încapsularea ne permite să îmbinăm retragerea de fonduri cu verificarea dacă sunt fonduri suficiente.

Exemplificare

Cerință II

În calitate de posesor de conturi, pot avea conturi de diferite tipuri: economii și debit. Pentru retragerea de fonduri din fiecare cont se percep comisioane diferite în funcție de tipul acestuia:

- Pentru contul de debit: 0 %
- Pentru contul de economii: 0.5 %

Exemplificare

Cerință III

În calitate de posesor de conturi pot avea un cont de credit cu un comision de retragere de 0.7%

Moștenirea

În exemplul dat moștenirea ne permite să declarăm o clasă de bază cu proprietățile comune și să implementăm particularitățile în clasele derivate.

Polimorfismul

Polimorfismul ne permite să aplicăm reguli de calcul diferite pentru aceeași metodă.

Clase abstracte vs. Interfețe

Clasa abstractă

- Oferă o implementare implicită pentru metode și proprietăți.
- Nu se pot crea instanțe ale claselor abstracte.
- Este mai puțin abstractă decât înterfața.
- Poate impune anumite constrângeri (ex. constructorul).
- Un tip de date poate deriva dintr-o singură clasă abstractă.

Interfața

- Nu oferă nicio implementare.
- Cel mai mare grad de abstractizare.
- Nu impune constrângeri decât asupra semnăturii metodelor.
- Un tip de date poate implementa mai multe interfețe.

Modularizarea codului-sursă

Acuplare și Coeziune

Două atribute foarte importante ale unui produs software de succes sunt:

- Grad mic de acuplare,
- Grad mare de coeziune.

Acuplarea

Acuplarea este o măsură a gradului de interdependență dintre modulele unui produs software[1].

[1] https://en.wikipedia.org/wiki/Coupling_(computer_programming)

Acuplarea — exemplu

Implementarea clasică a șablonului `Singleton` este un exemplu de grad înalt de acuplare: clasele care folosesc metodele definite de `Singleton` sunt dependente de acesta.

Coeziunea

Coeziunea este măsura în care elementele unui modul aparțin unul de celălalt[2].

[2] https://en.wikipedia.org/wiki/Cohesion_(computer_science)

Încheiere

Ce am discutat azi

- Tipuri de date în `.net`.
- Principiile programării orientată-obiect.
- Clase abstracte vs. interfețe.
- Acuplare și Coeziune.

Vă mulțumesc!