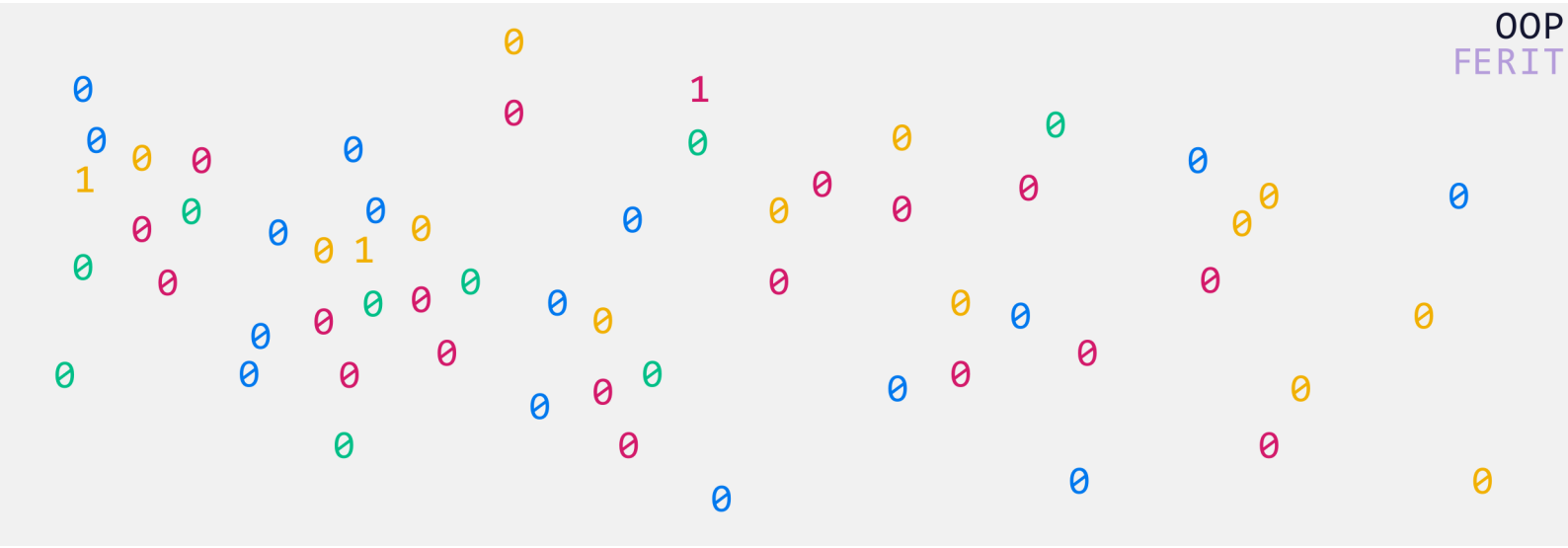


LV6: Lista je u rječniku pod ključem

Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek.
2023.

1 Pripremni list: LV6

Ime i prezime: _____ Grupa: _____ Datum: _____

1. Koja sučelja vezujete uz generičke kolekcije i uz koje su konkretne kolekcije pojedina od njih vezana?

2. Kako se inicijalizira lista cjelobrojnog tipa vrijednostima `1,2,3,4,5`?

3. U kojim je slučajevima prikladno koristiti rječnik?

4. Što su `Func` i `Action` i koja je razlika među njima?

5. Koji je općeniti oblik pisanja lambda izraza?

6. Pronađite, označite i objasnite pogreške u izlistanju:

```
1  public void RunSimpleDemo()
2  {
3      Dictionary<string> colors = new Dictionary<string>();
4      colors.Add("red", new Color(255, 0, 0));
5      Console.WriteLine(colors[0]);
6  }
```

2 Zadaci

Radite na sustavu za prodaju rabljenih automobila. Sustav već sadrži klase potrebne za rad s automobilima, i to klasu koja predstavlja automobil, sučelje koje predstavlja repozitorij dostupnih rabljenih automobila te implementaciju sučelja koja glumi alat za dohvaćanje podataka o automobilima s mreže.

```
1  public class Car
2  {
3      public string VehicleId { get; private set; }
4      public string Make { get; private set; }
5      public string Model { get; private set; }
6      public int Kilometrage { get; private set; }
7      public int BuildYear { get; private set; }
8
9      public Car(string vehicleId, string brand, string type, int kilometrage, int buildYear)
10     {
11         VehicleId = vehicleId;
12         Make = brand;
13         Model = type;
14         Kilometrage = kilometrage;
15         BuildYear = buildYear;
16     }
17
18     public override string ToString() => $"{Make} - {Model} - {Kilometrage} - {BuildYear}";
19
20     public override int GetHashCode() => HashCode.Combine(VehicleId, Make, Model, BuildYear);
21
22     public override bool Equals(object other)
23     {
24         Car car = other as Car;
25         if (car is null) return false;
26         return this.Equals((Car)other);
27     }
28
29     public bool Equals(Car other)
30     {
31         return this.Make == other.Make &&
32             this.Model == other.Model &&
33             this.BuildYear == other.BuildYear &&
34             this.Kilometrage == other.Kilometrage;
35     }
36 }
37
38 public interface ICarRepository
39 {
40     public IEnumerable<Car> GetAll();
41 }
42
43 public class NetworkCarRepository : ICarRepository
44 {
45     private static List<Car> usedCars = new List<Car>
46     {
47         new Car("JT2BF28K0X0158222","Peugeot", "308", 178000, 2011),
48         new Car("1FALP5747PA186161","Volkswagen", "Passat", 129000, 2016),
49         new Car("1J4AA2D10AL156565","Volvo", "V40", 141600, 2016),
50         new Car("1J4FA49S02P795977","Opel", "Insignia", 320395, 2011),
51         new Car("1C6RR7MT2DS539778","Dacia", "Duster", 53509, 2015),
52         new Car("1C3CCCAB0FN527140","Volkswagen", "Golf VI", 170000, 2010),
53         new Car("1FMEU73E96UA79031","Volkswagen", "Golf VI", 149000, 2009),
54         new Car("WBA5A5C57FD506754","Opel", "Astra", 146200, 2007),
55         new Car("1N4AL21E49N496914","Peugeot", "4007", 249800, 2008),
56         new Car("2GCEK19T4Y1166993","Ford", "Fiesta", 158111, 2013),
57         new Car("1FTFW1ET4EKD48588","Dacia", "Sandero", 121000, 2014),
58         new Car("1GF5ACVKXPD186618","Peugeot", "208", 134878, 2016),
59         new Car("WDDGF4HB1EA968951","Opel", "Meriva", 128500, 2010),
60         new Car("5UXZV4C5XD0G75062","Volkswagen", "Golf VII", 178000, 2013),
61         new Car("SHSRD78916U429111","Opel", "Astra", 167000, 2011),
62         new Car("2C8GF68454R687087","Ford", "C-Max", 1203835, 2009),
63         new Car("2LNBL8CVXBX754805","Volkswagen", "Passat", 132700, 2014),
64         new Car("WAUDF78E57A005138","Renault", "Megane", 186650, 2009),
```

```
65          new Car("2G4WC582X81151833","Ford", "Focus", 182459, 2012),
66          new Car("1HGCR2F80EA172625","Hyundai", "i30", 163258, 2013),
67          new Car("KNADC125X46314035","Peugeot", "308", 148000, 2014),
68          new Car("WDDGF4HB0DA755925","Renault", "Megane", 150000, 2016),
69          new Car("19UUA66246A028043","Opel", "Astra", 230000, 2011),
70          new Car("1GNDV23WX8D193588","Dacia", "Duster", 126000, 2015),
71          new Car("2GCEK19T3Y1102766","Renault", "Clio", 118000, 2007),
72          new Car("3N1AB6AP8BL669509","Volkswagen", "Polo", 102000, 2014),
73          new Car("1G1BL837XNW191401","Nissan", "Qashqai", 132000, 2014),
74          new Car("1XP5DB9X66D699186","Volkswagen", "Golf V", 280000, 2005),
75          new Car("2G1FC3D33E9253794","Porsche", "Panamera", 73000, 2013),
76          new Car("1HGCD7256VA017634","Renault", "Scenic", 98895, 2017),
77      };
78      public IEnumerable<Car> GetAll() => usedCars.AsReadOnly();
79  }
```

Zadatak 2.1 (2 boda)**.** Proširite sučelje `ICarRepository` metodom `Filter` koja na temelju predanog delegata tipa `Func<Car,bool>` vraća pobrojenje koje sadrži samo one automobile koje zadovoljavaju uvjet propisan predanim delegatom. Implementirajte navedenu metodu u klasi `NetworkCarRepository`. Ispitajte implementaciju korištenjem dvaju različitih lambda izraza. U prvom slučaju filtriraju se svi automobili s kilometražom unutar raspona učitanog s konzole. U drugom slučaju filtriraju se svi automobili proizvođača čije je ime učitano s konzole, pri čemu ne treba uzimati u obzir velika i mala slova (npr. `VolksWagen`, `Volkswagen`, `volkswagen` svi predstavljaju istog proizvođača).

Zadatak 2.2 (2 boda)**.** Korištenjem metode za dohvaćanje svih automobila iz repozitorija automobila, unutar klase `Tester` napišite testne metode u kojima ćete korištenjem LINQ-a odgovoriti i zatim na ekran ispisati odgovore na sljedeća pitanja:

- Koji su svi dostupni proizvođači automobila, sortirani uzlazno?
- Koji je proizvođač automobila koji je prešao najviše kilometara?
- Koliko su prosječno stare Dacie?
- Koji je raspon godina (*gggg. - gggg.*) u kojem su proizvedeni svi automobili?

Zadatak 2.3 (2 boda)**.** U sustav je potrebno dodati alat za provjeru sudjelovanja u nesrećama pojedinog automobila prilikom kupnje rabljenih automobila, čija je uporaba prikazana izlistanjem na kraju ovog zadatka.

Za tu potrebu najprije je potrebno definirati klasu koja predstavlja izvješće o nezgodi. Svako izvješće o nezgodi čuva identifikator svakog vozila koje je sudjelovalo u nezgodi i uz njega vezan iznos štete koja je na tom vozilu nastala prilikom nezgode. Za opisanu klasu potrebno je omogućiti

- stvaranje praznog izvješća,
- izradu duboke kopije izvješća,
- uključivanje automobila i na njemu učinjene štete u izvješće,

- izračun ukupne štete na svim uključenim automobilima,
- dobivanje informacije o šteti za zadani automobil,
- dobivanje popisa svih automobila uključenih u nezgodu,
- provjeru je li automobil s danim brojem šasije bio sudjelovao u nezgodi.

Zatim je potrebno u sustav dodati klasu imena `CarHorizontal` koja čuva sva izvješća i omogućuje provjere automobila, odnosno potrebno je omogućiti

- dodavanje izvješća,
- dobivanje svih izvješća za zadani automobil,
- dohvaćanje ukupnog broja nezgoda za sve automobile gdje iz vraćenog rezultata mora biti moguće preko broja šasije pristupiti broju nezgoda za taj automobil.

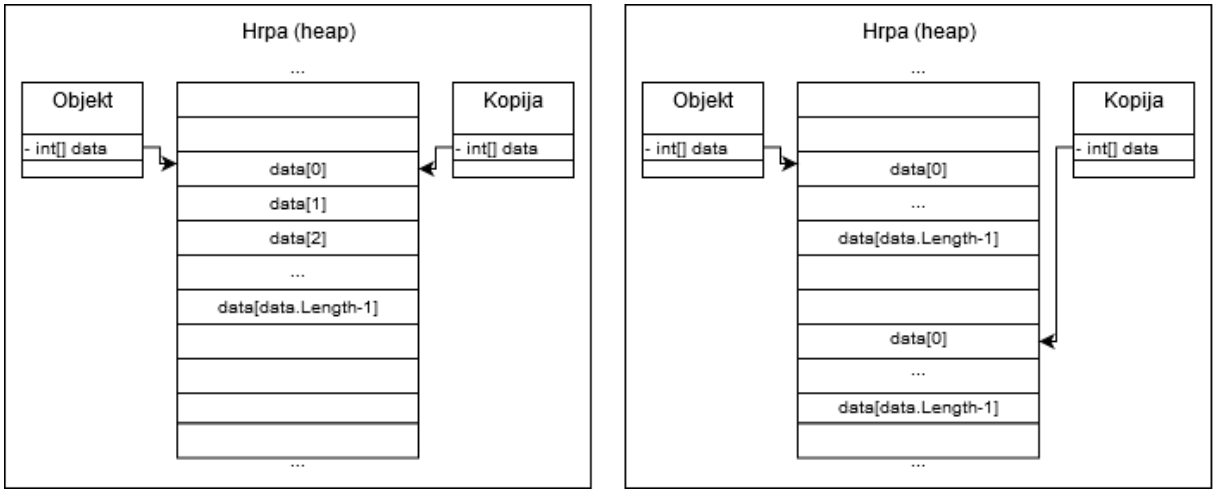
Za testiranje rješenja može poslužiti testni program dan izlistanjem. Proširite test tako da prikažete u koliko je nesreća sudjelovala *Porsche Panamera*.

```
1  public static void Test()
2  {
3      List<Car> cars = new List<Car>()
4      {
5          new Car("JT2BF28K0X0158222","Peugeot", "308", 178000, 2011),
6          new Car("1FALP5747PA186161","VolksWagen", "Passat", 129000, 2016),
7          new Car("1G1BL837XNW191401","Nissan", "Qashqai", 132000, 2014),
8          new Car("1XP5DB9X66D699186","VolksWagen", "Golf V", 280000, 2005),
9          new Car("2G1FC3D33E9253794","Porsche", "Panamera", 73000, 2013),
10     };
11
12     Console.WriteLine("Original:");
13     AccidentReport fenderbender = new AccidentReport();
14     fenderbender.AddCarInvolved("JT2BF28K0X0158222", 323.99m);
15     fenderbender.AddCarInvolved("1XP5DB9X66D699186", 463.22m);
16     fenderbender.AddCarInvolved("2G1FC3D33E9253794", 11333.00m);
17     Console.WriteLine(string.Join(",", fenderbender.GetAllInvolvedCars()));
18     Console.WriteLine(fenderbender.CalculateTotalDamage());
19     Console.WriteLine(fenderbender.GetDamageForCar("FakeVinNumber"));
20     Console.WriteLine(fenderbender.WasCarInvolved("2G1FC3D33E9253794"));
21
22     Console.WriteLine("Copy:");
23     AccidentReport fenderBenderDuplicate = new AccidentReport(fenderbender);
24     fenderbender.AddCarInvolved("1G1BL837XNW191401", 256.54m);
25     Console.WriteLine(string.Join(",", fenderbender.GetAllInvolvedCars()));
26     Console.WriteLine(string.Join(",", fenderBenderDuplicate.GetAllInvolvedCars()));
27
28     Console.WriteLine("Car horizontal checking:");
29     AccidentReport recklessDriving = new AccidentReport();
30     recklessDriving.AddCarInvolved("2G1FC3D33E9253794", 2312.00m);
31     CarHorizontal carHorizontalCroatia = new CarHorizontal();
32     /*
33      * TODO: Add both reports.
34      * Find total accidents for the Panamera using the CarHorizontal instance.
35      */
36 }
```

3 Dodatne upute

3.1 Plitko i duboko kopiranje

Kada se za vlastite klase piše konstruktor kopije, tada treba biti oprezan. Naime, ako klasa kao atribute sadrži samo ugrađene tipove podataka implementacija kopiranja vrlo je jednostavna. Potrebno je samo kopirati vrijednosti iz objekta predanog preko reference u objekt koji se stvara. Situacija se komplicira ako se konstruktor kopije piše za složeniju klasu, onu koja u sebi drži tip reference. U takvom će slučaju samo prepisivanje vrijednosti dovesti do toga da i originalni objekt i kopija u sebi sadrže jednake reference. Primjer ovoga moguće je vidjeti na slici 3.1, dok je primjer koda gdje je važna razlika između plitkog i dubokog kopiranja dan primjerom 3.1.



Slika 1: Potreba za dubokim kopiranjem

Primjer 3.1. Na primjeru pokažite razliku između plitkog i dubokog kopiranja.

```
1  public class Dog
2  {
3      public int Age { get; set; }
4      public string Name { get; set; }
5
6      public Dog(int age, string name)
7      {
8          this.Age = age;
9          this.Name = name;
10     }
11
12     public Dog(Dog other)
13     {
14     //Shallow copy, works fine:
15         this.Age = other.Age;
16         this.Name = other.Name;
17     }
18
19     public override string ToString() => $"{Name}: {Age}";
20 }
21
22 class Storage
23 {
24     private List<int> data;
25     public Storage() { data = new List<int>(); }
26     public Storage(Storage other)
27     {
28         //Shallow copy, bad in this case, try by uncommenting:
```

```
        //data = other.data;

        // Deep copy, proper way:
        data = new List<int>();
        data.AddRange(other.data);
    }

    public void Insert(int value) { data.Add(value); }
    public override string ToString() { return string.Join(",", data);
}

public void RunSimpleDemo()
{
    Dog rex = new Dog(1, "Rex");
    Dog rexCopy = new Dog(rex);
    Console.WriteLine($"{rex} and {rexCopy} ");
    rexCopy.Name = "RexCopy";
    Console.WriteLine($"{rex} and {rexCopy} ");

    Storage storage = new Storage();
    storage.Insert(10);
    Storage storageCopy = new Storage(storage);
    Console.WriteLine($"Original: {storage}{Environment.NewLine}Copy: {storageCopy}");
    storageCopy.Insert(1200);
    Console.WriteLine($"Original: {storage}{Environment.NewLine}Copy: {storageCopy}");
}
```