# App Development & Modelling

BSc in Applied Computing



Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie



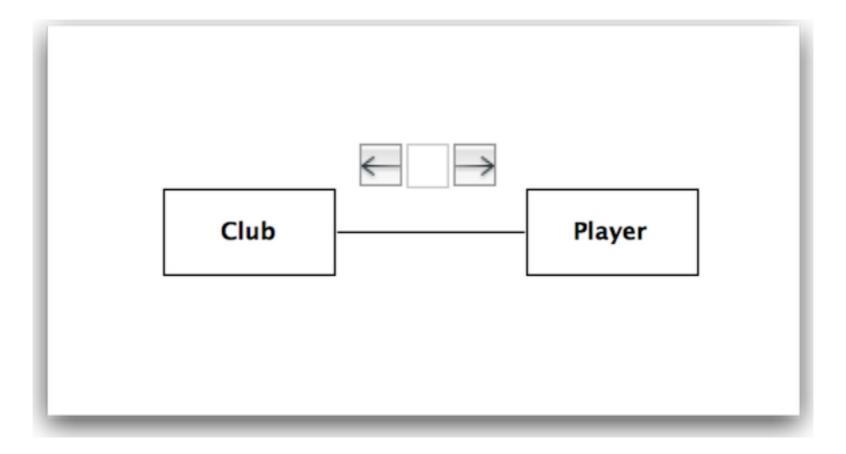
Waterford Institute of Technology INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



## Modeling Relationships

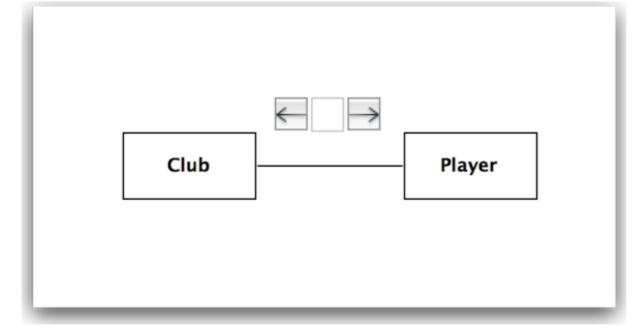
#### Associations

• In Visual Paradigm, on the palette on the left, select the 'association' element and use it to connect Club and Player.



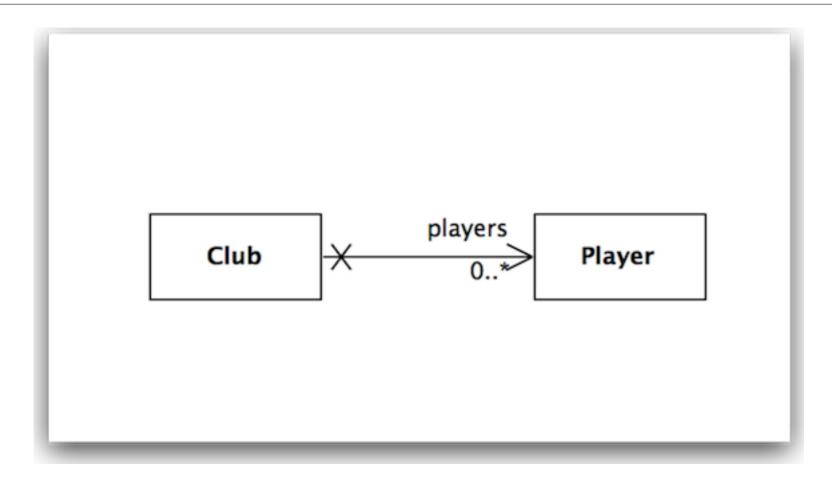
## Association Attributes

 Select the association (the line), and locate the following panel:

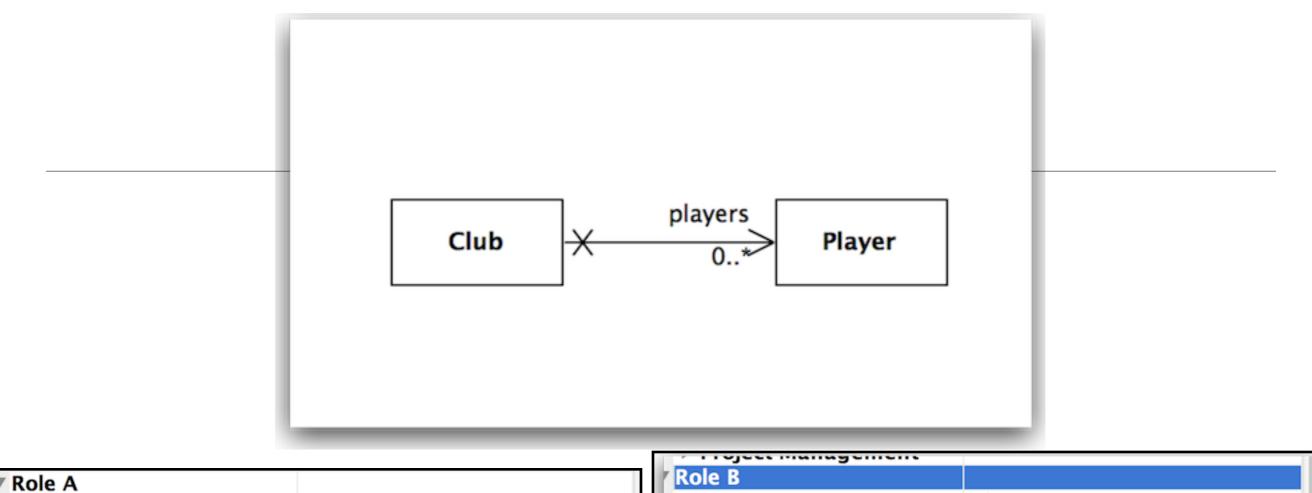


🔁 Property 🔯 Diagram	🛃 Documen   🔣 Stencil 🔯 Teamwor
000	Property
Club-Player - Association	\$]
30 <b>12</b> 😤 🔄	
Name	
Parent	<none></none>
View	
Role A	
Name	
▶ Club	
Multiplicity	<unspecified></unspecified>
Navigable	<unspecified></unspecified>
Visibility	<unspecified></unspecified>
Aggregation Kind	None
Stereotypes	<unspecified></unspecified>
Tagged Values	
Comments	
Project Management	
Role B	
Name Name	
Player	<ul> <li>Increasified&gt;</li> </ul>
Multiplicity Navigable	<unspecified></unspecified>
Visibility	<unspecified></unspecified>
Aggregation Kind	<unspecified></unspecified>
Stereotypes	<unspecified></unspecified>
Tagged Values	<ol> <li>Source and a second seco</li></ol>
Comments	
Project Management	
Visibility	<unspecified></unspecified>
Abstract	Conspective
Leaf	
Stereotypes	<unspecified></unspecified>
Tagged Values	
Comments	
Project Management	

### Multiplicity & Navigation



- Club has a collection of zero or more players
- Players are unaware of Club



Role A		Role B	
Name		Name	players
▶ Club		▶ Player	
Multiplicity	<unspecified></unspecified>	Multiplicity	0*
Navigable	False	Navigable	True
Visibility	<unspecified></unspecified>	Visibility	<unspecified></unspecified>
Aggregation Kind	None	Aggregation Kind	d None
Stereotypes	<unspecified></unspecified>	Stereotypes	<unspecified></unspecified>
Tagged Values		Tagged Values	
Comments		Comments	

## Implementation Relationship in Java Classes

```
public class Club extends Model
{
 public String name;
 @OneToMany(cascade=CascadeType.ALL)
  public List<Player> players;
 public Club(String name)
    this.name = name;
    this.players = new ArrayList<Player>();
  }
  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()
  Ł
    return name;
  }
```

### Testing the Player / Club Relationship

 Use the fixture to set up some club / relationships

```
@Before
public void setup()
{
  p1 = new Player("mike");
  p2 = new Player("jim");
  p3 = new Player("frank");
  c1 = new Club("tramore");
  c2 = new Club("dunmore");
  c3 = new Club("fenor");
  c1.addPlayer(p1);
  c1.addPlayer(p2);
  c1.save();
  c2.save();
  c3.save();
}
```

#### testPlayers

 In the test, see if these relationship have been established

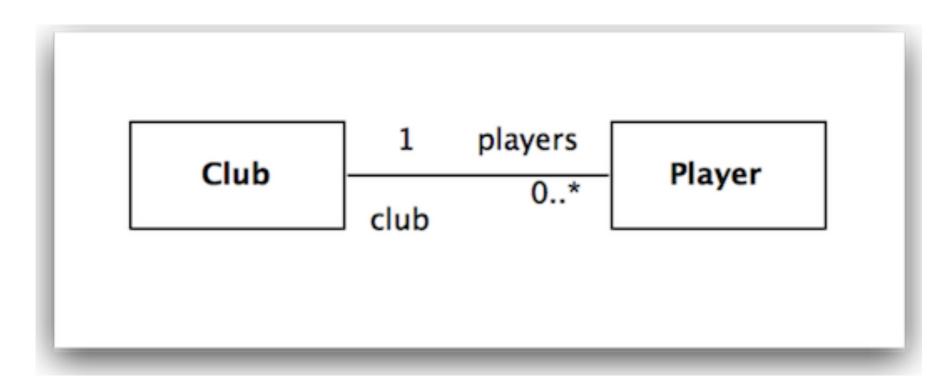
```
@Test
public void testPlayers()
{
  Club tramore = Club.findByName("tramore");
  assertEquals (2, tramore.players.size());
  Player mike = Player.findByName("mike");
  Player jim = Player.findByName("jim");
  Player frank = Player.findByName("framk");
  assertTrue (tramore.players.contains(mike));
  assertTrue (tramore.players.contains(jim));
  assertFalse (tramore.players.contains(frank));
}
```

#### testRemovePlayers

 Removing relationships must also be tested

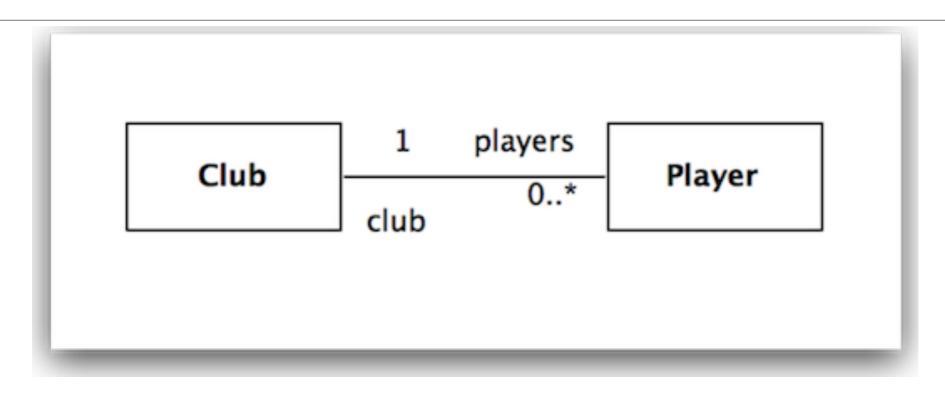
```
@Test
public void testRemovePlayer()
Ł
  Club tramore = Club.findByName("tramore");
  assertEquals(2, tramore.players.size());
  Player mike = Player.findByName("mike");
  assertTrue(tramore.players.contains(mike));
  tramore.players.remove(mike);
  tramore.save();
  Club c = Club.findByName("tramore");
  assertEquals(1, c.players.size());
  mike.delete();
}
```

## **Bidirectional Relationship**



- Club has a 'one to many' relationship with players
- Player has a 'many to one' relationship with club

## **Bidirectional Relationship**



Role A	
Name	club
▶ Club	
Multiplicity	1
Navigable	True
Visibility	<unspecified></unspecified>
Aggregation Kind	None
Stereotypes	<unspecified></unspecified>
Tagged Values	
Comments	

## Bidirectional Relationship in Java Classes

```
public class Club extends Model
{
 public String name;
 @OneToMany(mappedBy="club", cascade=CascadeType.ALL)
 public List<Player> players;
 public Club(String name)
  Ł
   this.name = name;
   this.players = new ArrayList<Player>();
  }
 public String toString()
    return name;
  }
 public void addPlayer(Player player)
    player.club = this;
    players.add(player);
```

```
public class Player extends
Model
Ł
  public String name;
  @ManyToOne
  public Club club;
  public Player(String name)
    this.name = name;
  }
  public String toString()
    return name;
  }
```

# Unidirectional Relationship in Java Classes

```
public class Club extends Model
```

```
public String name;
```

{

}

Ł

}

}

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

```
public Club(String name)
```

```
{
   this.name = name;
   this.players = new ArrayList<Player>();
```

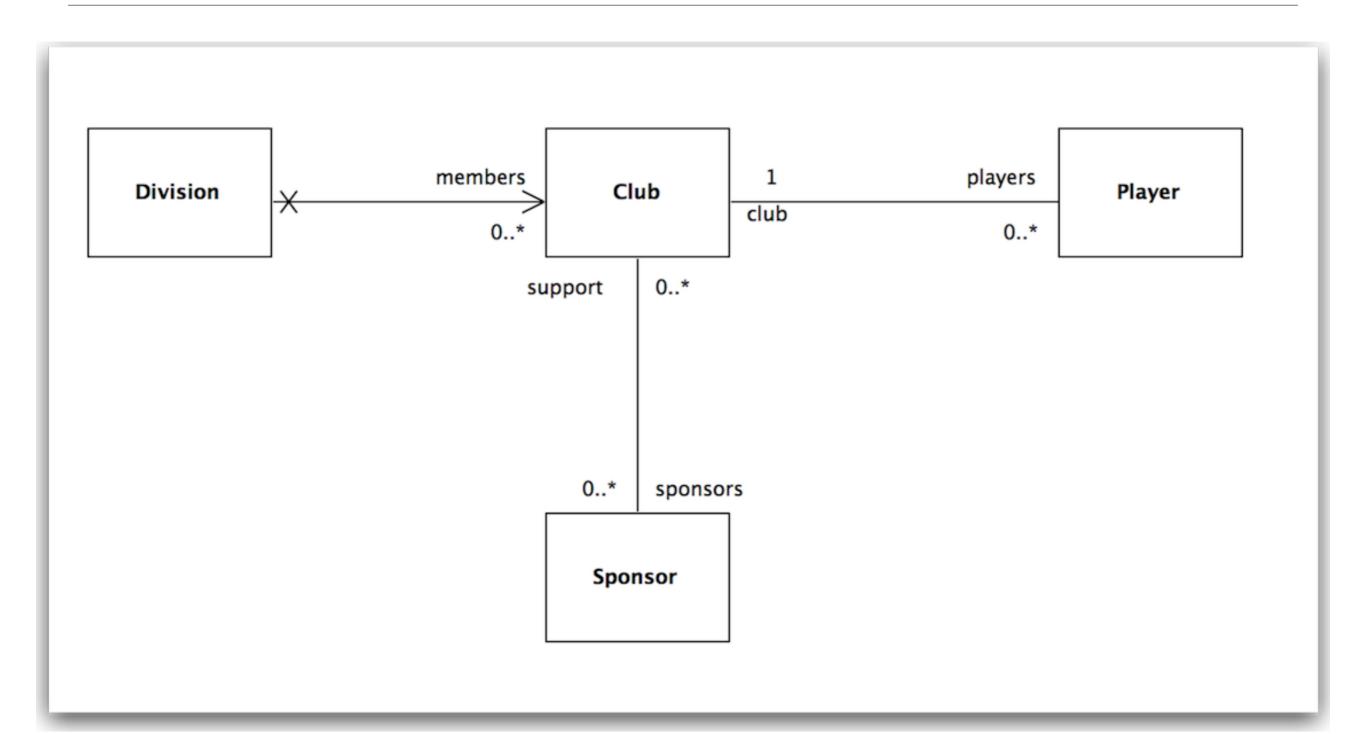
```
public String toString()
```

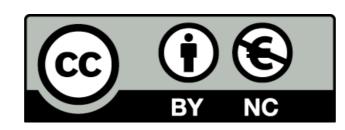
```
return name;
```

```
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;
   }
}
```

#### Exercise: Model This:





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see http:// creativecommons.org/licenses/by-nc/3.0/



Waterford Institute of Technology INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

