```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace DataAccessLayer
8  {
9      public class CourseRepository : Repository<Course>, ICourseRepository
10     {
11         public CourseRepository() : base(new SchoolDBEntities())
12         {
13
14         }
15     }
16 }
17
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using DataAccessLayer;
7
8  namespace BusinessLayer
9  {
10     public interface IBusinessLayer
11     {
12         IList<Standard> getAllStandards();
13         Standard GetStandardByID(int id);
14         Standard GetStandardByName(string name);
15         void AddStandard(Standard standard);
16         void UpdateStandard(Standard standard);
17         void RemoveStandard(Standard standard);
18
19         IList<Student> getAllStudents();
20         Student GetStudentByID(int id);
21         Student GetStudentByName(string name);
22         void AddStudent(Student student);
23         void UpdateStudent(Student student);
24         void RemoveStudent(Student student);
25
26         IList<Teacher> getAllTeachers();
27         Teacher GetTeacherByID(int id);
28         Teacher GetTeacherByName(string name);
29         void AddTeacher(Teacher teacher);
30         void UpdateTeacher(Teacher teacher);
31         void RemoveTeacher(Teacher teacher);
32
33         IList<Course> getAllCourses();
34         Course GetCourseByID(int id);
35         Course GetCourseByName(string name);
36         void AddCourse(Course course);
37         void UpdateCourse(Course course);
38         void RemoveCourse(Course course);
39
40         IList<Course> GetCoursesByTeacherID(int id);
41         IList<Course> GetCoursesByTeacherName(string name);
42
43         IList<Student> GetStudentsByStandardID(int id);
44     }
45 }
46
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace DataAccessLayer
 8  {
 9      public interface ICourseRepository : IRepository<Course>
10      {
11
12      }
13  }
14
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Linq.Expressions;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace DataAccessLayer
9  {
10     public interface IRepository<T> : IDisposable
11     {
12         void Insert(T entity);
13
14         void Delete(T entity);
15
16         void Update(T entity);
17
18         T GetById(int id);
19
20         IQueryable<T> SearchFor(Expression<Func<T, bool>> predicate);
21
22         IEnumerable<T> GetAll();
23
24         T GetSingle(Func<T, bool> where, params Expression<Func<T, object>>[]
             navigationProperties);
25     }
26 }
27
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace DataAccessLayer
8  {
9      public interface IStandardRepository : IRepository<Standard>
10     {
11
12     }
13 }
14
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace DataAccessLayer
8  {
9      public interface IStudentRepository : IRepository<Student>
10     {
11
12     }
13 }
14
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace DataAccessLayer
 8  {
 9      public interface ITeacherRepository : IRepository<Teacher>
10      {
11
12      }
13  }
14
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Linq.Expressions;
7  using System.Data.Entity;
8
9  namespace DataAccessLayer
10 {
11     public class Repository<T> : IRepository<T> where T : class
12     {
13         protected DbContext context;
14         protected DbSet<T> dbset;
15         public Repository(DbContext datacontext)
16         {
17             //You can use the cpmt
18             this.context = datacontext;
19             dbset = context.Set<T>();
20
21         }
22
23         public void Insert(T entity)
24         {
25             //Use the context object and entity state to save the entity
26             context.Entry(entity).State = EntityState.Added;
27             context.SaveChanges();
28
29         }
30
31         public void Delete(T entity)
32         {
33             //Use the context object and entity state to delete the entity
34             context.Entry(entity).State = EntityState.Detached;
35             context.SaveChanges();
36         }
37
38         public void Update(T entity)
39         {
40             //Use the context object and entity state to update the entity
41             context.Entry(entity).State = EntityState.Modified;
42             context.SaveChanges();
43         }
44
45         public T GetById(int id)
46         {
47             return dbset.Find(id);
48         }
49
```

```
50          public IQueryable<T> SearchFor(Expression<Func<T, bool>> predicate)
51          {
52              return dbset.Where(predicate);
53              //return context.Where(predicate);
54          }
55
56          public IEnumerable<T> GetAll()
57          {
58              return dbset.ToList();
59          }
60
61          //This method will find the related records by passing two argument
62          //First argument: lambda expression to search a record such as d =>
              d.StandardName.Equals(standardName) to search am record by standard
              name
63          //Second argument: navigation property that leads to the related records
               such as d => d.Students
64          //The method returns the related records that met the condition in the
              first argument.
65          //An example of the method GetStandardByName(string standardName)
66          //public Standard GetStandardByName(string standardName)
67          //{
68          //return _standardRepository.GetSingle(d => d.StandardName.Equals
              (standardName), d => d.Students);
69          //}
70          public T GetSingle(Func<T, bool> where, params Expression<Func<T,
              object>>[] navigationProperties)
71          {
72              T item = null;
73              IQueryable<T> dbQuery = context.Set<T>();
74              foreach (Expression<Func<T, object>> navigationProperty in
                  navigationProperties)
75                  dbQuery = dbQuery.Include<T, object>(navigationProperty);
76              item = dbQuery.AsNoTracking().FirstOrDefault(where);
77              return item;
78
79          }
80
81          public void Dispose()
82          {
83              throw new NotImplementedException();
84          }
85      }
86  }
87
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace DataAccessLayer
8  {
9      public class StandardRepository : Repository<Standard>, IStandardRepository
10     {
11         public StandardRepository() : base(new SchoolDBEntities())
12         {
13         }
14     }
15 }
16
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace DataAccessLayer
8  {
9      public class StudentRepository : Repository<Student>, IStudentRepository
10     {
11         public StudentRepository() : base(new SchoolDBEntities())
12         {
13
14         }
15     }
16 }
17
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace DataAccessLayer
8  {
9      public class TeacherRepository : Repository<Teacher>, ITeacherRepository
10     {
11         public TeacherRepository() : base(new SchoolDBEntities())
12         {
13         }
14     }
15 }
16
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6  using DataAccessLayer;
 7
 8  namespace BusinessLayer
 9  {
10      public class BusinessLayer : IBusinessLayer
11      {
12          private readonly IStandardRepository standardRepository;
13          private readonly IStudentRepository studentRepository;
14          private readonly ITeacherRepository teacherRepository;
15          private readonly ICourseRepository courseRepository;
16
17          public BusinessLayer()
18          {
19              standardRepository = new StandardRepository();
20              studentRepository = new StudentRepository();
21              courseRepository = new CourseRepository();
22              teacherRepository = new TeacherRepository();
23          }
24
25          public void AddStandard(Standard standard)
26          {
27              standardRepository.Insert(standard);
28              //throw new NotImplementedException();
29          }
30
31          public void AddStudent(Student student)
32          {
33              studentRepository.Insert(student);
34              //throw new NotImplementedException();
35          }
36
37          public IList<Standard> getAllStandards()
38          {
39              return standardRepository.GetAll().ToList();
40          }
41
42          public IList<Student> getAllStudents()
43          {
44              return studentRepository.GetAll().ToList();
45              //throw new NotImplementedException();
46          }
47
48          public Standard GetStandardByID(int id)
49          {
```

```
50              return standardRepository.GetById(id);
51              //throw new NotImplementedException();
52          }
53
54          public Standard GetStandardByName(string name)
55          {
56              return standardRepository.GetSingle(s => s.StandardName.Equals
                  (name), s => s.Students, s => s.Teachers);
57          }
58
59          public Student GetStudentByID(int id)
60          {
61              return studentRepository.GetById(id);
62              //throw new NotImplementedException();
63          }
64
65          public Student GetStudentByName(string name)
66          {
67              return studentRepository.GetSingle(s => s.StudentName.Equals(name),
                    s => s.StudentAddress);
68          }
69
70          public void RemoveStandard(Standard standard)
71          {
72              standardRepository.Delete(standard);
73              //throw new NotImplementedException();
74          }
75
76          public void RemoveStudent(Student student)
77          {
78              studentRepository.Delete(student);
79              //throw new NotImplementedException();
80          }
81
82          public void UpdateStandard(Standard standard)
83          {
84              standardRepository.Update(standard);
85              //throw new NotImplementedException();
86          }
87
88          public void UpdateStudent(Student student)
89          {
90              studentRepository.Update(student);
91              //throw new NotImplementedException();
92          }
93
94          public IList<Teacher> getAllTeachers()
95          {
96              return teacherRepository.GetAll().ToList();
```

```
 97          }
 98
 99          public Teacher GetTeacherByID(int id)
100          {
101              return teacherRepository.GetById(id);
102          }
103
104          public Teacher GetTeacherByName(string name)
105          {
106              return teacherRepository.GetSingle(t => t.TeacherName.Equals(name), ↵
                     t => t.Standard);
107          }
108
109          public void AddTeacher(Teacher teacher)
110          {
111              teacherRepository.Insert(teacher);
112          }
113
114          public void UpdateTeacher(Teacher teacher)
115          {
116              teacherRepository.Update(teacher);
117          }
118
119          public void RemoveTeacher(Teacher teacher)
120          {
121              teacherRepository.Delete(teacher);
122          }
123
124          public IList<Course> getAllCourses()
125          {
126              return courseRepository.GetAll().ToList();
127          }
128
129          public Course GetCourseByID(int id)
130          {
131              return courseRepository.GetById(id);
132          }
133
134          public Course GetCourseByName(string name)
135          {
136              return courseRepository.GetSingle(c => c.CourseName.Equals(name), c ↵
                     => c.Teacher);
137          }
138
139          public void AddCourse(Course course)
140          {
141              courseRepository.Insert(course);
142          }
143
```

```
144         public void UpdateCourse(Course course)
145         {
146             courseRepository.Update(course);
147         }
148 
149         public void RemoveCourse(Course course)
150         {
151             courseRepository.Delete(course);
152         }
153 
154         public IList<Course> GetCoursesByTeacherID(int id)
155         {
156             return courseRepository.SearchFor(c => c.TeacherId ==
                  id).ToList<Course>();
157         }
158 
159         public IList<Course> GetCoursesByTeacherName(string name)
160         {
161             return courseRepository.SearchFor(c => c.Teacher.TeacherName ==
                  name).ToList<Course>();
162         }
163 
164         public IList<Student> GetStudentsByStandardID(int id)
165         {
166             return studentRepository.SearchFor(c => c.StandardId ==
                  id).ToList<Student>();
167         }
168     }
169 }
170 
```