# JAVA/C# SLIČNOSTI/RAZLIKE

## Getters and setters

|  |  |
| --- | --- |
| **JAVA** | **C#** |
| **public** **class** Test {  **public** **int** broj;  **public** String slovo;  **public** Date datum;  **public** **int** getBroj() {  **return** broj;  }  **public** **void** setBroj(**int** broj) {  **this**.broj = broj;  }  **public** String getSlovo() {  **return** slovo;  }  **public** **void** setSlovo(String slovo) {  **this**.slovo = slovo;  }  **public** Date getDatum() {  **return** datum;  }  **public** **void** setDatum(Date datum) {  **this**.datum = datum;  }  } | public class TestC  {  public int Broj { get; set; }  public string Slovo { get; set; }  public DateTime datum { get; set; }  }  **Custom getters and setters**  private string message = "Message";  public string Message  {  get { return "Custom message"; }  set { message = value; }  }  **Pristup i upotreba**  TestC test = new TestC();  test.Broj = 0;  test.Slovo = "Slovo";  test.datum = DateTime.Now; |

## Collections

|  |  |
| --- | --- |
| **JAVA** | **C#** |
| ArrayList<String> lista = **new** ArrayList<String>();  **Iteracija i pristup elementima**  **for**(**int** i = 0; i<lista.size();++i){  lista.get(i);  }  **for**(String str: lista){  System.***out***.println(str);  } | List<string> lista = new List<string>();  **Iteracija i pristup elementima**  for (int i = 0; i < lista.Count; ++i)  {  Console.WriteLine(lista[i]);  }  foreach (string str in lista)  {  Console.WriteLine(str);  } |

Primer upotrebe rečnika **Dictionary** (**HashMap** u Javi) :

public Dictionary<int, int> \_h = new Dictionary<int, int>();

static void Main()

{

// Add items to dictionary.

\_h.Add(5, 4);

\_h.Add(4, 3);

\_h.Add(2, 1);

// Standard foreach loop.

foreach (KeyValuePair<int, int> pair in \_h)

{

Console.WriteLine("{0},{1}", pair.Key, pair.Value);

}

}

## Serialization in C#

[Serializable]

public class TestC

anotacija [Serializable] <=> implementiranje Serializable interfejsa u Javi

**LOAD**

**Tekstualni fajlovi**

foreach (string line in File.ReadLines("c:\\file.txt"))

{

Console.WriteLine("-- {0}", line);

}

**Binarni fajlovi**

TestC test;

//pitanja do foldera gde se nalazi projekat

string serializationFile = Path.Combine(Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory

.GetCurrentDirectory())), "test.bin");

using (Stream stream = File.Open(serializationFile, FileMode.Open))

{

var bformatter = new System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();

test = (TestC)bformatter.Deserialize(stream);

}

**SAVE**

**Tekstualni fajlovi**

string lines = "First line.\r\nSecond line.\r\nThird line.";

// Write the string to a file.

System.IO.StreamWriter file = new System.IO.StreamWriter("c:\\test.txt");

file.WriteLine(lines);

**Binarni fajlovi**

TestC test = new TestC();

string serializationFile =

Path.Combine(Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory.GetCurrentDirectory())),

"test.bin");

//serialize

using (Stream stream = File.Open(serializationFile, FileMode.Create))

{

var bformatter = new System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();

bformatter.Serialize(stream, test);

}

**4. Lambda izrazi**

Lambda izrazi služe za difinisanje anonimnih funkcija:

(parametri) => izraz

Na ovaj način se dobija funkcija koja se  može dodeliti delegatu koji je definisan za iste parametre.

//delegat polje, može mu se dodeliti bilo kakva funkcija sa int parametrom.

delegate int del(int i);

//delegat sa dva parametra.

delegate int del2(int i, int j);

static void Main(string[] args){

del myDelegate = x => x \* x;

int rez = myDelegate(5); //rez = 25

del2 myDelegate2 = (x1 , x2) => x1\*x2;

rez = myDelegate(5,5); //rez = 25

}

Sa desne strane lamda izraza moguće je definisati i blok iskaza, uz obavezne vitičaste zagrade.

x => {

...

   int rez = x\*x;

   Console.WriteLine(rez);

...

}

Primer sortiranja niza pomoću lambda izraza:

List<int> list = new List<int>();

list.Sort(new Comparison<int>((x, y) => x < y));

# Tricks and shortcuts

Ukucate **prop,**  a zatim pritisnete **TAB, TAB** dobije se

public int MyProperty { get; set; }

Ukucate **for,**  a zatim pritisnete **TAB, TAB** dobije se

for (int i = 0; i < length; i++)

{

}

Formatiranje teksta: **CTRL+K,D**

Definicija promenjlive/metode: **F12**

Istorija pozivanja promenjljive/metode: **SHIFT+F12**

Lista taskova (TODO) : **CTRL+W,T**