C# Advanced

Agenda

- Tipuri valoare si referinta
- Aruncarea si prinderea exceptiilor
- Boxing si unboxing
- Utilizarea tipurilor generice
- Delegati si expresii lambda
- LINQ - Language-Integrated Query
- Programare asincrona Task-Based

cegeka

Tipuri de valoare

Tipuri de valoare

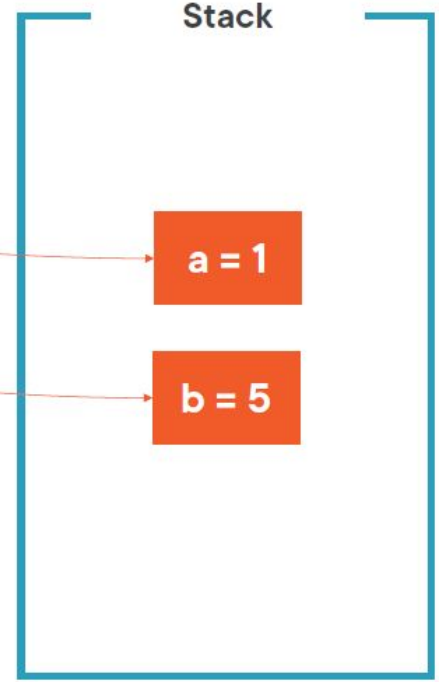
- Tipuri valoare in .NET:
  - **int**, **float**, **double**, **bool**, **char**
  - **structuri** si **enumeratii**

- Tipurile valoare sunt copiate prin valoarea acestora
  - metodele primesc o copie a acestora

- Reguli de alocare si dezalocare
  - Alocare pe **stiva**
  - Dezalocarea atunci cand variabila iese din domeniul de vizibilitate (ex.: bloc **for**, metoda)

```
int a = 1;

int b = 5;
```



cegeka

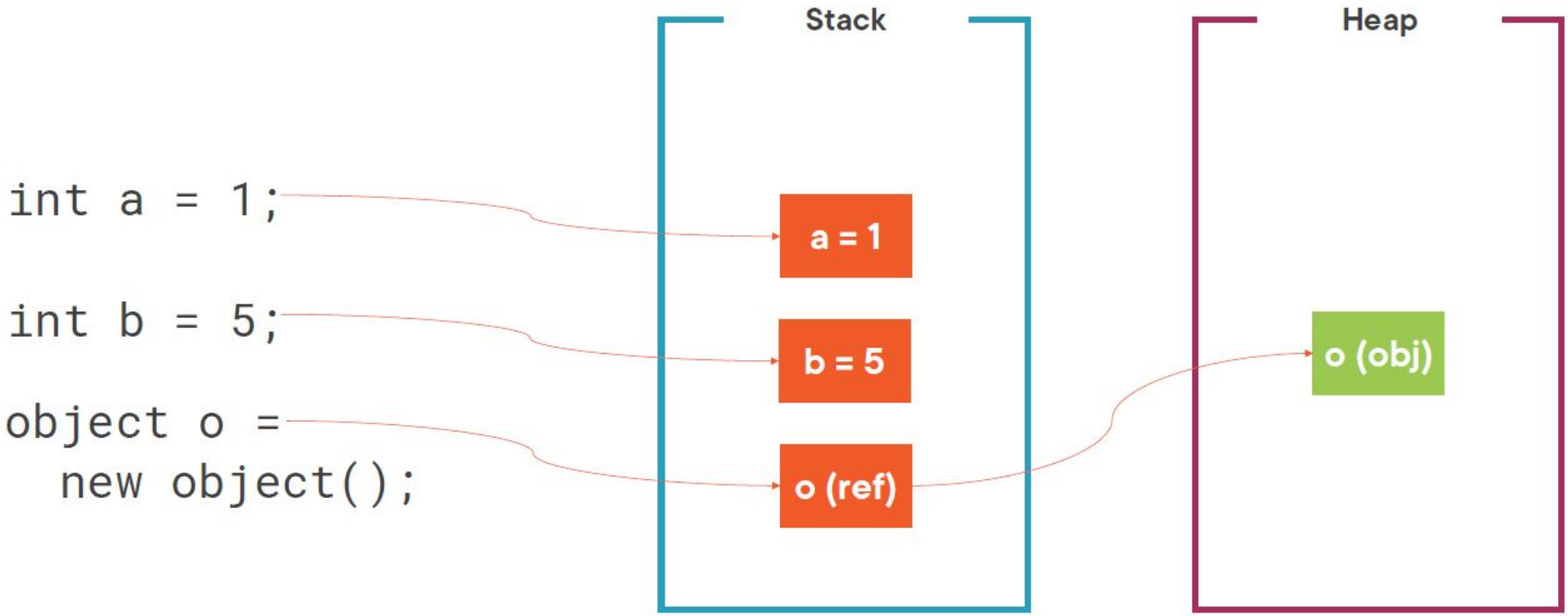
Tipuri de referinta

Tipuri de referinta

- Tipuri referinta in .NET:
  - **clase** si **delegati**

- Tipurile referinta sunt copiate prin referinta acestora
  - Metodele primesc o referinta catre obiectul original

- Reguli de alocare si dezalocare
  - Alocare pe **heap**-ul programului
  - Dezalocarea realizata de **garbage collector**



cegeka

```
Employee emp1 = new Employee();
emp1.firstName = "John";
Employee emp2 = emp1;
emp2.firstName = "Jack";
string check = emp1.firstName;// Ce valoare avem aici ?
```

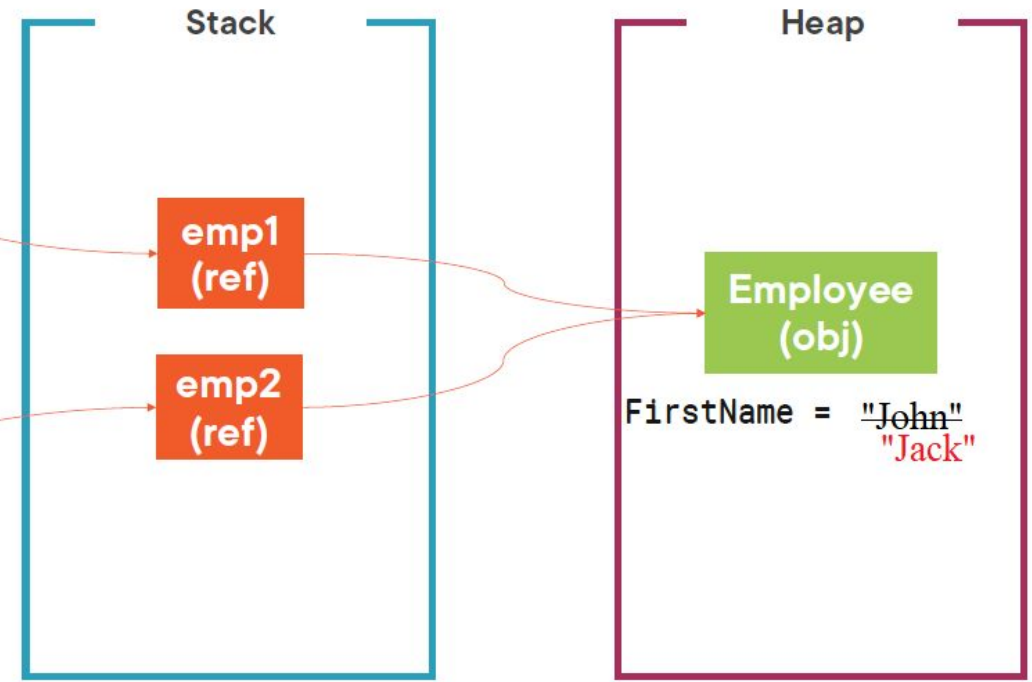
cegeka

```
Employee emp1 =
new Employee();

emp1.FirstName =
    "John"

Employee emp2
= emp1;

emp2.FirstName
=   "Jack"
```



_

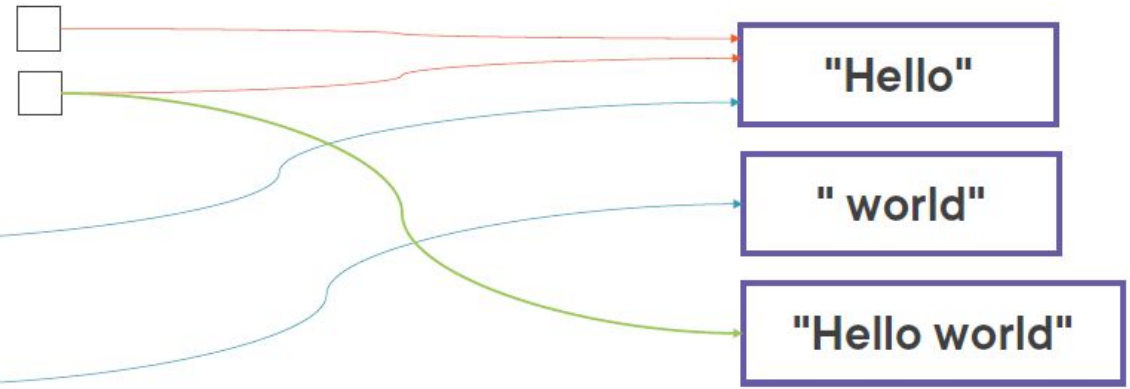
string?

_

```
string a = "Hello";
string b;
b = a;
b += " world";
Console.WriteLine(a);//Output: Hello
```

```
string a;
string b;

a = "Hello";
b = a;
b += " world";
```




```
Console.WriteLine("a = {0}", a);//a = Hello
Console.WriteLine("b = {0}", b);//b = Hello world
```

Aruncarea si prinderea exceptiilor

Aruncarea si prinderea exceptiilor

- Exceptiile se arunca atunci cand se detecteaza conditiile unei erori
  - Exceptia este un obiect care descrie o eroare
  - Toate exceptiile mostenesc **System.Exception**

- Pentru a ridica o exceptie se foloseste **throw**

- Pentru a prinde o exceptie se folosesc blocuri **try/catch** si **finally**

Boxing si unboxing

Boxing si unboxing

- Boxing
  - O variabila de tip valoare este asignata uneia de tip referinta
  - Runtime va crea o copie a obiectului valoare pe heap

- Unboxing
  - Operația inversa “împachetarii”
  - Runtime creaza o copie a obiectului pe stiva

Utilizarea tipurilor generice

Utilizarea tipurilor generice

- Este un tip exprimat sub forma unui parametru tip
  - Un parametru tip este un inlocuitor al tipului actual
  - Codul client va inlocui parametrul tip cu un tip explicit oferit ca argument

- Avantaje
  - Type Safety
  - Performanta
  - Reutilizarea codului
  - Algoritmi generici

- Tipurile generice folosesc sintaxa **ClassName<T>**, unde **T** reprezinta tipul de date pe care clientul trebuie sa il specifice cand foloseste aceasta clasa.

Utilizarea tipurilor generice

- **Dictionary**: colecție tip dicționar puternic tipată
  - Echivalent cu **Hashtable**
- **SortedDictionary**: reține informația sub forma unui arbore binar
- **SortedList**: vector puternic tipat de perechi cheie/valoare

Utilizarea tipurilor generice

- Se pot defini constrangeri asupra tipurilor argumente pentru a specifica ca acestea:
  - Trebuie sa fie valoare
  - Trebuie sa fie referinta
  - Trebuie sa aiba un constructor fara parametrii
  - Trebuie sa fie o clasa specifica sau o subclasa a acestei clase
  - Trebuie sa implementeze o interfata specifica
  - Trebuie sa fie la fel sau să mostenească un al tip argument

Delegati si expresii lambda

Delegati si expresii lambda

- Un tip delegat incapsuleaza semnatura unei metode
  - Util pentru callback methods, apeluri asincrone de metode si apeluri amanate ale metodelor
- Expresiile lambda reprezinta o solutie pentru diferite situatii in care trebuie create o functie cu scopul de a realiza un calcul.
- In momentul compilarii, se creeaza o metoda temporara care contine codul descris de lambda expresie si se realizeaza o identificare locala.
- Imbunatatirea propriu-zisa este ca lambda expresiile nu au nici un nume in cadrul programului si sunt mai usor de scris.
- La executie, lambda expresiile functioneaza asemanator unei metode statice globale.

LINQ - Language-Integrated Query

LINQ - Language-Integrated Query

- Interogheza orice colectie care implementeaza IEnumerable<T> sau IQueryable<T>.
- LINQ consta din mai multe metode extinse peste tiputile care implementeaza IEnumerable<T> sau IQueryable<T>
- LINQ ofera o modalitate simplificata de a interoga si manipula datele indiferent de sursa.

Programare asincrona Task-Based

Programare asincrona Task-Based

- O metoda normala C # poate fi facuta asincrona prin specificarea cuvantului cheie async și a tipului de retur la Task.
- In cadrul metodei asincrone ar trebui sa existe cel putin un bloc await, altfel va va da un avertisment. Pentru a astepta o alta metoda asincrona, veti utiliza cuvantul cheie await inainte de operatie.

Programarea asincrona

Fiecare cuvant await “**desparte**” codul in doua “**blocuri**” de cod:

- Bloc inainte de await
- Bloc dupa await

```
public async Task<HttpResponseMessage> Get(string url)
{
    HttpClient httpClient = new HttpClient();

    HttpResponseMessage result = await httpClient.GetAsync(url);

    if (!result.IsSuccessStatusCode)
    {
        Console.WriteLine($"request failed for {url}");
    }

    return result;
}
```

Inainte de await

Dupa await

cegeka

Programarea asincrona

Fiecare bloc e executat de un anumit Thread

Blocul **Inainte** de await: Threadul curent

Blocul **Dupa** await: Threadul curent **sau** alt thread

```
public async Task<HttpResponseMessage> Get(string url)
{
    HttpClient httpClient = new HttpClient();

    HttpResponseMessage result = await httpClient.GetAsync(url);

    if (!result.IsSuccessStatusCode)
    {
        Console.WriteLine($"request failed for {url}");
    }

    return result;
}
```

Executat de Threadul Curent

Executat de Threadul curent

cegeka

Programarea asincrona

Mai multe cuvinte await in aceeasi metoda

```
public async Task<T> Get<T>(string url)
{
    Console.WriteLine($"Trying to get the value for {url}");

    var httpResult = await Get(url);

    if (httpResult.IsSuccessStatusCode)
    {
        Console.WriteLine($"request succesfull for {url}");

        var stringResult = await httpResult.Content.ReadAsStringAsync();

        return JsonConvert.DeserializeObject<T>(stringResult);
    }

    return default(T);
}
```

Bloc 1

Bloc 2

Bloc 3

cegeka

Programarea asincrona

Fiecare cuvant await “**desparte**” codul in doua “**blocuri**” de cod:

- Bloc inainte de await
- Bloc dupa await

```
public async Task<HttpResponseMessage> Get(string url)
{
    HttpClient httpClient = new HttpClient();

    HttpResponseMessage result = await httpClient.GetAsync(url);

    if (!result.IsSuccessStatusCode)
    {
        Console.WriteLine($"request failed for {url}");
    }

    return result;
}
```

Inainte de await

Dupa await

cegeka

Q&A

Bibliografie

- Toata documentatia C# - https://docs.microsoft.com/en-us/dotnet/csharp/
- Value types - https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-types
- Reference types - https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/reference-types
- string - https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0
- Exceptii - https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/exceptions/
- Boxing/Unboxing - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/boxing-and-unboxing
- Tipuri generice - https://docs.microsoft.com/en-us/dotnet/standard/generics/
- Delegati - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/
- Expresii Lambda - https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions
- Task-uri - https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming

cegeka

Tema

- Implementati o clasa generica **Set<T>** (multime) care sa include metodele
  void Insert(T item),
  void Remove(T item),
  bool Contains(T item),
  Set<T> Merge(Set<T> other)

- Folosind exceptii*, implementati o validare pe metode a.i. setul sa nu poata contine dubluri.

- Implementati metoda Filter, care va lua ca parametru o functie lambda (o conditie) si va returna un subset al setului ce continue doar membrii care respecta conditia.

cegeka

Thank you

cegeka