EF Writing (creating, updating, deleting) Lab

So far, we've got the R in CRUD, but we still need the C, U, and D. In this lab, we can round out our repositories by adding the ability to create, update and delete records.

Writing the tests first

1. For each of your EmployeeTests, CustomerTests, and ProductTests classes, add these tests:
   - CanCreate()
   - CanUpdate()
   - CanDelete()

They may look something like this:

```
public void CanCreate()
{
  var r = new
    EntityRepository();
  var e = new Entity() {
    //Set properties
  };
  r.Create(e);
  var e2 = r.Read(e.Id);
  Assert.AreEqual(
    e.prop, e1.prop);
  r.Delete(e);
}
```

```
public void CanUpdate()
{
  var r = new
    EntityRepository();
  var e = r
    .ReadAll().First();
  var oldProp =e.Prop;
  e.Prop = "changed";
  r.Update(e);
  var e2 = r.Read(e.Id);
  Assert.AreEqual(
    e.Prop, e2.Prop);
}
```

```
public void CanDelete()
{
  var r = new
    EntityRepository();
  var e = new Entity() {
    //Set properties
  };
  r.Create(e);
  Assert.True(r.Exists(e));
  r.Remove(e);
  Assert.False(r.Exists(e));
}
```

Where Entity is Employee or Customer or Product. That's nine tests in all.

2. Try to build. You shouldn't be able to ... yet.

Writing the repository methods

Let's make it compile.

3. Open each of your repositories in turn and write the three methods, Create(), Remove(), and Update().
   HInts:
   - To create, go DbSet.Add(newEntity);
   - To update, go DbContext.Entry(Entity).State = EntityState.Modified;
   - To delete, go DbSet.Remove(Entity);
   - Don't forget to DbContext.SaveChanges() after each of your changes.
4. Build and test.

Once your tests all turn green, you can be finished.