App Development & Modeling

BSc in Applied Computing

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

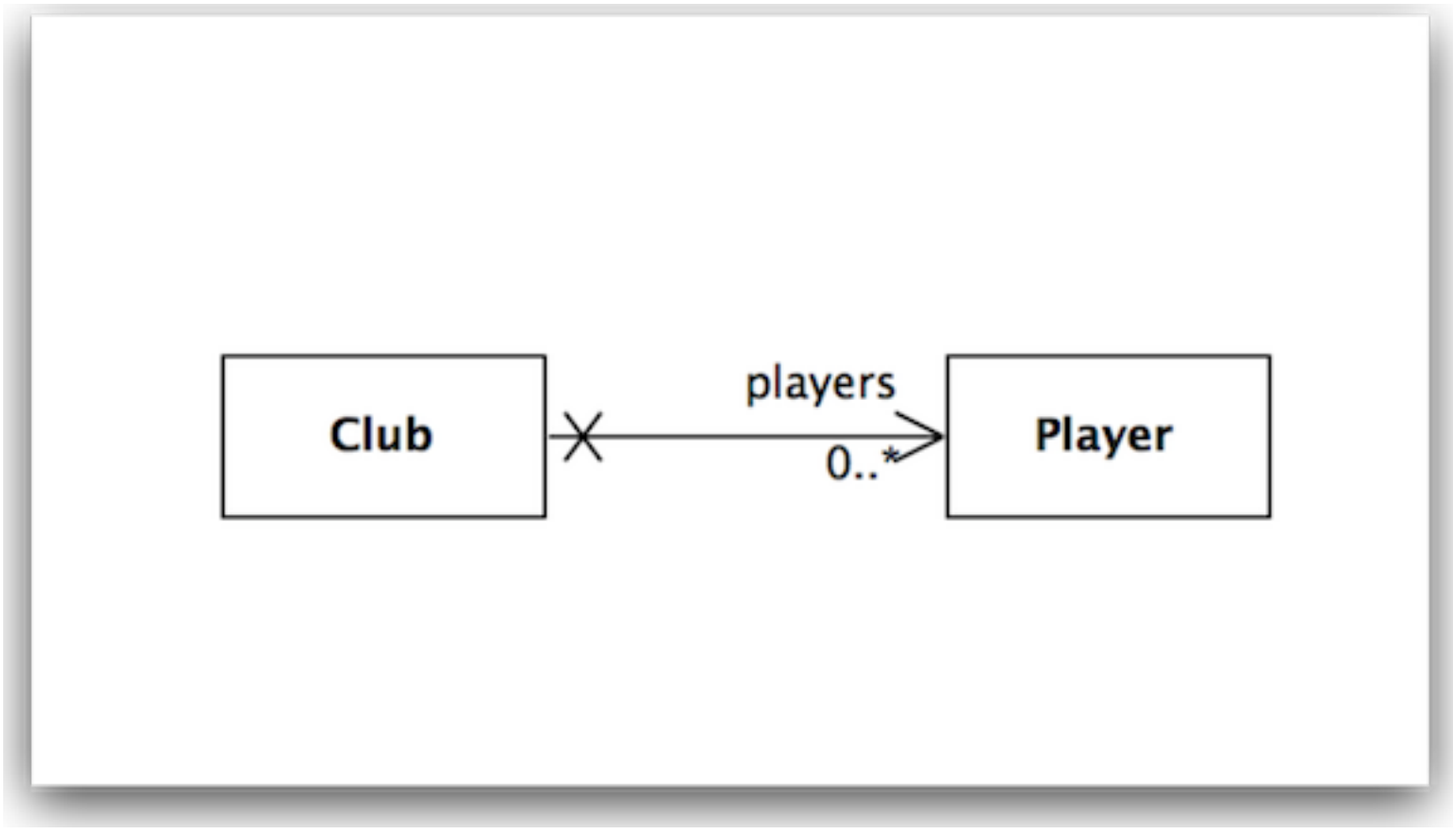
http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Modeling & JPA 2

OneToMany

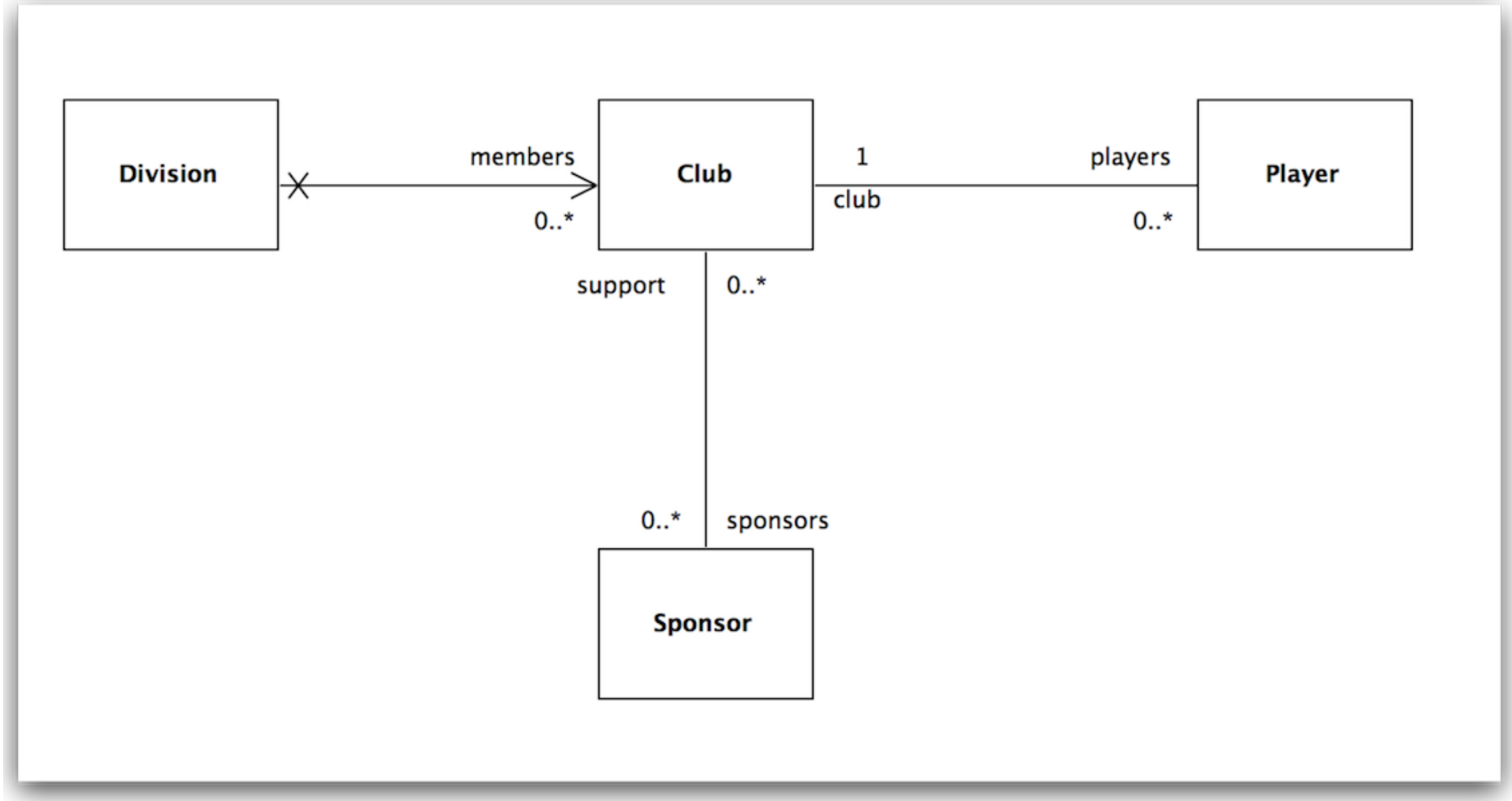


OneToMany - Unidirectional

```
public class Club extends Model
{
  public String name;

  @OneToMany(cascade=CascadeType.ALL)
  public List<Player>
              players = new ArrayList<Player>();

  public Club(String name)
  {
    this.name = name;
  }

  public String toString()
  {
    return name;
  }

  public void addPlayer(Player player)
  {
    players.add(player);
  }
}
```

```
public class Player extends Model
{
  public String name;


  public Player(String name)
  {
    this.name = name;
  }

  public String toString()
  {
    return name;
  }
}
```

OneToMany, ManyToOne, ManyToMany



OneToMany

```
public class Division extends Model
{
  public String name;

  @OneToMany(cascade=CascadeType.ALL)
  public List<Club> members new ArrayList<Club>();

  public Division(String name)
  {
    this.name = name;
  }

  public void addClub(Club club)
  {
    members.add(club);
  }

  public String toString()
  {
    return name;
  }

  public static Division findByName(String name)
  {
    return find("name", name).first();
  }
}
```

```
public class Club extends Model
{
  public String name;

  @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
  public List<Player> players = new ArrayList<Player>();

  @ManyToMany
  public List<Sponsor> sponsors = new ArrayList<Sponsor>();

  public Club(String name)
  {
    this.name = name;
  }
  public String toString()
  {
    return name;
  }
  public static Club findByName(String name)
  {
    return find("name", name).first();
  }
  public void addPlayer(Player player)
  {
    player.club = this;
    players.add(player);
  }
  public void addSponsor(Sponsor company)
  {
    sponsors.add(company);
  }
  public void removePlayer(Player player)
  {
    players.remove(player);
  }
}
```

ManyToOne

```
public class Club extends Model
{
  public String name;

  @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
  public List<Player> players = new ArrayList<Player>();

  //..
}
```

```
public class Player extends Model
{
  public String name;

  @ManyToOne
  public Club club;

  public Player(String name)
  {
    this.name = name;
  }

  public String toString()
  {
    return name;
  }

  public static Player findByName(String name)
  {
    return find("name", name).first();
  }
}
```

ManyToMany

```
public class Sponsor extends Model
{
  public String name;

  @ManyToMany (mappedBy="sponsors")
  public List<Club> support = new ArrayList<Club>();

  public Sponsor(String name)
  {
    this.name = name;
  }

  public void addSuport(Club club)
  {
    support.add(club);
  }

  public String toString()
  {
    return name;
  }
}
```

```
public class Club extends Model
{
  public String name;

  @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
  public List<Player> players = new ArrayList<Player>();

  @ManyToMany
  public List<Sponsor> sponsors = new ArrayList<Sponsor>();

  public Club(String name)
  {
    this.name = name;
  }
  public String toString()
  {
    return name;
  }
  public static Club findByName(String name)
  {
    return find("name", name).first();
  }
  public void addPlayer(Player player)
  {
    player.club = this;
    players.add(player);
  }
  public void addSponsor(Sponsor company)
  {
    sponsors.add(company);
  }
  public void removePlayer(Player player)
  {
    players.remove(player);
  }
}
```

Tests

- For more complex models, create fixtures in data.yml.
- These models can be loaded in unit tests

data.yml

```
Club(dunmore):
    name: dunmore

Club(tramore):
    name: tramore

Club(fenor):
    name: fenor

Player(jim):
    name: jim
    club: dunmore

Player(mary):
    name: mary
    club: dunmore

Player(sam):
    name: sam
    club: tramore

Player(john):
    name: john
    club: tramore

Player(mike):
    name: mike
    club: fenor

Player(linda):
    name: john
    club: fenor

Division(senior):
    name: senior
    members:
            - tramore
            - dunmore

Division(junior):
    name: junior
    members:
            - fenor

Sponsor(newsagent):
    name: newsagent

Sponsor(pub):
    name: pub
```

data.yml

```
Club(dunmore):
    name: dunmore

Club(tramore):
    name: tramore

Club(fenor):
    name: fenor

Player(jim):
    name: jim
    club: dunmore

Player(mary):
    name: mary
    club: dunmore

Player(sam):
    name: sam
    club: tramore

Player(john):
    name: john
    club: tramore

Player(mike):
    name: mike
    club: fenor

Player(linda):
    name: john
    club: fenor

Division(senior):
    name: senior
    members:
            - tramore
            - dunmore

Division(junior):
    name: junior
    members:
            - fenor

Sponsor(newsagent):
    name: newsagent

Sponsor(pub):
    name: pub
```

ComprehensiveTest

```
public class ComprehensiveTest extends UnitTest
{
  @Before
  public void setup()
  {
    Fixtures.deleteDatabase();
    Fixtures.loadModels("data.yml");
  }

  @After
  public void teardown()
  {
    Fixtures.deleteAllModels();
  }
```

Test Strategy

- For each relationship:
  - ‘short’ test - quick sanity check
  - ‘long’ test - full exercise of relationship, in both directions if present
  - ‘edit’ test - perform change on objects

Player/Club

```
 @Test
public void testPlayerClub()
{
  Club   dunmore = Club.find("byName", "dunmore").first();
  Player jim     = Player.find("byName", "jim").first();
  Player mary    = Player.find("byName", "mary").first();
  assertNotNull(mary);

  assertTrue (dunmore.players.contains(jim));
  assertTrue (dunmore.players.contains(mary));
}
```

```
@Test
public void testPlayerClubLong()
{
  Player jim;
  Club   dunmore;

  jim = Player.find("byName", "jim").first();
  assertNotNull(jim);
  assertEquals(jim.name, "jim");

  dunmore = jim.club;
  assertEquals("dunmore", dunmore.name);

  dunmore = Club.find("byName", "dunmore").first(
  assertNotNull(dunmore);
  assertEquals("dunmore", dunmore.name);
  assertEquals(2, dunmore.players.size());

  Player p1 = dunmore.players.get(0);
  assertTrue (p1.name.equals("jim") || p1.name.equals("mary"));
  Player p2 = dunmore.players.get(1);
  assertTrue (p2.name.equals("jim") || p2.name.equals("mary"));
}
```

```
@Test
public void testEditPlayerClub()
{
  Club   dunmore = Club.find("byName", "dunmore").first();
  Player jim     = Player.find("byName", "jim").first();
  Player mary    = Player.find("byName", "mary").first();

  dunmore.players.remove(mary);
  mary.delete();
  dunmore.save();

  assertEquals (dunmore.players.size(), 1);
  assertTrue (dunmore.players.contains(jim));

  assertEquals(0, Player.find("byName", "mary").fetch().size());

  Player sara     = new Player("sara");
  dunmore.addPlayer(sara);
  dunmore.save();
  assertEquals (dunmore.players.size(), 2);
}
```

Forward References

- In yaml files, representing many-to-many relationships cannot be easily represented.

- e.g:

  - dunmore->newsagent

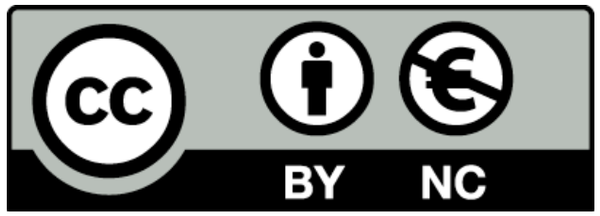
  - newsagent->dunmore

```
Club(dunmore):
    name: dunmore

Player(jim):
    name: jim
    club: dunmore

Player(mary):
    name: mary
    club: dunmore

Division(junior):
    name: junior
    members:
            - dunmore

Sponsor(newsagent):
    name: newsagent
```

Forward References - Workaround

- Load the data.yaml model without ManyToMany
- Establish the relationship after the fixture is loaded

```
public class ComprehensiveTest extends UnitTest
{
  public static void loadSponsorships()
  {
    Club    tramore   = Club.find("byName", "tramore").first();
    Club    dunmore   = Club.find("byName", "dunmore").first();
    Sponsor newsagent = Sponsor.find("byName", "newsagent").first();

    tramore.addSponsor(newsagent);
    dunmore.addSponsor(newsagent);

    newsagent.addSuport(tramore);
    newsagent.addSuport(dunmore);

    tramore.save();
    dunmore.save();
    newsagent.save();
  }

  @Before
  public void setup()
  {
    Fixtures.loadModels("data.yml");
    loadSponsorships();
  }
```



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit