**----Test variables----**

<<var [ManagerTestClass manager = null]>>

**----Test variables----**

----All (Predicate)----

<<[Managers.all((ManagerTestClass m) => m.getAge() < 30)]>>

----Any()----

<<[Managers.any()]>>

----Any(Predicate)----

<<[Managers.any((ManagerTestClass m) => m.getName() == “John Smith”)]>>

----Average(Selector)----

<<[Managers.average((ManagerTestClass m) => m.getAge())]>>

----Concat(IEnumerable)----

<<foreach [ManagerTestClass m in Managers.concat(Managers)]>>Name = <<[m.getName()]>>, <</foreach>>

----Contains(Object)----

<<var [manager = Managers.first()]>><<[Managers.contains(manager)]>>

----Count()----

<<[Managers.count()]>>

----Count(Predicate)----

<<[Managers.count((ManagerTestClass m) => m.getAge() < 30)]>>

----Distinct()----

<<foreach [ManagerTestClass m in Managers.distinct()]>><<[m.getName()]>><</foreach>>

----First()----

<<var [manager = Managers.first()]>><<[manager.getName()]>>

----First(Predicate)----

<<var [manager = Managers.first((ManagerTestClass m) => m.getAge() < 40)]>><<[ manager.getName()]>>

----FirstOrDefault()----

<<var [manager = Managers.firstOrDefault()]>><<[manager.getName()]>>

----FirstOrDefault(Predicate)----

<<var [manager = Managers.firstOrDefault((ManagerTestClass m) => m.getAge() < 50)]>><<[ manager.getName()]>>

----GroupBy(Selector)----

<<foreach [m in Managers.groupBy((ManagerTestClass m) => new { age = m.getAge(), name = m.getName() })]>><<foreach [ManagerTestClass c in m]>>Age = <<[c.getAge()]>>, Name = <<[c.getName()]>>; <</foreach>><</foreach>>

----Last()----

<<var [manager = Managers.last()]>><<[manager.getName()]>>

----Last(Predicate)----

<<var [manager = Managers.last((ManagerTestClass m) => m.getAge() < 50)]>><<[manager.getName()]>>

----LastOrDefault()----

<<var [manager = Managers.lastOrDefault()]>><<[manager.getName()]>>

----LastOrDefault(Predicate)----

<<var [manager = Managers.lastOrDefault((ManagerTestClass m) => m.getAge() < 50)]>><<[manager.getName()]>>

----Max(ComparableSelector)----

<<[Managers.max((ManagerTestClass m) => m.getAge())]>>

----Min(ComparableSelector)----

<<[Managers.min((ManagerTestClass m) => m.getAge())]>>

----OrderBy(ComparableSelector)----

<<foreach [ManagerTestClass m in Managers.orderBy((ManagerTestClass m) => m.getAge()).thenBy((ManagerTestClass m) => m.getContracts().sum(c => c.getPrice()))]>>Name = <<[m.getName()]>>, <</foreach>>

----OrderByDescending(ComparableSelector)----

<<foreach [ManagerTestClass m in Managers.orderByDescending((ManagerTestClass m) => m.getAge())]>>Name = <<[m.getName()]>>, <</foreach>>

----Single()----

<<var [manager = Managers.where((ManagerTestClass m) => m.getName() == “John Smith”).single()]>><<[manager.getName()]>>

----Single(Predicate)----

<<var [manager = Managers. where((ManagerTestClass m) => m.getName() == “John Smith”).single((ManagerTestClass m) => m.getName() == “John Smith”)]>><<[manager.getName()]>>

----SingleOrDefault()----

<<var [manager = Managers.where((ManagerTestClass m) => m.getName() == “John Smith”).singleOrDefault()]>><<[manager.getName()]>>

----SingleOrDefault(Predicate)----

<<var [manager = Managers.where((ManagerTestClass m) => m.getName() == “John Smith”).singleOrDefault((ManagerTestClass m) => m.getName() == “John Smith”)]>><<[manager.getName()]>>

----Skip(Int)----

<<foreach [ManagerTestClass m in Managers.skip(2)]>><<[m.getName()]>><</foreach>>

----SkipWhile(Predicate)----

<<foreach [ManagerTestClass m in Managers.skipWhile((ManagerTestClass m) => m.getAge() < 37)]>><<[m.getName()]>><</foreach>>

----Sum(Selector)----

<<[Managers.sum((ManagerTestClass m) => m.getContracts().count())]>>

----Take(int)----

<<foreach [ManagerTestClass m in Managers.take(3)]>>Name = <<[m.getName()]>>, <</foreach>>

----TakeWhile(Predicate)----

<<foreach [ManagerTestClass m in Managers.takeWhile((ManagerTestClass m) => m.getAge() < 40)]>>Name = <<[m.getName()]>>, <</foreach>>

----Union(IEnumerable)----

<<foreach [ManagerTestClass m in Managers.union(Managers)]>>Name = <<[m.getName()]>>, <</foreach>>

----Where(Predicate)----

<<foreach [ManagerTestClass m in Managers.where((ManagerTestClass m) => m.getAge() < 45)]>>Name = <<[m.getName()]>>, <</foreach>>