

Završni ispit iz predmeta Razvoj primijenjene programske potpore

22	4	6	0	1	2	3	4	5	6	7	8
				😊	😊	😊	😊	😞	😞	😊	😞
				14	15	16	17	18	19	20	21
				😊	😞	😊	😊	😊	😊	😞	
				26	27	28	29	30	31	32	
				😊	😞	😞	😊	😊	😊	😊	

Score: 13.150 (=65.8%)

Score: 0.625 (=100.0%)

Id: 43253

Koji nivo Richardsonovog modela zrelosti zadovoljavaju web-servisi koji koriste SOAP protocol?

a 1

b 0

c 3

d GET + POST

e 2

22 4 6 0

Score: 13.150 (=65.8%)

1	😊	2	😊	3	😊	4	😊	5	😞	6	😞	7	😊	8	😞
14	😊	15	😞	16	😊	17	😊	18	😊	19	😊	20	😐		
26	😊	27	😐	28	😐	29	😊	30	😊	31	😊	32	😊		

Score: 0.625 (=100.0%)

Id: 43207

Ako nekom svojstvu postavimo validacijski atribut `Remote`, npr.

```
public class MyModel {  
    [Remote(action:"CheckA", controller: "C", ErrorMessage = "Vrijednost nije ispravna")]  
    public int A { get; set; }  
}
```

tada u upravljaču C mora biti definiran postupak CheckA sljedećeg oblika

- a `public async Task<string> CheckA(int A) { ... }`
- b `public async Task<bool> CheckA(int A) { ... }`**
- c `public async Task<string> CheckA(MyModel model) { ... }`
- d `public async Task<ActionResult> CheckA(MyModel model) { ... }`
- e `public async Task<ActionResult> CheckA(int A) { ... }`

22 4 6 0

Score: 13.150 (=65.8%)

1	😊	2	😊	3	😊	4	😊	5	😞	6	😞	7	😊	8	😞	9	😊
14	😊	15	😞	16	😊	17	😊	18	😊	19	😊	20	😐	21	😊	22	😊
26	😊	27	😐	28	😐	29	😊	30	😊	31	😊	32	😊				

Score: 0.625 (=100.0%)

Id: 38421

Ako WebApi (REST) servis uspješno obriše podatak korisniku će se vratiti statusna poruka čiji je statusni kod

- a 410 GONE
- b 404 NOT FOUND
- c 205 RESET CONTENT
- d 204 NO CONTENT**
- e 301 MOVED PERMANENTLY



Score: 0.625 (=100.0%)

Id: 38420

Ako WebApi (REST) servis uspješno ažurira postojeći podatak korisniku će se vratiti statusna poruka čiji je statusni kod

a

201 CREATED

b

202 ACCEPTED

c

302 FOUND (REDIRECT)

d

204 NO CONTENT

e

205 RESET CONTENT

22

4

6

0

Score: 13.150 (=65.8%)

1



2



3



4



5



6



7



8



9



10



11



14



15



16



17



18



19



20



21



22



23



26



27



28



29



30



31



32



Score: -0.150 (=24.0%)

Id: 43214

Koju od navedenih kombinacija treba upotrijebiti u EF kontekstu da bi mogli izvršiti naredbu

```
var list = ctx.vw_Pribor.ToList();
```

ako je u EF modelu definiran razred ViewPribor koji svojim oblikom prati rezultat pogleda vw_Pribor iz baze podataka?

a

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
    modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
        entity.ToView("vw_Pribor");
    });
}
```

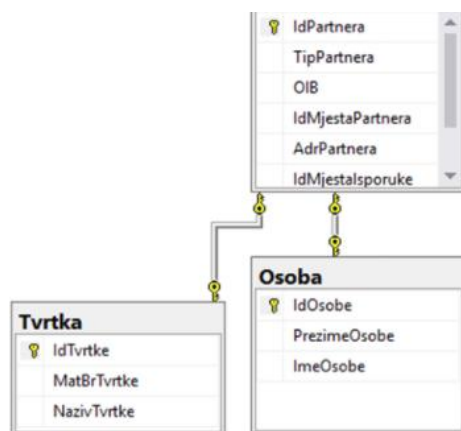
b

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
    modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
    });
}
```

c

```
public virtual DbSet<ViewPribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
    modelBuilder.Entity<ViewPribor>(entity => {

```

Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Za dodavanje novog partnera, koji može biti osoba ili tvrtka, koristi se prezentacijski model *PartnerViewModel*. Na koji od navedenih atributa treba dodati validacijski atribut `[Required]`?

```

public class PartnerViewModel {
    public int IdPartnera { get; set; }
    public string TipPartnera { get; set; }
    public string PrezimeOsobe { get; set; }
    public string ImeOsobe { get; set; }
    public string MatBrTvrtke { get; set; }
    public string NazivTvrtke { get; set; }
    public string Oib { get; set; }
    public string AdrPartnera { get; set; }
    public int? IdMjestaPartnera { get; set; }
    public string NazMjestaPartnera { get; set; }
}
  
```

a OIB

b MatBrTvrtke

c IdPartnera

d PrezimeOsobe

Score: 0.625 (=100.0%)

Id: 43212

Želimo li model iz EF modela, proširiti svojstvom koje nema svoj ekvivalent u tablici u bazi podataka, upotrijebit ćemo atribut?

a BindNever

b NotMapped

c Partial

d NotFromSql

e Computed

Score: -0.150 (= -24.0%)

Id: 38424

Ako je REST servis firma oblikovan prema standardu OData tada se artikl sa šifrom 8 nalazi na sljedećoj adresi

a `http://.../firma/artikl?filterByPK(8)`

b `http://.../firma/artikl?SifArtikla=8`

c `http://.../firma/artikl/8`

d `http://.../firma/artikl?filter=8`

e `http://.../firma/artikl(8)`

Score: 0.625 (=100.0%)

Id: 10020

REST servisi pozivaju se sljedećim HTTP zahtjevima

a GET, PUT, POST, DELETE

b LOAD, INSERT, UPDATE, DELETE

c GET, ADD, UPDATE, DELETE

d CREATE, READ, UPDATE, DELETE

e MOVE, SET, PEEK, POP

Score: 0.625 (=100.0%)

Id: 43247

Command Query Separation pomaže riješiti probleme

- ☐ a ORM-a
- ☒ b debelog klijenta
- ☐ c CRUD aplikacija
- ☐ d tankog klijenta
- ☐ e *constructor overinjectiona*

Score: 0.625 (=100.0%)

Id: 38436

Ako je REST servis firma oblikovan prema standardu OData tada se prva 3 artikla poredana po cijeni silazno mogu dohvatiti na adresi

a

`http://.../firma/artikl?$top=3&$orderby=CijArtikla desc`

b

`http://.../firma/3/artikl?$$by=CijArtiklaDesc`

c

`http://.../firma/artikl?$top=3&$filter=CijArtikla desc`

d

`http://.../firma/artikl(3)/By/CijArtkla`

e

`http://.../firma/artikl/3?orderby=CijArtkla&desc=true`

Id: 43215

Što ispisuje sljedeći programski odsječak?

```
foreach(string s in GetData()) {  
    Console.WriteLine(s);  
}  
...  
IEnumerable<string> GetData() {  
    Console.WriteLine("GetData");  
    yield return "A";  
    yield return "B";  
    yield return "C";  
}
```

a

GetData

b

GetData

A

GetData

B

GetData

C

c

GetData

A

d

GetData

A

B

C

Score: 0.625 (=100.0%)

Id: 43245

Razredi koji sadrže samo podatke te se koriste za prijenos podataka između različitih slojeva ili preko mreže nazivaju se

a Network Lying Objects

b Data Transfer Objects

c Network Transfer Objects

d Data Access Objects

e Data Access Layers

Score: 0.625 (=100.0%)

Id: 38432

Akcija GetImage na nekom upravljaču unutar MVC aplikacije treba vratiti sliku (byte[]) ili status 404 ako slika ne postoji. Ako se za dohvat slike koristi await i asinkrona metoda, tada postupak GetImage može biti sljedećeg oblika

a `public await Task<IActionResult> GetImage(int id) { ... }`

b `public async Task<IActionResult> GetImage(int id) { ... }`

c `public async Task<byte[]> GetImage(int id) { ... }`

d `public async IActionResult GetImage(int id) { ... }`

e `public async byte[] GetImage(int id) { ... }`

Score: -0.150 (=24.0%)

Id: 43216

Što ispisuje sljedeći programski odsječak?

```
var data = GetData();  
Console.WriteLine(data.First());  
...  
IEnumerable<string> GetData() {  
    Console.WriteLine(1);  
    yield return "A";  
    Console.WriteLine(2);  
    yield return "B";  
    Console.WriteLine(3);  
    yield return "C";  
}
```

a

A
1
2
3

b

1
2
3
C

c

1
2
3
A

Score: 0.625 (=100.0%)

Id: 38427

Proces kojim program može pregledavati i modificirati vlastitu strukturu tijekom izvođenja zove se

a perspektiva

b simetrija

c retrospektiva

d refleksija

e introspektiva

Score: 0.625 (=100.0%)

Id: 43218

U pogledu ili glavnoj stranici potrebno je definirati varijablu koja sadrži putanju do aplikacije tako da se ta vrijednost može koristiti u skriptnim datoteka.

Npr. neka se akcija *Index* i upravljač *Home* nalazi na adresi `https://rppp.fer.hr:4443/42/Apps/20200316.2/Home/Index`. Tada je putanja `/42/Apps/20200316.2/`.

Koja od sljedećih naredbi u varijablu `root` posprema putanju aplikacije u odnosu na server?

a

```
<script>
root = '@Application.Root';
</script>
```

b

```
<script>
root = '@System.IO.Path.Get("~")';
</script>
```

c

```
<script>
root = '@Url.Content("~/")';
</script>
```

d

```
<script>
root = 'asp-action=~';
</script>
```

e

```
<script>
root = '~';
</script>
```

Score: 0.625 (=100.0%)

Id: 43217

Što ispisuje sljedeći programski odsječak?

```
var data = GetData();  
Console.WriteLine(data.Last());  
...  
IEnumerable<string> GetData() {  
    Console.WriteLine(1);  
    yield return "A";  
    Console.WriteLine(2);  
    yield return "B";  
    Console.WriteLine(3);  
    yield return "C";  
}
```

a

3
C

b

3
2
1
C

c

1
2
3
C

Score: 0.625 (=100.0%)

Id: 38426

Koji od navedenih programskih odsječaka **ne omogućava** dobivanje informacija o tipu MojRazred

a

```
Type t = Type.GetType("NazivProjekta.MojRazred");
```

b

```
Type t = typeof(MojRazred)
```

c

```
Type t = MojRazred.Invoke(GetType());
```

d

```
Type t = Type.GetType("NazivProjekta.MojRazred, NazivAsemblija");
```

e

```
Type t = new MojRazred().GetType();
```

Score: 0.000 (=0.0%)

Id: 38428

Želimo li definirati vlastiti atribut koji se može primijeniti **samo na svojstva**, pa primjerice možemo imati sljedeći odsječak:

```
public class Test{  
    [Hide(When="Web")]  
    public int X { get; set; }  
}
```

napisat ćemo:

a

```
[AttributeUsage(AttributeTargets.Property)]  
public class HideAttribute : Attribute {  
    public string When { get; set; }  
}
```

b

```
[AttributeUsage(AttributeTargets.Property)]  
public class HideWhen : Attribute {  
    public string Web { get; set; }  
}
```

c

```
[AttributeUsage(AttributeTargets.Property)]  
public class Hide : Attribute {  
    public string When(string web, int x) { }  
}
```

d

```
public class Hide : PropertyAttribute {  
    public string When { get; set; }  
}
```


Score: 0.625 (=100.0%)

Id: 43256

Izbacite uljeza

a FromQuery

b FromRoute

c FromForm

d FromServices

e FromScript

Score: 0.000 (=0.0%)

Id: 38434

Ako je referenca *obj* tipa *object* referenca na objekt tipa *Robot* iz **nereferenciranog** asemblija pri čemu je *Robot* definiran na sljedeći način

```
public class Robot{  
    public Robot(name) { ...  
    public void Move(string direction, int moves) { ...  
}
```

robota možemo poslati 2 koraka na sjeverozapad sljedećim programskim odsječkom

a

```
type.GetMethod("Move").Invoke("NW", 2);
```

b

```
obj.Invoke(type.GetMethod("Move"), "NW", 2);
```

c

```
((Robot) obj).Move("NW", 2);
```

d

```
obj.Move("NW", 2);
```

e

```
type.GetMethod("Move").Invoke(obj, new object[] { "NW", 2});
```



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: ctx je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Osoba o = new Osoba();  
o.PrezimeOsobe = ...  
o.ImeOsobe = ...  
ctx.Add(o);  
ctx.SaveChanges();
```

- a** Izvršit će se dva INSERT upita, prvo INSERT u tablicu Osoba kako bi se dobio PK, koji će onda biti korišten prilikom INSERTa u tablicu Partner kao PK, pri čemu će ostale vrijednosti u tablici Partner biti null.
- b** Izvršavanjem neće doći do iznimke, ali se neće dogoditi niti jedan INSERT upit.
- c** Izvršit će se dva INSERT upita, prvo INSERT u tablicu Partner koristeći null za sve vrijednosti osim primarnog ključa koji će biti automatski generiran, a zatim INSERT u tablicu Osoba s vrijednosti PK koja odgovara generiranom PK prilikom unosa u tablicu Partner.
- d** Izvršit će se samo jedan INSERT upit i to u tablicu Osoba.
- e** Izvršavanje izaziva iznimku jer se osoba ne može dodati bez dodavanja postojećeg partnera.

Score: 0.625 (=100.0%)

Id: 43250

HATEOAS je skraćenica od

- a** Hypermedia Automated Extension of a Service
- b** Hypermedia Automated Engine of a Service
- c** Hypermedia and the reverse SOA
- d** Hypermedia as the Envelope of Application State
- e** Hypermedia as the Engine of Application State

Score: 0.000 (=0.0%)

Id: 43244

Ako su zadani sljedeća razredi i sučelja (u svrhu ostvarenja principa *Command Query Separation*)

```
public interface IQuery <TResult> { }
public interface IQueryHandler<TQuery, TResult> where TQuery : IQuery<TResult> {
    Task<TResult> Handle (TQuery query);
}

public class ContainsCount : IQuery<int> {
    public string[] Names { get; set; }
}

public interface IContainsCountQueryHandler : IQueryHandler<ContainsCount, int> { }
```

koja od sljedećih naredbi je ispravna?

Napomena: referenca `IContainsCountQueryHandler handler` je ispravno definira te joj je pridružen konkretni rukovatelj upitom.

a

```
int q = await handler.Handle(new ContainsCount { Names = new [] { "a", "b" } });
```

b

```
IQuery<int> q = await handler.Handle(new ContainsCount { Names = new [] { "a", "b" } });
```

c

```
int q = await handler.Handle(new [] { "a", "b" });
```

d

```
string[] q = await handler.Handle(33);
```

e

```
string[] q = await handler.Handle(new ContainsCount(33) );
```

Score: 0.625 (=100.0%)

Id: 43242

Ako je upit (po principu *Command Query Separation*) definiran na sljedeći način

```
public interface IQuery <TResult> { }  
public class GetLongNamesQuery : IQuery<List<string>> {  
    public int LongerThan { get; set; }  
}
```

a sučelje za rukovatelje upita definirano s

```
public interface IQueryHandler<TQuery, TResult> where TQuery : IQuery<TResult> {  
    Task<TResult> Handle (TQuery query);  
}
```

tada rukovatelja upita za dohvat imena dužih od određenog broja znakova možemo opisati sučeljem:

a `public interface IGetLongNamesQueryHandler : IQueryHandler<IQuery<List<string>>, List<string>> { }`

b `public interface IGetLongNamesQueryHandler : IQueryHandler<int, List<string>> { }`

c `public interface IGetLongNamesQueryHandler : IQueryHandler<GetLongNamesQuery, int> { }`

d `public interface IGetLongNamesQueryHandler : IQueryHandler<GetLongNamesQuery, List<string>> { }`

e `public interface IGetLongNamesQueryHandler : IQueryHandler<List<string>, int> { }`

Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: ctx je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Partner p = new Partner();  
p.AdrIsporuke = ...;  
p.AdrPartnera = ...;  
p.IdMjestaIsporuke = ...;  
p.IdMjestaPartnera = ...;  
p.Oib = ...;  
ctx.Add(p);  
ctx.SaveChanges();
```

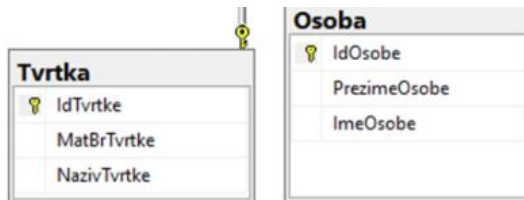
a Izvršit će se tri INSERT upita, po jedan za svaku od tablica, pri čemu će IdTvrtke i IdOsobe imati istu vrijednost 0, neovisno o vrijednosti koja je dobivena za IdPartnera.

b Izvršit će se samo jedan INSERT upit i to u tablicu Partner.

c Izvršit će se tri INSERT upita, po jedan za svaku od tablica, pri čemu će IdTvrtke i IdOsobe imati istu vrijednost koja je automatski dobivena za IdPartnera.

d Upit se ne može izvršiti (tj. izaziva iznimku prilikom izvršavanja), jer za Partnera nije postavljena osoba ili tvrtka.

e Izvršavanjem neće doći do iznimke, ali se neće dogoditi niti jedan INSERT upit.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: *ctx* je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. *IdPartnera* je tipa *identity* i sve trenutne vrijednosti u bazi podataka za *IdPartnera* su 1 ili više.

```

Partner p = ctx.Find(...)
p.OIB = ...
p.Osoba = new Osoba { IdOsobe = p.IdPartnera, ImeOsobe = ..., PrezimeOsobe = ... };
ctx.Entry(partner.Osoba).State = EntityState.Modified;
ctx.SaveChanges();
  
```

- a** Izvršit će se INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner na način da se promijeni OIB. Iznimka se neće dogoditi ako već nije postojala osoba s istom vrijednosti primarnog ključa.
- b** Izvršit će se 2 UPDATE upita: jedan za promjenu Osobe koja ima IdOsobe jednak IdPartnera partnera kojem mijenjamo OIB.
- c** Pokušat će se izvršiti INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner, ali će se dogoditi iznimka jer se za IdOsobe koristi vrijednost postojećeg partnera.
- d** Izvršit će se samo jedan UPDATE upit te će se promijeniti samo OIB dohvaćenog partnera.
- e** Izvršit će se jedan UPDATE upit i jedan DELETE upit. UPDATE mijenja OIB partnera, a DELETE briše osoba koja je prethodno imala IdOsobe koji je odgovarao IdPartnera. Veza Osobe i Partnera je uklonjena zbog new p.Osoba = , a ta osoba nije dodana u kontekst.

Score: 0.625 (=100.0%)

Id: 38433

Ako objekt tipa *Type* sadrži informaciju o razredu *Robot* iz **nereferenciranog** asemblija pri čemu je *Robot* definiran na sljedeći način

```
public class Robot{  
    public Robot(name) { ...  
    public void Move(string direction, int moves) { ...  
}
```

novi robot pod nazivom R1P3 možemo dinamički instancirati na sljedeći način:

a

```
object robot = Activator.CreateInstance("Robot", "R1P3");
```

b

```
Robot robot = Activator.CreateInstance(Robot::new("R1P3"));
```

c

```
Robot robot = Activator.CreateInstance(type, "R1P3");
```

d

```
object robot = Activator.CreateInstance(type, "R1P3");
```

e

```
Robot robot = new Roboto("R1P3");
```

Score: 0.625 (=100.0%)

Id: 20176

Koja od navedenih formi omogućava prihvrat datoteke?

a

```
<form asp-action="Edit" post="multipart/form-data">
```

b

```
<form asp-action="Create" method="post">
```

c

```
<form asp-action="Create" method="get" enctype="multipart/form-data">
```

d

```
<form asp-action="Create" method="multipart/post">
```

e

```
<form asp-action="Edit" method="post" enctype="multipart/form-data">
```



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: ctx je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```

Partner p = ctx.Find(...)
p.OIB = ...
p.Osoba = new Osoba { ImeOsobe = ..., PrezimeOsobe = ... };
ctx.SaveChanges();
  
```

- a** Izvršit će se samo jedan UPDATE upit te će se promijeniti samo OIB dohvaćenog partnera.
- b** Izvršit će se jedan UPDATE upit i jedan DELETE upit. UPDATE mijenja OIB partnera, a DELETE briše osoba koja je prethodno imala IdOsobe koji je odgovarao IdPartnera. Veza Osobe i Partnera je uklonjena zbog new p.Osoba = , a ta osoba nije dodana u kontekst.
- c** Izvršit će se 2 UPDATE upita: jedan za promjenu Osobe koja ima IdOsobe jednak IdPartnera partnera kojem mijenjamo OIB.
- d** Izvršit će se INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner na način da se promijeni OIB i IdPartnera u skladu s dobivenom vrijednosti primarnog ključa u tablici Osoba.
- e** Pokušat će se izvršiti INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner, ali će se dogoditi iznimka jer se za IdOsobe koristi vrijednost 0.

Ako je akcija na upravljaču definirana na sljedeći način

```
public IActionResult Test(Stavka[] stavke) { ... }
```

tada unutar forme moraju biti elementi nalik:

a

```
<input type="hidden" name="stavke.Index" value="knjiga" />
<input type="text" name="stavke[knjiga].IdStavke" ... />

<input type="hidden" name="stavke.Index" value="5" />
<input type="text" name="stavke[5].IdStavke" ... />
```

b

```
<input type="text" name="stavke.IdStavke[0]" ... />
<input type="text" name="stavke.IdStavke[1]" ... />
<input type="text" name="stavke.IdStavke[2]" ... />
```

c

```
<input type="hidden" name="stavke.Index" value="0-2" />
<input type="text" name="stavke[Index].IdStavke" ... />
<input type="text" name="stavke[Index].IdStavke" ... />
<input type="text" name="stavke[Index].IdStavke" ... />
```

d

```
<input type="text" name="stavke.IdStavke" ... />
<input type="text" name="stavke.IdStavke" ... />
<input type="text" name="stavke.IdStavke" ... />
```

e

```
<input type="text" name="stavke[0].IdStavke" ... />
<input type="text" name="stavke[2].IdStavke" ... />
<input type="text" name="stavke[4].IdStavke" ... />
```


Što ispisuje sljedeći programski odsječak?

```
foreach(string s in GetData()) {  
    Console.WriteLine(s);  
}  
...  
IEnumerable<string> GetData() {  
    Console.WriteLine("GetData");  
    yield return "A";  
    yield return "B";  
    yield return "C";  
}
```

a

GetData
A

b

GetData
A
GetData
B
GetData
C

c

GetData
A
GetData
A
GetData
A
... beskonačno ponavljanje...

d

GetData
A
B
C

Ako je referenca *obj* tipa *object* referenca na objekt tipa *Robot* iz **nereferenciranog** asemblija pri čemu je *Robot* definiran na sljedeći način

```
public class Robot{  
    public Robot(name) { ...  
    public void Move(string direction, int moves) { ...  
}
```

robota možemo poslati 2 koraka na sjeverozapad sljedećim programskim odsječkom

- a `type.GetMethod("Move").Invoke("NW", 2);`
- b `type.GetMethod("Move").Invoke(obj, new object[] { "NW", 2});`
- c `((Robot) obj).Move("NW", 2);`
- d `obj.Move("NW", 2);`
- e `obj.Invoke(type.GetMethod("Move"), "NW", 2);`

Koju od navedenih kombinacija treba upotrijebiti u EF kontekstu da bi mogli izvršiti naredbu

```
var list = ctx.ViewPribor.ToList();
```

ako je u EF modelu definiran razred ViewPribor koji svojim oblikom prati rezultat pogleda vw_Pribor iz baze podataka?

a

```
public virtual DbSet<ViewPribor> ViewPribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<ViewPribor>(entity => {  
        entity.HasNoKey();  
        entity.ToView("vw_Pribor");  
    });  
};
```

b

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<vw_Pribor>(entity => {  
        entity.HasNoKey();  
    });  
};
```

c

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<ViewPribor>(entity => {  
        entity.HasNoKey();  
        entity.ToView("vw_Pribor");  
    });  
};
```

d

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<ViewPribor>(entity => {  
        entity.HasNoKey();  
        entity.ToView("vw_Pribor");  
    });  
};
```

- a Mehanizam za povezivanje udaljenih postupaka.
- b XML shemu za opis web servisa.**
- c Atribut kojim se označava postupak web servisa.
- d Protokol za razmjenu informacija u distribuiranim, heterogenim okruženjima
- e Postupak lociranja web servisa.

Ako je REST servis firma oblikovan prema standardu OData tada se artikl sa šifrom 8 nalazi na sljedećoj adresi:

a `http://.../firma/artikl(8)`

b `http://.../firma/artikl?filter=8`

c `http://.../firma/artikl/8`

d `http://.../firma/artikl?SifArtikla=8`

e `http://.../firma/artikl?filterByPK(8)`

Koju od navedenih kombinacija treba upotrijebiti u EF kontekstu da bi mogli izvršiti naredbu

```
var list = ctx.vw_Pribor.ToList();
```

ako je u EF modelu definiran razred ViewPribor koji svojim oblikom prati rezultat pogleda vw_P

a

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<ViewPribor>(entity => {  
        entity.HasNoKey();  
        entity.ToView("vw_Pribor");  
    });
```

b

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<ViewPribor>(entity => {  
        entity.HasNoKey();  
    });
```

c

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<ViewPribor>(entity => {  
        entity.HasNoKey();  
        entity.ToView("ViewPribor");  
    });
```

d

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }  
...  
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {  
    modelBuilder.Entity<vw_Pribor>(entity => {
```

Ako su zadani sljedeća razredi i sučelja (u svrhu ostvarenja principa *Command Query Separation*)

```
public interface IQuery<TResult> { }  
public interface IQueryHandler<TQuery, TResult> where TQuery : IQuery<TResult> {  
    Task<TResult> Handle (TQuery query);  
}  
  
public class ContainsCount : IQuery<int> {  
    public string[] Names { get; set; }  
}  
  
public interface IContainsCountQueryHandler : IQueryHandler<ContainsCount, int> { }
```

koja od sljedećih naredbi je ispravna?

Napomena: referenca `IContainsCountQueryHandler handler` je ispravno definira te joj je pridružen konkretni rukovatelj upitom.

a

```
int q = await handler.Handle(new [] { "a", "b" });
```

b

```
string[] q = await handler.Handle(new ContainsCount(33) );
```

c

```
string[] q = await handler.Handle(33);
```

d

```
IQuery<int> q = await handler.Handle(new ContainsCount { Names = new [] { "a", "b" } });
```

e

```
int q = await handler.Handle(new ContainsCount { Names = new [] { "a", "b" } });
```


Koji nivo Richardsonovog modela zrelosti zadovoljavaju web-servisi koji koriste SOAP protocol?

a 0

b 1

c 3

d 2

e GET + POST

Želimo li definirati vlastiti atribut koji se može primijeniti **samo na svojstva**, pa primjerice možemo imati sljedeći odsječak:

```
public class Test{  
    [Hide(When="Web")]  
    public int X { get; set; }  
}
```

napisat ćemo:

a

```
[AttributeUsage(AttributeTargets.Property)]  
public class Hide : Attribute {  
    public string When(string web, int x) { }  
}
```

b

```
public class Hide : PropertyAttribute {  
    public string When { get; set; }  
}
```

c

```
[AttributeUsage(AttributeTargets.Property)]  
public class HideWhen : Attribute {  
    public string Web { get; set; }  
}
```

d

```
[AttributeUsage(AttributeTargets.Property)]  
public class HideWhenAttribute : Attribute {  
    public int X { get; set; }  
}
```

e

```
[AttributeUsage(AttributeTargets.Property)]  
public class HideAttribute : Attribute {  
    public string When { get; set; }  
}
```

Koja od navedenih formi omogućava prihvatanje datoteke?

a

```
<form asp-action="Create" method="multipart/post">
```

b

```
<form asp-action="Create" method="get" enctype="multipart/form-data">
```

c

```
<form asp-action="Edit" method="post" enctype="multipart/form-data">
```

d

```
<form asp-action="Create" method="post">
```

e

```
<form asp-action="Edit" post="multipart/form-data">
```

Ako je REST servis firma oblikovan prema standardu OData tada se prva 3 artikla poredana po cijeni silazno mogu dohvatiti na adresi

a `http://.../firma/artikl?$top=3&$filter=CijArtikla desc`

b `http://.../firma/artikl/3?orderby=CijArtikla&desc=true`

c `http://.../firma/artikl(3)/By/CijArtikla`

d `http://.../firma/artikl?$top=3&$orderby=CijArtikla desc`

e `http://.../firma/3/artikl?$$by=CijArtiklaDesc`

Ako je upit (po principu *Command Query Separation*) definiran na sljedeći način

```
public interface IQuery <TResult> { }  
public class GetLongNamesQuery : IQuery<List<string>> {  
    public int LongerThan { get; set; }  
}
```

a sučelje za rukovatelje upita definirano s

```
public interface IQueryHandler<TQuery, TResult> where TQuery : IQuery<TResult> {  
    Task<TResult> Handle (TQuery query);  
}
```

tada rukovatelja upita za dohvat imena dužih od određenog broja znakova možemo opisati sučeljem:

a `public interface IGetLongNamesQueryHandler : IQueryHandler<int, List<string>> { }`

b `public interface IGetLongNamesQueryHandler : IQueryHandler<List<string>, int> { }`

c `public interface IGetLongNamesQueryHandler : IQueryHandler<IQuery<List<string>>, List<string>> { }`

d `public interface IGetLongNamesQueryHandler : IQueryHandler<GetLongNamesQuery, List<string>> { }`

e `public interface IGetLongNamesQueryHandler : IQueryHandler<GetLongNamesQuery, int> { }`

Koji od navedenih programskih odsječaka **ne omogućava** dobivanje informacija o tipu MojRazred

a

```
Type t = MojRazred.Invoke(GetType());
```

b

```
Type t = typeof(MojRazred)
```

c

```
Type t = new MojRazred().GetType();
```

d

```
Type t = Type.GetType("NazivProjekta.MojRazred");
```

e

```
Type t = Type.GetType("NazivProjekta.MojRazred, NazivAsenblija");
```

Želimo li da WebApi servis automatski izvrši provjeru primljenog modela te vrati status 400 u slučaju validacijskih pogrešaka, upravljaču ćemo dodati atribut

a BadRequestFilter

b ControllerBase

c ExceptionFilter

d ApiController

e TypeFilter

Želimo li model iz EF modela, proširiti svojstvom koje nema svoj ekvivalent u tablici u bazi podataka, upotrijebit ćemo atribut?

a Partial

b Computed

c NotFromSql

d BindNever

e NotMapped

Ako je razred *Result* zadan s

```
public class Result {  
    [JsonPropertyName("action")]  
    public string ActionName { get; set; }  
    [JsonPropertyName("status")]  
    public bool ResultStatus { get; set; }  
}
```

a u *Startup.cs* datoteci se nalazi

```
services...  
    .AddJsonOptions(configure => configure.JsonSerializerOptions.PropertyNamingPolicy = JsonNamingPolicy.CamelCase);
```

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

a

```
{"action": "Run", "status": true}
```

b

```
{"actionName": "Run", "resultStatus": true}
```

c

```
{"result@7eal246f" : {"ActionName": "Run", "ResultStatus": true}}
```

d

```
{"Action": "Run", "Status": true}
```

e

```
{"ActionName": "Run", "ResultStatus": true}
```

Označite načine kako se u bazi podataka mogu implementirati specijalizacija i generalizacija.

Napomena: Pitanje ima više tačnih odgovora!

a table per primary key

b table per type

c table per hierarchy

d table per foreign key

e table per concrete class

Ako je akcija na upravljaču definirana na sljedeći način

```
public IActionResult Test(Stavka[] stavke) { ... }
```

tada unutar forme moraju biti elementi nalik:

a

```
<input type="text" name="stavke.IdStavke" ... />  
<input type="text" name="stavke.IdStavke" ... />  
<input type="text" name="stavke.IdStavke" ... />
```

b

```
<input type="hidden" name="stavke.Index" value="0-2" />  
<input type="text" name="stavke[Index].IdStavke" ... />  
<input type="text" name="stavke[Index].IdStavke" ... />  
<input type="text" name="stavke[Index].IdStavke" ... />
```

c

```
<input type="text" name="stavke.IdStavke[0]" ... />  
<input type="text" name="stavke.IdStavke[1]" ... />  
<input type="text" name="stavke.IdStavke[2]" ... />
```

d

```
<input type="text" name="stavke[0].IdStavke" ... />  
<input type="text" name="stavke[2].IdStavke" ... />  
<input type="text" name="stavke[4].IdStavke" ... />
```

e

```
<input type="hidden" name="stavke.Index" value="knjiga" />  
<input type="text" name="stavke[knjiga].IdStavke" ... />  
  
<input type="hidden" name="stavke.Index" value="5" />  
<input type="text" name="stavke[5].IdStavke" ... />
```

Novi/drugi naziv za Swagger je

a

OpenAPI

b

RESTful

c

JSON

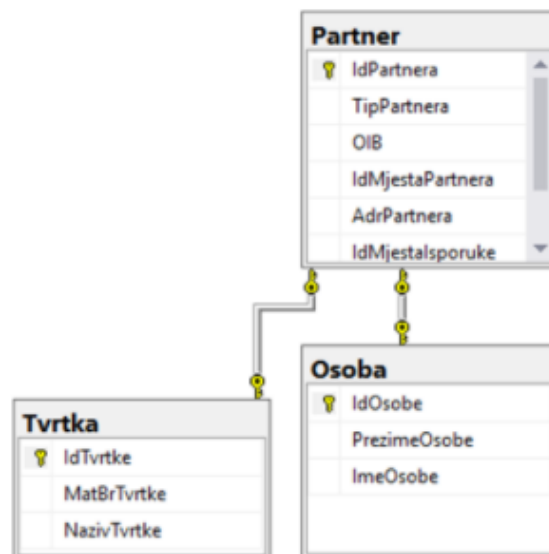
d

OData

e

REST API

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: ctx je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Partner p = new Partner();
p.AdrIsporuke = ...;
p.AdrPartnera = ...;
p.IdMjestaIsporuke = ...;
p.IdMjestaPartnera = ...;
p.Oib = ...;
ctx.Add(p);
ctx.SaveChanges();
```

- a** Izvršit će se samo jedan INSERT upit i to u tablicu Partner.
- b** Izvršavanjem neće doći do iznimke, ali se neće dogoditi niti jedan INSERT upit.
- c** Upit se ne može izvršiti (tj. izaziva iznimku prilikom izvršavanja), jer za Partnera nije postavljena osoba ili tvrtka.

Ako je u WebAPI upravljaču definirano sljedeće:

```
public class MjestoController : ControllerBase {  
    ...  
    [HttpPost]  
    public IActionResult Create (string s, int i) {  
        ...  
    }  
}
```

argumenti će se

a rekonstruirati iz parova ključ=vrijednost u FormData

b rekonstruirati iz tijela zahtjeva

c rekonstruirati iz query stringa

d rekonstruirati iz route

e dobiti korištenjem *dependency injectiona*

Ako nekom svojstvu postavimo validacijski atribut `Remote`, npr.

```
public class MyModel {  
    [Remote(action:"CheckA", controller: "C", ErrorMessage = "Vrijednost nije ispravna")]  
    public int A { get; set; }  
}
```

tada u upravljaču C mora biti definiran postupak CheckA sljedećeg oblika

- a** `public async Task<bool> CheckA(int A) { ... }`
- b** `public async Task<string> CheckA(MyModel model) { ... }`
- c** `public async Task<ActionResult> CheckA(MyModel model) { ... }`
- d** `public async Task<ActionResult> CheckA(int A) { ... }`
- e** `public async Task<string> CheckA(int A) { ... }`

Što ispisiš sljedeći programski odsječak?

```
var data = GetData();  
Console.WriteLine(data.Last());  
...  
IEnumerable<string> GetData() {  
    Console.WriteLine(1);  
    yield return "A";  
    Console.WriteLine(2);  
    yield return "B";  
    Console.WriteLine(3);  
    yield return "C";  
}
```

a

C
1
2
3

b

1
2
3

c

3
2
1
C

d

3
C

e

1
2
3
C

Command Query Separation pomaže riješiti probleme

a

ORM-a

b

CRUD aplikacija

c

tankog klijenta

d

debelog klijenta

e

constructor overinjectiona

Ako je razred *Result* zadan s

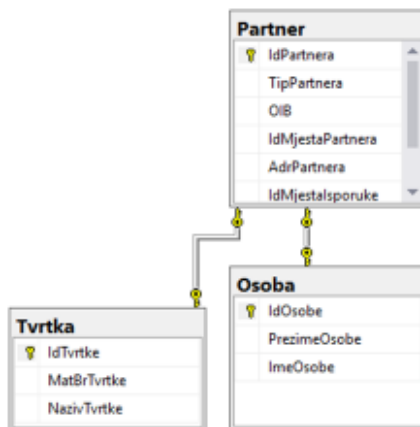
```
public class Result {  
    public string ActionName { get; set; }  
    public bool ResultStatus { get; set; }  
}
```

a u *Startup.cs* nije eksplicitno ništa navedeno oko serijalizacijskih opcija

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

- a {"ActionName":"Run", "ResultStatus" : true}
- b {"Name":"Run", "Status" : true}
- c {"result@7ea1246f" : {"ActionName":"Run", "ResultStatus" : true}}
- d {"actionName":"Run", "resultStatus" : true}**
- e {"name":"Run", "status" : true}

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Za dodavanje novog partnera, koji može biti osoba ili tvrtka, koristi se prezentacijski model `PartnerViewModel`. Na koji od navedenih atributa treba dodati validacijski atribut `[Required]`?

```
public class PartnerViewModel {
    public int IdPartnera { get; set; }
    public string TipPartnera { get; set; }
    public string PrezimeOsobe { get; set; }
    public string ImeOsobe { get; set; }
    public string MatBrTvrtke { get; set; }
    public string NazivTvrtke { get; set; }
    public string Oib { get; set; }
    public string AdrPartnera { get; set; }
    public int? IdMjestaPartnera { get; set; }
    public string NazMjestaPartnera { get; set; }
}
```

a `PrezimeOsobe`

b `IdPartnera`

c `MatBrTvrtke`

d `ImeOsobe`

e `OIB`

Ako forma izgleda nalik sljedećoj

```
<form action="Edit" method="post">  
  <input name="X" value="A"/>  
  <input name="X" value="B"/>  
  <input name="X" value="C"/>  
</form>
```

tada, za prihvatanje vrijednosti, akcija na upravljaču treba biti definirana na sljedeći način

a `public IActionResult Edit(X[] name) { ... }`

b `public IActionResult Edit(string[] X) { ... }`

c `public IActionResult Edit(string A, string B, string C) { ... }`

d `public IActionResult Edit([From(Name="X")] string[] values) { ... }`

e `public IActionResult Edit(X A, X B, X C) { ... }`

Ako WebApi (REST) servis uspješno doda novi podatak korisniku će se vratiti statusna poruka čiji je statusni kod

a

201 CREATED

b

302 FOUND (REDIRECT)

c

200 OK

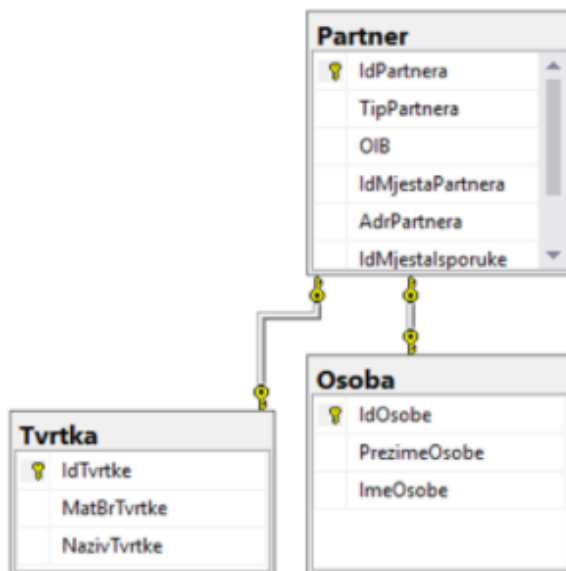
d

304 NOT MODIFIED

e

204 NO CONTENT

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: ctx je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Partner p = ctx.Find(...  
p.OIB = ...  
p.Osoba = new Osoba { ImeOsobe = ..., PrezimeOsobe = ... };  
ctx.SaveChanges();
```

- a** Izvršit će se jedan UPDATE upit i jedan DELETE upit. UPDATE mijenja OIB partnera, a DELETE briše osoba koja je prethodno imala IdOsobe koji je odgovarao IdPartnera. Veza Osobe i Partnera je uklonjena zbog new p.Osoba = , a ta osoba nije dodana u kontekst.
- b** Pokušat će se izvršiti INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner, ali će se dogoditi iznimka jer se za IdOsobe koristi vrijednost 0.
- c** Izvršit će se INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner na način da se promijeni OIB i IdPartnera u skladu s dobivenom vrijednosti primarnog ključa u tablici Osoba.
- d** Izvršit će se samo jedan UPDATE upit te će se promijeniti samo OIB dohvaćenog partnera.

Proces kojim program može pregledavati i modificirati vlastitu strukturu tijekom izvođenja zove se

a perspektiva

b refleksija

c introspektiva

d simetrija

e retrospektiva

REST servisi pozivaju se sljedećim HTTP zahtjevima

a

CREATE, READ, UPDATE, DELETE

b

GET, PUT, POST, DELETE

c

GET, ADD, UPDATE, DELETE

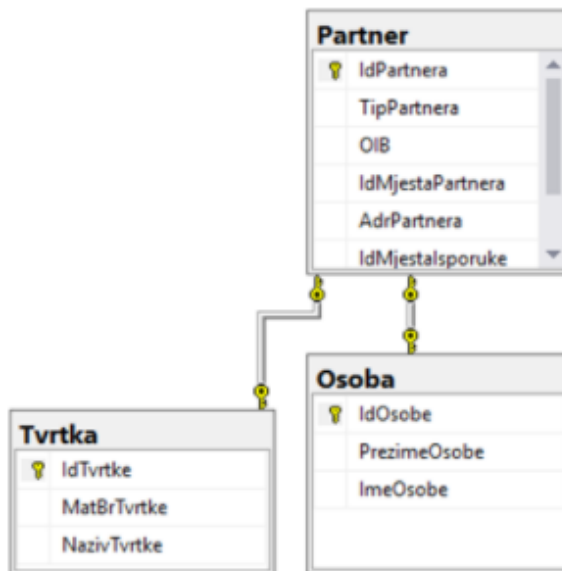
d

LOAD, INSERT, UPDATE, DELETE

e

MOVE, SET, PEEK, POP

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: ctx je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Partner p = ctx.Find(...
p.OIB = ...
p.Osoba = new Osoba { IdOsobe = p.IdPartnera, ImeOsobe = ..., PrezimeOsobe = ... };
ctx.Entry(partner.Osoba).State = EntityState.Modified;
ctx.SaveChanges();
```

a

Izvršit će se samo jedan UPDATE upit te će se promijeniti samo OIB dohvaćenog partnera.

b

Izvršit će se INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner na način da se promijeni OIB. Iznimka se neće dogoditi ako već nije postojala osoba s istom vrijednosti primarnog ključa.

c

Izvršit će se 2 UPDATE upita: jedan za promjenu Osobe koja ima IdOsobe jednak IdPartnera partnera kojem mijenjamo OIB.

d

Pokušat će se izvršiti INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner, ali će se dogoditi iznimka jer se za IdOsobe koristi vrijednost postojećeg partnera.