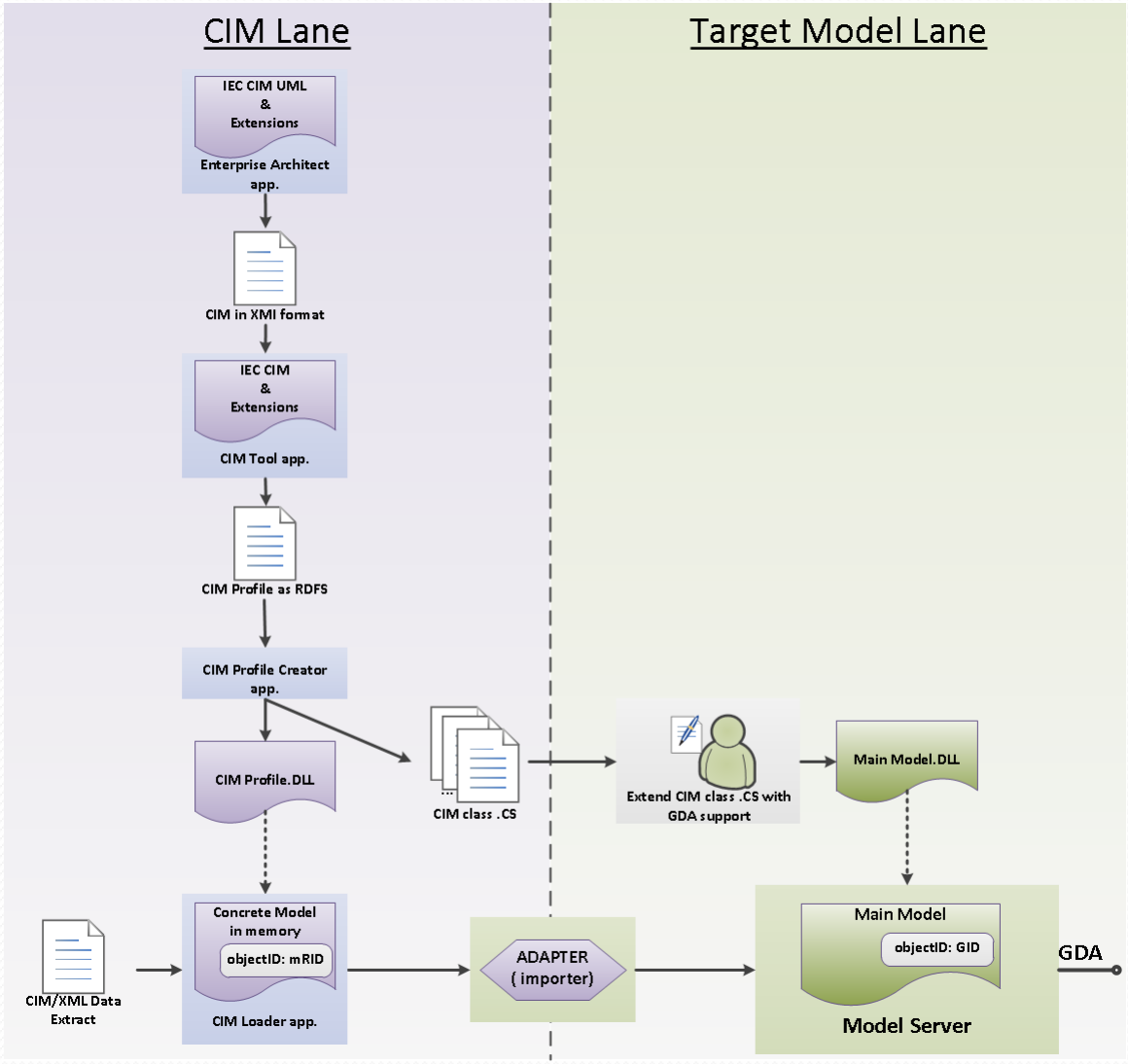
Standardi i modeliranje elektroenergetskih sistema

VEŽBA 12:

Dodavanje nove klase (sa dodatnim referencama) u Network Model Service

Tok podataka pri inicijalizaciji modela elektroenergetske mreže



Struktura DataModel klasa 1/2

- Atribut klase koji predstavlja referencu je tipa *long*, a njegova vrednost je globalni identifikator entiteta koga referencira

- Atribut klase koji predstavlja listu referenci (*target*) je tipa *Lista<long>* i njegova vrednost je lista globalnih identifikatora entiteta koji imaju referencu na pomenutu klasu

- Klase koje sadrže reference (jednu referencu ili listu referenci) implementiraju dodatne metode koje će omogućiti servisu manipulaciju nad referencama

```
public class TransformerWinding : ConductingEquipment
{
    private WindingConnection connectionType;

    private WindingType windingType;

    private bool grounded;

    private float ratedS;

    private float ratedU;

    private float phaseToGroundVoltage;

    private float phaseToPhaseVoltage;

    private long powerTransformer = 0;

    private List<long> windingTests = new List<long>();

    public TransformerWinding(long globalId)
        : base(globalId)
    {
    }
```

Struktura DataModel klasa 2/2

- *Equals()* metoda mora da proverava jednakost i ovih atributa (atributi tipa referenca i lista referenci (*target*))

```
public override bool Equals(object obj)
{
    if (base.Equals(obj))
    {
        TransformerWinding x = (TransformerWinding)obj;
        return (x.windingType == this.windingType && x.grounded == this.grounded && x.connectionType == this.connectionType &&
                x.ratedS == this.ratedS && x.ratedU == this.ratedU && x.phaseToGroundVoltage == this.phaseToGroundVoltage &&
                x.phaseToPhaseVoltage == this.phaseToPhaseVoltage && x.powerTransformer == this.powerTransformer &&
                CompareHelper.CompareLists(x.windingTests, this.windingTests));
    }
    else
    {
        return false;
    }
}
```

Metode za pristup podacima 1/4

- U *HasProperty()* metodi potrebno je navesti *ModelCode*-ove dodeljene ovim atributima.

```
public override bool HasProperty(ModelCode t)
{
    switch (t)
    {
        case ModelCode.POWERTRWINDING_CONNTYPE:
        case ModelCode.POWERTRWINDING_GROUNDED:
        case ModelCode.POWERTRWINDING_RATEDS:
        case ModelCode.POWERTRWINDING_RATEDU:
        case ModelCode.POWERTRWINDING_WINDTYPE:
        case ModelCode.POWERTRWINDING_PHASETOGRNDVOLTAGE:
        case ModelCode.POWERTRWINDING_PHASETOPHASEVOLTAGE:
        case ModelCode.POWERTRWINDING_POWERTRW:
        case ModelCode.POWERTRWINDING_TESTS:
            return true;

        default:
            return base.HasProperty(t);

    }
}
```

Metode za pristup podacima 2/4

- *GetProperty()* metoda omogućava čitanje ovih atributa.

```
public override void GetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.POWERTRWINDING_CONNTYPE:
            property.SetValue((short)connectionType);
            break;

            ...

        case ModelCode.POWERTRWINDING_PHASETOPHASEVOLTAGE:
            property.SetValue(phaseToPhaseVoltage);
            break;

        case ModelCode.POWERTRWINDING_POWERTRW:
            property.SetValue(powerTransformer);
            break;

        case ModelCode.POWERTRWINDING_TESTS:
            property.SetValue(windingTests);
            break;

        default:
            base.GetProperty(property);
            break;
    }
}
```

Metode za pristup podacima 3/4

- *SetProperty()* metoda omogućava upis samo atributa tipa referenca. Vrednost atribut tipa lista referenci (*target*) kreira sam servis i njegov upis se NE SME dozvoliti kroz *SetProperty()* metodu

```
public override void SetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.POWERTRWINDING_CONNTYPE:
            connectionType = (WindingConnection)property.AsEnum();
            break;

            ...

        case ModelCode.POWERTRWINDING_POWERTRW:
            powerTransformer = property.AsReference();
            break;

        default:
            base.SetProperty(property);
            break;
    }
}
```

Metode za pristup podacima 4/4

- Pošto vrednost atribut tipa lista referenci ne može da se postavi od strane klijenta, već je određuje servis, potrebno je navesti njegov *ModelCode* u listu *notSettable* atributa u klasi *ModelResourceDescs*

```
private void InitializeNotSettablePropertyIds()
{
    notSettablePropertyIds.Add(ModelCode.IDOBJ_GID);
    notSettablePropertyIds.Add(ModelCode.BASEVOLTAGE_CONDEQS);
    notSettablePropertyIds.Add(ModelCode.LOCATION_PSRS);
    notSettablePropertyIds.Add(ModelCode.POWERTRWINDING_TESTS);
}
```

Metode za rad sa referencama 1/4

- *IsReferenced* implementiraju samo klase koje imaju listu referenci (*target*-e) kao neki od atributa. Metoda vraća indikaciju da li postoji neki entitet koji referencira instancu ove klase (referencira neki od *target*-a). Metoda vraća *true* ukoliko bar jedan entitet refrencira instancu ove klase (proverava se da li neka lista referenci koje sadrži klasa ima bar jednu vrednost – globalni identifikator).

```
public override bool IsReferenced
{
    get
    {
        return windingTests.Count!= 0 || base.IsReferenced;
    }
}
```

- Metodu koristi servis kada dobije zahtev za brisanjem odgovarajućeg entiteta. Ukoliko je entitet referenciran od strane nekog drugog entiteta, zahtev se odbija. Smatra se da je entitet refrenciran nekim drugim entitetom ukoliko neki od njegovih *target*-a ima bar jedan globalni identifikator u listi.

Metode za rad sa referencama 2/4

- *GetReferences()* metoda se implementira za klasu koja ima atribute tipa referenca i/ili lista referenci (*target*). Kreira se mapa koja vraća vrednosti globalnih identifikatora koji su vrednosti atributa tipe referenca ili lista referenci (*target*-a).

```
public override void GetReferences(Dictionary<ModelCode, List<long>> references, TypeOfReference refType)
{
    if (powerTransformer != 0 && (refType == TypeOfReference.Reference || refType == TypeOfReference.Both))
    {
        references[ModelCode.POWERTRWINDING_POWERTRW] = new List<long>();
        references[ModelCode.POWERTRWINDING_POWERTRW].Add(powerTransformer);
    }

    if (windingTests != null && windingTests.Count != 0 && (refType == TypeOfReference.Target || refType == TypeOfReference.Both))
    {
        references[ModelCode.POWERTRWINDING_TESTS] = windingTests.GetRange(0, windingTests.Count);
    }

    base.GetReferences(references, refType);
}
```

Metode za rad sa referencama 3/4

- *AddReference()* metodu implementiraju samo klase koje imaju listu referenci (*target*) kao neki od atributa. Metodu koristi servis kako bi dodao odgovarajuću vrednost u atribut tipa lista referenci (*target*) .
  NAPOMENA: Treba primetiti da se u *switch* naredbi ne koristi *ModelCode* koji je dodeljen atributu tipa lista referenci (target) već *ModelCode* atributa koji pripada klasi koja referencira pomenuti target.

```
public override void AddReference(ModelCode referenceId, long globalId)
{
    switch (referenceId)
    {
        case ModelCode.WINDINGTEST_POWERTRWINDING:
            windingTests.Add(globalId);
            break;

        default:
            base.AddReference(referenceId, globalId);
            break;
    }
}
```

Metode za rad sa referencama 4/4

- *RemoveReference()* metodu implementiraju samo klase koje imaju listu referenci (*target*) kao neki od atributa. Metodu koristi servis kako bi uklonio odgovarajuću vrednost iz atributa tipa lista referenci (*target*) .
  NAPOMENA: Treba primetiti da se u *switch* naredbi ne koristi *ModelCode* koji je dodeljen atributu tipa lista referenci (target) već *ModelCode* atributa koji pripada klasi koja referencira pomenuti target.

```
public override void RemoveReference(ModelCode referenceId, long globalId)
{
    switch (referenceId)
    {
        case ModelCode.WINDINGTEST_POWERTRWINDING:

            if (windingTests.Contains(globalId))
            {
                windingTests.Remove(globalId);
            }
            else
            {
                CommonTrace.WriteTrace(CommonTrace.TraceWarning, "Entity (GID = 0x
            }

            break;

        default:
            base.RemoveReference(referenceId, globalId);
            break;
    }
}
```

InsertEntity() implementacija

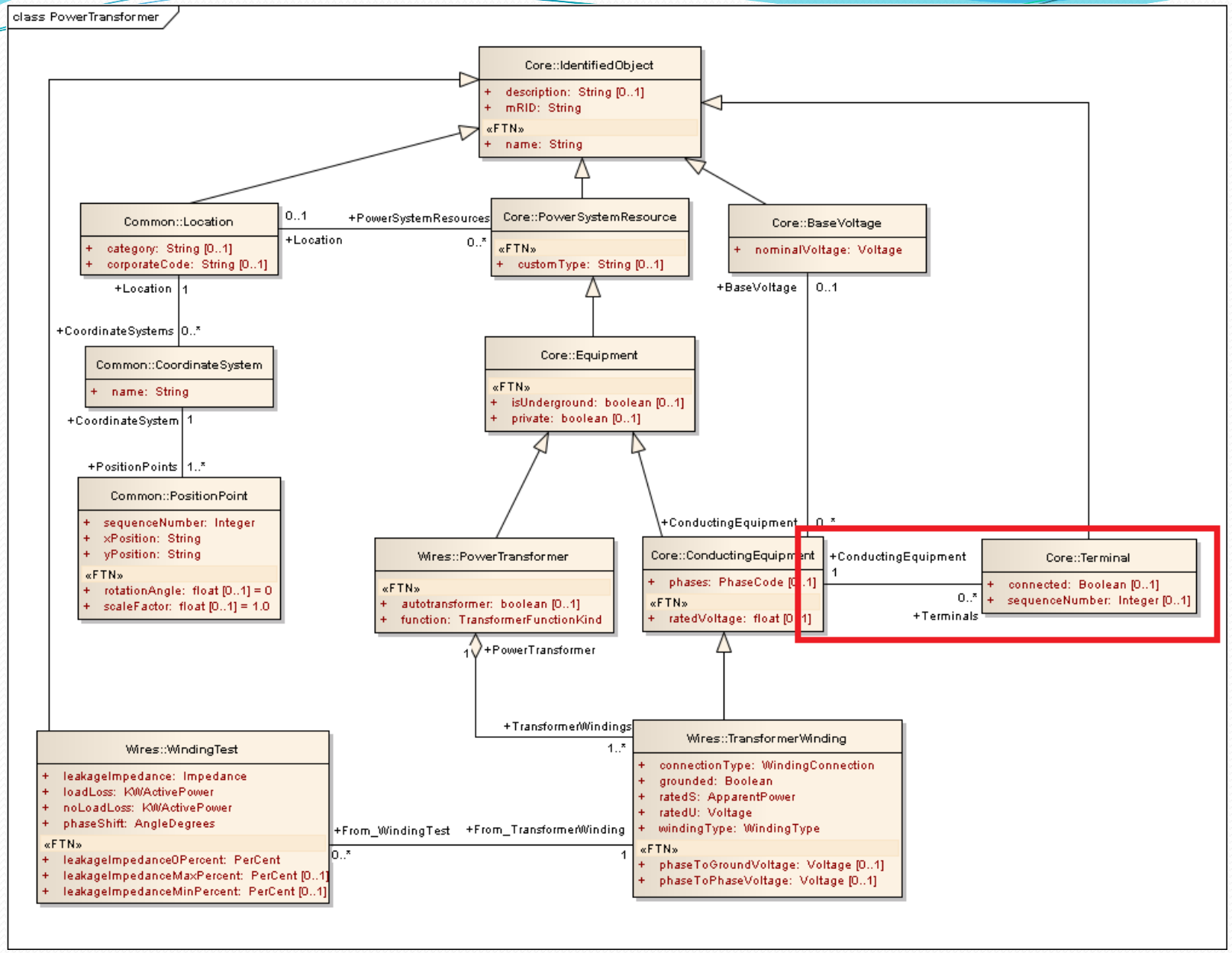
- Razmotrićemo implementaciju *InsertEntity()* metode kako bi dodatno razjasnili ulogu metoda za kreiranje atributa tipa lista referenci (*target*).
- Prilikom primene *delte*, za svaku insert operaciju servis poziva metodu *InsertEntity()* i prosleđuje joj *ResourceDescription* instancu na osnovu koje je potrebno kreirati entitet i smestiti ga u odgovarajuću instancu klase *Container*.
- Za svaki *property* koji postoji u prosleđenom *ResourceDescription*-u se poziva metoda *SetProperty()*. Ukoliko je u pitanju *property* tipa referenca potrebna je dodatna obrada:
  - Pročita se vrednost *property*-a tipa referenca. Vrednost je globalni identifikator entiteta koji se referncira tim *property*-em.
  - Ukoliko je vrednost različita od nule, pronalazi se referencirani entitet i poziva se njegova *AddReference()* metoda.
  - Na ovaj način se *target* atributu referenciranog entiteta (referencira ga prethodno pomenuti *property*) doda globalni identifikator entiteta koji ga referencira.

```
/// <summary>
/// Inserts entity into the network model.
/// </summary>
/// <param name="rd">Description of the resource that should be inserted</param>
private void InsertEntity(ResourceDescription rd)
{
    if (rd == null)
    {
        CommonTrace.WriteTrace(CommonTrace.TraceVerbose, "Insert entity is not done because update operation is empty.");
        return;
    }

    long globalId = rd.Id;

    CommonTrace.WriteTrace(CommonTrace.TraceInfo, "Inserting entity with GID ({0:x16}).", globalId);

    // check if mapping for specified global id already exists
    if (this.EntityExists(globalId))
    {
        string message = String.Format("Failed to insert entity because entity with specified GID ({0:x16}) already exists
        CommonTrace.WriteTrace(CommonTrace.TraceError, message);
        throw new Exception(message);
    }

    try
    {
        // find type
        DMSType type = (DMSType)ModelCodeHelper.ExtractTypeFromGlobalId(globalId);

        Container container = null;

        // get container or create container
        if (ContainerExists(type))
        {
            container = GetContainer(type);
        }
        else
        {
            container = new Container();
            networkDataModel.Add(type, container);
        }

        // create entity and add it to container
        IdentifiedObject io = container.CreateEntity(globalId);
```

```
    // create entity and add it to container
    IdentifiedObject io = container.CreateEntity(globalId);

    // apply properties on created entity
    if (rd.Properties != null)
    {
        foreach (Property property in  rd.Properties)
        {
            // globalId must not be set as property
            if (property.Id == ModelCode.IDOBJ_GID)
            {
                continue;
            }

            if (property.Type == PropertyType.Reference)
            {
                // if property is a reference to another entity
                long targetGlobalId = property.AsReference();

                if (targetGlobalId != 0)
                {

                    if (!EntityExists(targetGlobalId))
                    {
                        string message = string.Format("Failed to get target entity with GID: 0x{0:X16}. {1}", targetGlobalId);
                        throw new Exception(message);
                    }

                    // get referenced entity for update
                    IdentifiedObject targetEntity = GetEntity(targetGlobalId);
                    targetEntity.AddReference(property.Id, io.GlobalId);

                    io.SetProperty(property);
                }
            }
            else
            {
                io.SetProperty(property);
            }
        }
    }

    CommonTrace.WriteTrace(CommonTrace.TraceVerbose, "Inserting entity with GID ({0:x16}) successfully finished.", globalId);
}
catch (Exception ex)
```

Rezime

- Prilikom dodavanja nove klase (ili atributa postojeće klase) potrebno je pratiti sledeće korake:
  1. Dodati odgovarajuće *ModelCode*-ove za nove klase i atribute
  2. Ukoliko su dodate nove enumeracije, proširiti *Enums.cs* fajl
  3. Ukoliko je neki od atributa tipa enumeracija proširiti *EnumDescs* klasu dodatnim mapiranjem
  4. Ukoliko imamo *notSettable* atributa proširiti njima *ModelResourcesDesc* klasu (doraditi metodu *InitializeNotSettablePropertyIds()*)
  5. Ukoliko je dodata konkretna klasa proširiti *ModelResourcesDesc i Container* klase (metode *InitializeTypeIdsInInsertOrder() i CreateEntity()*)
  6. Implementirati nove klase i po potrebi doraditi postojeće na osnovu prethodno definisanih pravila.


class PowerTransformer
Core::IdentifiedObject
+ description: String [0..1]
+ mRID: String
«FTN»
+ name: String
Common::Location
+ category: String [0..1]
+ corporateCode: String [0..1]
0..1
+PowerSystemResources
+Location
0..*
Core::PowerSystemResource
«FTN»
+ customType: String [0..1]
Core::BaseVoltage
+ nominalVoltage: Voltage
+Location
1
+BaseVoltage
0..1
+CoordinateSystems
0..*
Common::CoordinateSystem
+ name: String
Core::Equipment
«FTN»
+ isUnderground: boolean [0..1]
+ private: boolean [0..1]
+CoordinateSystem
1
+PositionPoints
1..*
Common::PositionPoint
+ sequenceNumber: Integer
+ xPosition: String
+ yPosition: String
«FTN»
+ rotationAngle: float [0..1] = 0
+ scaleFactor: float [0..1] = 1.0
+ConductingEquipment
0..*
Wires::PowerTransformer
«FTN»
+ autotransformer: boolean [0..1]
+ function: TransformerFunctionKind
Core::ConductingEquipment
+ phases: PhaseCode [0..1]
«FTN»
+ ratedVoltage: float [0..1]
+ConductingEquipment
1
Core::Terminal
+ connected: Boolean [0..1]
+ sequenceNumber: Integer [0..1]
0..*
+Terminals
1
+PowerTransformer
+TransformerWindings
1..*
Wires::WindingTest
+ leakageImpedance: Impedance
+ loadLoss: KWActivePower
+ noLoadLoss: KWActivePower
+ phaseShift: AngleDegrees
«FTN»
+ leakageImpedance0Percent: PerCent
+ leakageImpedanceMaxPercent: PerCent [0..1]
+ leakageImpedanceMinPercent: PerCent [0..1]
+From_WindingTest
0..*
+From_TransformerWinding
1
Wires::TransformerWinding
+ connectionType: WindingConnection
+ grounded: Boolean
+ ratedS: ApparentPower
+ ratedU: Voltage
+ windingType: WindingType
«FTN»
+ phaseToGroundVoltage: Voltage [0..1]
+ phaseToPhaseVoltage: Voltage [0..1]

Zadaci

1. Dodati klasu *Terminal* koja ima referencu prema klasi *ConductingEquipment* (Napomena: Obratiti pažnju u koji paket se dodaje nova klasa i da su potrebne dodatne izmene nad klasom *ConductingEquipment*)