

КЛАСА OBJECT

Object је корен, основна или супер – класа свих класа. Све предефинисане класе или кориснички дефинисане класе су поткласе класе **Object**. На пример, ако кажемо да је **byte** целобројна променљива са опсегом од 0 до 255, онда можемо да кажемо да је **object** било који тип.

Основне карактеристике **object** типа:

- **Object** је референтни тип, захваљујући томе што је класа
- Сваки тип се може имплицитно конвертовати у **Object**
- **Object** се може експлицитно конвертовати нпр. у изворни тип **int**.

Пример како се дефинише променљива типа **object**, имплицитна и експлицитна конверзија.

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int x = 9;
        object obj1 = x;           //implicitna konverzija
        int y = (int)obj1;         //eksplicitna konverzija
        Console.WriteLine(x);
        Console.WriteLine(y);

        string s = "c#";
        object obj2 = s;
        string t = (string)obj2;
        Console.WriteLine(s);
        Console.WriteLine(t);

        Console.ReadKey();
    }
}
```

Линијом програмског кода **object obj1 = x;** врши се имплицитно претварање типа **int** у тип **object**. Овом конвезијом омогућавамо да се вредносни тип третира као референтни тип.

Линијом програмског кода **int y = (int)obj1;** врши се експлицитна конекверзија у **int**.

197. Дат је код програма у програмском језику C#. У Main() методи декларисане су променљиве **s**, **obj** и **t**. Анализирати декларацију и одредити на који објекат указују променљиве **s**, **obj** и **t**.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args)
        {
            string s = "c#";
            object obj = s;
            string t = (string)obj;
        }
    }
}
```

Заокружити број испред очекиваног одговора:

1. Када се вредност променљиве **s** додељује променљивој **obj** у наредби **object obj = s**, конструише се нови објекат.
2. Када се конвертује тип променљиве **obj** и њена вредност додељује променљивој **t** у наредби **string t = (string)obj**, конструише се нови објекат.
3. Када се конвертује тип променљиве **obj** и њена вредност додељује променљивој **t** у наредби **string t = (string)obj**, садржај променљиве **obj** се мења.
4. Променљиве **s**, **obj** и **t** указују на исти објекат типа **string**.

МЕТОДА EQUALS

Метода **Equals** класе **Object** слична је оператору `==`, осим то је `Equals` виртуелна метода, а оператор `==` је статички оператор.

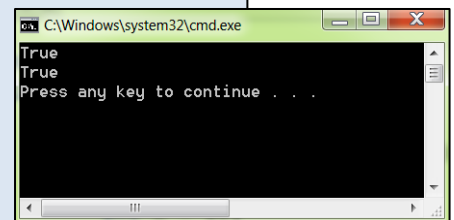
```
...public class Object
{
    ...public Object();
    ...public virtual bool Equals(object obj);
    ...public static bool Equals(object objA, object objB);
    ...public virtual int GetHashCode();
    ...public Type GetType();
    ...protected object MemberwiseClone();
    ...public static bool ReferenceEquals(object objA, object objB);
    ...public virtual string ToString();
}
```

Equals - виртуелна метода

Дакле, метода **Equals** проверава да ли два објекта садрже исте податке. У другом случају ова метода проверава да ли објекти заузимају исти простор у меморији.

1. Задатак: Анализирати пример и одредити шта се приказује?

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int x = 10;
        int y = 10;
        Console.WriteLine(x == y);
        Console.WriteLine(x.Equals(y));
    }
}
```

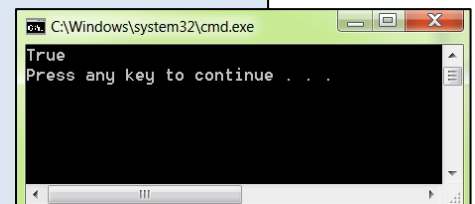


```
C:\Windows\system32\cmd.exe
True
True
Press any key to continue . . .
```

2. Задатак: Анализирати пример и одредити шта се приказује?

```
using System;
class Ucenik {
    string ime;
    string prezime;
    int godina;

    public Ucenik(string x, string y, int z) {
        ime = x;
        prezime = y;
        godina = z;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Ucenik U1 = new Ucenik("Pera", "Peric", 17);
        Ucenik U2 = U1;
        Console.WriteLine(U1 == U2);
    }
}
```

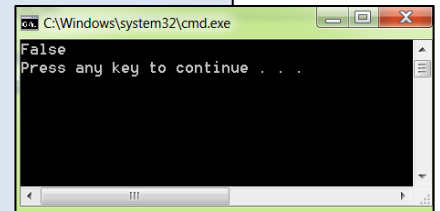


```
C:\Windows\system32\cmd.exe
True
Press any key to continue . . .
```

3. Задатак: Анализирати пример и одредити шта се приказује?

```
using System;
class Ucenik {
    string ime;
    string prezime;
    int godina;

    public Ucenik(string x, string y, int z) {
        ime = x;
        prezime = y;
        godina = z;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Ucenik U1 = new Ucenik("Pera", "Peric", 17);
        Ucenik U2 = new Ucenik("Pera", "Peric", 17);
        Console.WriteLine(U1.Equals(U2));
    }
}
```



На овај начин проверили смо да ли се објекти налазе и истој меморијској локацији (у динамичкој меморији) али још увек не можемо да утврдимо да ли се садржај података подудара. Дакле, потребно је користити Equals методу у класи **Ucenik**:

1. начин:

```
public override bool Equals(object obj)
{
    return (ime == ((Ucenik)obj).ime &&
            prezime == ((Ucenik)obj).prezime &&
            godina == ((Ucenik)obj).godina);
}
```

2. начин:

```
public override bool Equals(object obj)
{
    Ucenik a = (Ucenik)obj;
    return ime == a.ime && prezime == a.prezime && godina == a.godina;
}
```

3. начин:

```
public bool Equals(Ucenik a) {
    return ime == a.ime && prezime == a.prezime && godina == a.godina;
}
```

АНАЛИЗИРАТИ СЛЕДЕЋЕ ПРИМЕРЕ И ДАТИ ТАЧАН ОДГОВОР:

171. Дат је код програма у програмском језику C# којим су дефинисане две класе: `class Program` која садржи `Main(string[] args)` методу и `class A`. Анализирати дати код и одредити да ли је код исправно написан. Понуђени одговори дају опис последица извршавања овог кода. Заокружити број испред тачног исказа.

```
class Program {
    public static void Main(string[] args) {
        A a1 = new A();
        A a2 = new A();
        Console.WriteLine(a1.Equals(a2));
    }
}
class A {
    int x;
    public bool Equals(A a) {
        return this.x == a.x;
    }
}
```

2

1. Програм има грешку, јер се изразом `a1.Equals(a2)` проверава једнакост објеката `a1` и `a2` различитог типа од `Object`.
2. Програм има грешку, јер се једнакост објеката `a1` и `a2` типа `A` проверава изразом `a1 == a2`.
3. Програм се извршава без грешке и приказује се `true` на екрану.
4. Програм се извршава без грешке и приказује се `false` на екрану.

196. У програмском језику C#, метод ***Equals()*** за проверу једнакости два објекта је дефинисан у класи ***Object***. У датом програмском коду у класи ***Klasa*** је надјачан (override-ован) метод ***Equals()***. Анализирати код и проценити тачност извршења.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args) {
            Object obj1 = new Klasa();
            Object obj2 = new Klasa();
            Console.WriteLine(obj1.Equals(obj2));
        }
    }
    class Klasa {
        int x;
        public override bool Equals(object o) {
            Klasa a = (Klasa)o;
            return this.x == a.x;
        }
    }
}
```

2

Заокружити број испред очекиваног одговора:

1. Програм има грешку, јер се изразом `obj1.Equals(obj2)` проверава једнакост објеката `obj1` и `obj2` различитог типа од `Object`.
2. Програм има грешку, јер се једнакост објеката `obj1` и `obj2` типа ***Klasa*** проверава изразом `obj1 == obj2`.
3. Програм се извршава без грешке и приказује се ***true*** на екрану.
4. Програм се извршава без грешке и приказује се ***false*** на екрану

У наведеном примеру класа ***Object*** дефинише нову методу:

```
public virtual bool Equals(object obj);
```

Потребно је у класи ***Klasa*** дефинисати методу која ће надјачати наведену методу:

```
public override bool Equals(object obj);
```

Решење: TRUE

195. У програмском језику C#, метод **Equals(...)** је метод инстанце класе **object** којим се проверава да ли је објект из кога се метод позива једнак неком задатом објекту. Овај метод се може надјачати (override-овати) у наслеђеним класама. Одредити заглавље овог метода у класи **string** у којој би метод био надјачан. Заокружити број испред понуђеног тачног одговора:

1. public override bool Equals(string s)
2. public new bool Equals(string s)
3. public override bool Equals(object obj)
4. public static bool Equals(object obj)
5. public bool Equals(string s1, string s2)

2

194. У програмском језику C#, метод **Equals()** за проверу једнакости два објекта је дефинисан у класи **Object**. У датом програмском коду, у класи **Klasa** је предефинисан метод **Equals()**. Анализирати код и проценити тачност извршења.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args) {
            Object obj1 = new Klasa();
            Object obj2 = new Klasa();
            Console.WriteLine(obj1.Equals(obj2));
        }
    }
    class Klasa {
        int x;
        public new bool Equals(Klasa o) {return this.x == o.x; }
    }
}
```

Заокружити број испред очекиваног одговора:

1. Програм има грешку, јер се изразом obj1.Equals(obj2) проверава једнакост објеката obj1 и obj2 различитог типа од Object.
2. Програм има грешку, јер се једнакост објеката obj1 и obj2 типа **Klasa** проверава изразом obj1 == obj2.
3. Програм се извршава без грешке и приказује се **true** на екрану.
4. Програм се извршава без грешке и приказује се **false** на екрану

2

170. Дат је код програма у програмском језику C# којим су дефинисане две класе: **class Program** која садржи **Main(string[] args)** методу и **class A**. Анализирати дати код и одредити да ли је код исправно написан. Понуђени одговори дају опис последица извршавања овог кода. Заокружити број испред тачног исказа.

```
class Program {
    public static void Main(string[] args) {
        Object a1 = new A();
        Object a2 = new A();
        Console.WriteLine(a1.Equals(a2));
    }
}
class A {
    int x;
    public bool Equals(A a) {
        return this.x == a.x;
    }
}
```

1. Програм има грешку, јер се изразом a1.Equals(a2) проверава једнакост објеката a1 и a2 различитог типа од Object.
2. Програм има грешку, јер се једнакост објеката a1 и a2 типа A проверава изразом a1 == a2.
3. Програм се извршава без грешке и приказује се true на екрану.
4. Програм се извршава без грешке и приказује се false на екрану.

2

Обратити пажњу да ће класа **Object** креирати виртуелну методу која је са различитим потписом у односу на методу дефинисану у класи **Klasa**. **Решење: FALSE**

МЕТОДА TOSTRING

Метода **ToString** класе **Object**, при креирању објекта дефинише виртуелну методу:

```
public virtual string ToString();
```

System.Object → Go To Definition

```
...public class Object
{
    ...public Object();

    ...public virtual bool Equals(object obj);
    ...public static bool Equals(object objA, object objB);
    ...public virtual int GetHashCode();
    ...public Type GetType();
    ...protected object MemberwiseClone();
    ...public static bool ReferenceEquals(object objA, object objB);
    ...public virtual string ToString();
}
```

АНАЛИЗИРАТИ СЛЕДЕЋИ ПРИМЕР И ДАТИ ТАЧНЕ ОДГОВОРЕ:

217. Дат је код програма у програмском језику C#. Код садржи објекте две класе у којима је дефинисан метод **ToString()**. Аналиzirати код датог програма и одредити који од датих исказа су тачни.

```
namespace TestPrimer {
    class Program {
        static void Main(string[] args) {
            Object a = new Klasa();
            Object obj = new Object();
            Console.WriteLine(a);
            Console.WriteLine(obj);
        }
    }
}
class Klasa{
    int x;
    public override string ToString() {return "x u A je " + x;}
}
```

2

Заокружити бројеве испред очекиваних одговора:

1. Програм има грешку, јер наредбу **Console.WriteLine(a)** треба заменити наредбом **Console.WriteLine(a.ToString())**.
2. Приликом извршавања наредбе **Console.WriteLine(a)**, програм позива се метод **ToString()** наслеђен из класе **Object**.
3. Приликом извршавања наредбе **Console.WriteLine(a)**, програм позива метод **ToString()** из класе **A**.
4. Приликом извршавања наредбе **Console.WriteLine(obj)**, програм позива метод **ToString()** из класе **Object**.

Објекат **a** ће позвати методу **ToString()** из класе **Klasa**, пошто ће **override** метода у класи **Klasa** надјачати виртуелну методу класе **Object**.

Код објекта **obj** је чиста ситуација и извршиће се виртуелна метода класе **Object**.

РЕШЕЊЕ: 3. и 4.

