Enter here to filter...

Namespace SchoolReportSystem.main

Classes

SchoolReportApplication

This class is responsible for running the entire program.

Author: Yashwant Rathor

Class SchoolReportApplication

This class is responsible for running the entire program.

Author: Yashwant Rathor

Inheritance

- System.Object
  - SchoolReportApplication

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.main

Assembly: SchoolReportSystem.dll

Syntax

```
public class SchoolReportApplication
```

Methods

Main(String[])

This method runs the entire program.

Declaration

```
public static void Main(string[] args)
```

Parameters

Type	Name	Description
System.String[]	*args*	

Namespace SchoolReportSystem.controller

Classes

ReportBuilder

This class is responsible for creating instances of the `School`, `DataClass`, `ImportData` and `ExportData` classes.

Author: Yashwant Rathor

Class ReportBuilder

This class is responsible for creating instances of the `School`, `DataClass`, `ImportData` and `ExportData` classes.

Author: Yashwant Rathor

Inheritance

- System.Object
  - ReportBuilder

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.controller

Assembly: SchoolReportSystem.dll

Syntax

```
public class ReportBuilder
```

Constructors

ReportBuilder()

The default constructor responsible for creating a ReportBuilder object.

Declaration

```
public ReportBuilder()
```

Methods

GenerateReports()

This method effectively executes both import/export main methods for the system.

Declaration

```
public void GenerateReports()
```

Namespace SchoolReportSystem.model.classes

Classes

AcademicYear

This class defines a `AcademicYear` object.

Implements IEnumerable to allow Student objects inside the AcademicYear object to be iterated through. Implements IComparable to allow AcademicYear objects to be compared.

Author: Yashwant Rathor

School

This class defines a `School` object.

Implements IEnumerable to allow AcademicYear objects inside the School object to be iterated through.

Author: Yashwant Rathor

Student

This class defines a `Student` object.

Implements IComparable to allow Student objects to be compared.

Author: Yashwant Rathor

Subject

This class defines a `Subject` object.

Implements IComparable to allow Subject objects to be compared.

Author: Yashwant Rathor

Class School

This class defines a `School` object.

Implements IEnumerable to allow AcademicYear objects inside the School object to be iterated through.

Author: Yashwant Rathor

Inheritance

- System.Object
  - School

Implements

System.Collections.IEnumerable

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: SchoolReportSystem.model.classes

Assembly: SchoolReportSystem.dll

Syntax

```
public class School : IEnumerable
```

Constructors

School(String, Int32)

This custom constructor is responsible for creating a School object.

Declaration

```
public School(string name, int year)
```

Parameters

Type	Name	Description
System.String	*name*	The school name.
System.Int32	*year*	The year number.

Examples

```
School sl = new School("Portsborough Secondary", 8);
```

This creates a School object with the name as "Portsborough Secondary" and year as '8'.

School(String, Int32, Int32)

This alternate custom constructor is responsible for creating a School object.

Declaration

```
public School(string name, int first, int last)
```

Parameters

Type	Name	Description
System.String	*name*	The school name.
System.Int32	*first*	The lowest school year.
System.Int32	*last*	The highest school year

Examples

```
School sl = new School("Earlmount High", 7, 11);
```

This creates a School object with name as "Earlmount High", with years 7 to 11.

Properties

GetSchoolName

This method retrieves the school's name.

Declaration

```
public string GetSchoolName { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetSchoolName.

Examples

```
School sl = new School("Earlmount High", 7, 11);
sl.GetSchoolName;
```

This will return "Earlmount High".

GetYears

This method retrieves the school years.

Declaration

```
public List<AcademicYear> GetYears { get; }
```

Property Value

Type	Description
System.Collections.Generic.List<AcademicYear>	Returns the value of GetYears.

Examples

```
School sl = new School("Earlmount High", 7, 11);
sl.GetYears;
```

This will return {(7), (8), (9), (10), (11), (12), (13)}.

Methods

AddYear(AcademicYear)

This method stores a valid AcademicYear object into the School object.

Declaration

```
public void AddYear(AcademicYear y)
```

Parameters

Type	Name	Description
AcademicYear	*y*	AcademicYear object to be added to the school.

Examples

```
School sl = new School("Nottingham Secondary", 13);
sl.AddYear(12);
```

Year 12 has been added to 'sl'.

AddYears(Int32, Int32)

This method stores a range of valid AcademicYear objects into the School object, dependent on the 'min' and 'max' parameters.

Declaration

```
public void AddYears(int min, int max)
```

Parameters

Type	Name	Description
System.Int32	*min*	The minimum inclusive number value for the range.
System.Int32	*max*	The maximum inclusive number value for the range.

Examples

```
School sl = new School("Berkshire High", 7, 8);
sl.AddYears(9, 13);
```

Years: 9, 10, 11, 12 and 13 have been added to 'sl'.

GetEnumerator()

This method returns the IEnumerator for the School class.

Declaration

```
public IEnumerator GetEnumerator()
```

Returns

Type	Description
System.Collections.IEnumerator	Returns the iteration of the non-generic collection for a School object.

GetYearByNo(Int32)

This method retrieves the AcademicYear object which is equal to 'yearNo'.

Declaration

```
public AcademicYear GetYearByNo(int yearNo)
```

Parameters

Type	Name	Description
System.Int32	*yearNo*	

Returns

Type	Description
AcademicYear	Returns the year with the matching year number specifed by the 'yearNo' parameter.

Examples

```
School sl = new School("Midwest College", 9, 11);
sl.AddYears(12, 13);
```

sl.GetYearByNo(13); Year 13 will be returned.

ToString()

This method overrides the default 'ToString()' representation of the School class.

Declaration

```
public override string ToString()
```

Returns

Type	Description
System.String	The string representation of the School object.

Overrides

System.Object.ToString()

Examples

```
School sl = new School("Holloway High", 7, 13);
sl.ToString();
```

Implements

System.Collections.IEnumerable

Class AcademicYear

This class defines a `AcademicYear` object.

Implements IEnumerable to allow Student objects inside the AcademicYear object to be iterated through. Implements IComparable to allow AcademicYear objects to be compared.

Author: Yashwant Rathor

Inheritance

- System.Object
  - AcademicYear

Implements

System.Collections.IEnumerable
System.IComparable<AcademicYear>

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: SchoolReportSystem.model.classes

Assembly: SchoolReportSystem.dll

Syntax

```
public class AcademicYear : IEnumerable, IComparable<AcademicYear>
```

Constructors

AcademicYear(Int32)

This custom constructor is responsible for creating an AcademicYear object.

Declaration

```
public AcademicYear(int year)
```

Parameters

Type	Name	Description
System.Int32	*year*	The school year number.

Examples

```
AcademicYear yr = new AcademicYear(7);
```

This creates an AcademicYear object with the year as '7'.

Properties

GetStudents

This method retrieves all students within the year.

Declaration

```
public List<Student> GetStudents { get; }
```

Property Value

Type	Description
System.Collections.Generic.List<Student>	Returns the value of GetStudents.

Examples

```
AcademicYear yr = new AcademicYear(13);
yr.AddStudentToYear(new Student("8156", "Colin", "Jacobs", 13, "Lilac"));
yr.AddStudentToYear(new Student("8601", "Harris", "Ford", 13, "Rose"));
yr.AddStudentToYear(new Student("9509", "Scott", "Parker", 13, "Gold"));
yr.GetStudents;
```

This will return {("8156", "Colin", "Jacobs", 13, "Lilac"), ("8601", "Harris", "Ford", 13, "Rose"), ("9509", "Scott", "Parker", 13, "Gold")}.

GetYearNo

This method retrieves the year's number.

Declaration

```
public int GetYearNo { get; }
```

Property Value

Type	Description
System.Int32	Returns the value of GetYearNo.

Examples

```
AcademicYear yr = new AcademicYear(12);
yr.GetYearNo;
```

This will return '12'.

Methods

AddStudentToYear(Student)

This method stores a valid Student object into the AcademicYear object.

Declaration

```
public void AddStudentToYear(Student s)
```

Parameters

Type	Name	Description
Student	*s*	Student object to be added to the year.

Examples

```
AcademicYear yr = new AcademicYear(12);
yr.AddStudentToYear(new Student("8126", "Edward", "Willis", 12, "Red"));
```

A new student has been added to 'yr'.

CompareTo(AcademicYear)

This method compares one AcademicYear object with another AcademicYear object.

Declaration

```
public int CompareTo(AcademicYear other)
```

Parameters

Type	Name	Description
AcademicYear	*other*	A different AcademicYear object.

Returns

Type	Description
System.Int32	Returns 0 if the objects are the same and -1 or 1 if they are different.

Examples

```
AcademicYear yr = new AcademicYear(7);
AcademicYear yr2 = new AcademicYear(7);
yr.CompareTo(yr2);
```

This returns `0`, as the objects have the exact same year number.

GetEnumerator()

This method returns the IEnumerator for the AcademicYear class.

Declaration

```
public IEnumerator GetEnumerator()
```

Returns

Type	Description
System.Collections.IEnumerator	returns the iteration of the non-generic collection for an AcademicYear object.

GetStudent(Int32)

This method retrieves the Student object located at a given index.

Declaration

```
public Student GetStudent(int index)
```

Parameters

Type	Name	Description
System.Int32	*index*	The index position of the desired Student object.

Returns

Type	Description
Student	Returns the student located at the index position specifed by the 'index' parameter.

Examples

```
AcademicYear yr = new AcademicYear(13);
yr.AddStudentToYear(new Student("8156", "Colin", "Jacobs", 13, "Lilac"));
yr.AddStudentToYear(new Student("8601", "Harris", "Ford", 13, "Rose"));
yr.AddStudentToYear(new Student("9509", "Scott", "Parker", 13, "Gold"));
yr.GetStudent(2);
```

The student at index position 2 (Scott Parker) will be returned.

GetStudentById(String)

This method retrieves the Student object matched by the 'ID' string.

Declaration

```
public Student GetStudentById(string ID)
```

Parameters

Type	Name	Description
System.String	*ID*	The ID string to be used to get the matching Student object.

Returns

Type	Description
Student	Returns the student with the matching ID specifed by the 'ID' parameter.

Examples

```
AcademicYear yr = new AcademicYear(7);
yr.AddStudentToYear(new Student("0796", "Christine", "Lively", 7, "Indigo"));
yr.AddStudentToYear(new Student("1472", "Jessica", "Rhodes", 7, "Magenta"));
yr.AddStudentToYear(new Student("1140", "Frank", "London", 7, "Cyan"));
yr.GetStudentByID("0796");
```

The student with the ID value "0796" (Christine Lively) will be returned.

RemoveDuplicateStudents()

This method removes removes any duplicate students inside the AcademicYear object.

Declaration

```
public void RemoveDuplicateStudents()
```

Examples

```
AcademicYear yr = new AcademicYear(9);
yr.AddStudentToYear(new Student("2950", "Wendy", "Foakes", 9, "Mauve"));
yr.AddStudentToYear(new Student("2950", "Wendy", "Foakes", 9, "Mauve"));
yr.AddStudentToYear(new Student("4125", "Stewart", "Richards", 9, "Teal"));
yr.RemoveStudentFromYear(1);
```

The second occurence of "Wendy Foakes" will be removed as it already exists once.

RemoveStudentFromYear(Int32)

This method removes a student at a particular index position, inside the AcademicYear object.

Declaration

```
public void RemoveStudentFromYear(int i)
```

Parameters

Type	Name	Description
System.Int32	*i*	

Examples

```
AcademicYear yr = new AcademicYear(10);
yr.AddStudentToYear(new Student("5003", "Chris", "Archer", 10, "Green"));
yr.AddStudentToYear(new Student("4500", "Hayden", "Collins", 10, "Red"));
yr.AddStudentToYear(new Student("5210", "James", "Holder", 10, "Blue"));
yr.RemoveStudentFromYear(1);
```

The student at index position 1 (Hayden Collins) will be removed.

ToString()

This method overrides the default 'ToString()' representation of the AcademicYear class.

Declaration

```
public override string ToString()
```

Returns

Type	Description
System.String	The string representation of the AcademicYear object.

Overrides

System.Object.ToString()

Examples

```
AcademicYear yr = new AcademicYear(10);
yr.ToString();
```

Implements

System.Collections.IEnumerable
System.IComparable<T>

Class Student

This class defines a `Student` object.

Implements IComparable to allow Student objects to be compared.

Author: Yashwant Rathor

Inheritance

- System.Object
  - Student

Implements

System.IComparable<Student>

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: SchoolReportSystem.model.classes

Assembly: SchoolReportSystem.dll

Syntax

```
public class Student : IComparable<Student>
```

Constructors

Student(String, String, String, Int32, String)

This custom constructor is responsible for creating a Student object.

Declaration

```
public Student(string ID, string f_name, string l_name, int year, string clName)
```

Parameters

Type	Name	Description
System.String	*ID*	A student's unique 4-digit ID number.
System.String	*f_name*	A student's first name.
System.String	*l_name*	A student's last name.
System.Int32	*year*	A student's current school year.
System.String	*clName*	A student's class name.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
```

This creates a Student object with the ID value as "0007", first name as "John", last name as "Smith", school year as '9' and class name as "Grey".

Properties

GetClassName

This method retrieves the student's current class name.

Declaration

```
public string GetClassName { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetClassName.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetClassName;
```

This will return "Grey".

GetFilePath

This method retrieves the unique filepath to be used to store the student's report.

Declaration

```
public string GetFilePath { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetFilePath.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetFilePath;
```

This will return "C:/Student Results/Year 9/John Smith.txt".

GetForename

This method retrieves the student's first name.

Declaration

```
public string GetForename { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetForename.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetForename;
```

This will return "John".

GetFullName

This method retrieves the student's full name.

Declaration

```
public string GetFullName { get; }
```

Property Value

Type	Description
System.String	Returns the concatenated value of GetForname and GetSurname

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetFullName;
```

This will return "John Smith".

GetStudentID

This method retrieves the student's ID.

Declaration

```
public string GetStudentID { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetStudentID.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetStudentID;
```

This will return "0007".

GetSubjects

This method retrieves all subjects that the student studies.

Declaration

```
public List<Subject> GetSubjects { get; }
```

Property Value

Type	Description
System.Collections.Generic.List<Subject>	Returns the value of GetSubjects.

Examples

```
Student sd = new Student("5451", "Dylan", "Thomas", 10, "Sapphire");
sd.AddSubjectToStudent(new Subject("FRE3", "French", "A", "A"));
sd.AddSubjectToStudent(new Subject("BIO3", "Biology", "C", "B"));
sd.AddSubjectToStudent(new Subject("PSY3", "Psychology", "B", "B"));
sd.GetSubjects;
```

This will return '{("FRE3", "French", "A", "A"), ("BIO3", "Biology", "C", "B"), ("PSY3", "Psychology", "B", "B")}'.

GetSurname

This method retrieves the student's last name.

Declaration

```
public string GetSurname { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetSurname.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetSurname;
```

This will return "Smith".

GetYear

This method retrieves the student's current school year.

Declaration

```
public int GetYear { get; }
```

Property Value

Type	Description
System.Int32	Returns the value of GetYear.

Examples

```
Student sd = new Student("0007", "John", "Smith", 9, "Grey");
sd.GetYear;
```

This will return '9'.

Methods

AddSubjectToStudent(Subject)

This method stores a valid Subject object into the Student object.

Declaration

```
public void AddSubjectToStudent(Subject s)
```

Parameters

Type	Name	Description
Subject	*s*	Subject object to be added to the student.

Examples

```
Student sd = new Student("0021", "Jane", "Harper", 12, "Navy");
sd.AddSubjectToStudent(new Subject("PSY4", "Psychology", "C", "A"));
```

A new subject has been added to 'sd'.

CompareTo(Student)

This method compares one Student object with another Student object.

Declaration

```
public int CompareTo(Student other)
```

Parameters

Type	Name	Description
Student	*other*	A different Student object.

Returns

Type	Description
System.Int32	Returns 0 if the objects are the same and -1 or 1 if they are different.

Examples

```
Student sd = new Student("2000", "Dean", "Ambrose", 8, "Crimson");
Student sd2 = new Student("2000", "Dean", "Ambrose", 8, "Crimson");
sd.CompareTo(sd2);
```

This returns `0`, as the objects have the exact same student ID.

GetSubject(Int32)

This method retrieves the Subject object located at a given index.

Declaration

```
public Subject GetSubject(int index)
```

Parameters

Type	Name	Description
System.Int32	*index*	The index position of the desired Subject object.

Returns

Type	Description
Subject	Returns the subject located at the index position specifed by the 'index' parameter.

Examples

```
Student sd = new Student("2000", "Dean", "Ambrose", 8, "Crimson");
sd.AddSubjectToStudent(new Subject("SOC4", "Sociology", "A", "B"));
sd.AddSubjectToStudent(new Subject("GMN4", "German", "B", "B"));
sd.AddSubjectToStudent(new Subject("MAT4", "Maths", "A", "A*"));
sd.GetSubject(0);
```

The subject at index position 0 (Sociology) will be returned.

GetTargetCount(String)

This method retrieves the total number of subjects that the Student exceeds, meets or falls belows their expected grades.

Declaration

```
public int GetTargetCount(string type)
```

Parameters

Type	Name	Description
System.String	*type*	

Returns

Type	Description
System.Int32	The total number of subjects that satisfy the 'type' string parameter are returned.

Examples

```
Student sd = new Student("3650", "Oliver", "Ricardo", 9, "Copper");
sd.AddSubjectToStudent(new Subject("MAT3", "Mathematics", "A*", "B"));
sd.AddSubjectToStudent(new Subject("BIO3", "Biology", "B", "B"));
sd.AddSubjectToStudent(new Subject("CHM3", "Chemistry", "A", "B"));
sd.AddSubjectToStudent(new Subject("PHY3", "Physics", "C", "B"));
sd.AddSubjectToStudent(new Subject("ENG3", "English", "B", "A"));
```

```
sd.GetTargetCount("above");
```

This will return '2'.

```
sd.GetTargetCount("equal");
```

This will return '1'.

```
sd.GetTargetCount("below");
```

This will return '2'.

RemoveDuplicateSubjects()

This method removes any duplicate subjects inside the Student object.

Declaration

```
public void RemoveDuplicateSubjects()
```

Examples

```
Student sd = new Student("5001", "Tom", "Curry", 10, "Purple");
sd.AddSubjectToStudent(new Subject("GEO3", "Geography", "A", "B"));
sd.AddSubjectToStudent(new Subject("HIS3", "History", "B", "A*"));
sd.AddSubjectToStudent(new Subject("GEO3", "Geography", "A", "B"));
sd.RemoveDuplicateSubjects();
```

The second occurence of 'Geography' will be removed as it already exists once.

RemoveSubject(Int32)

This method removes a subject at a particular index position, inside the Student object.

Declaration

```
public void RemoveSubject(int i)
```

Parameters

Type	Name	Description
System.Int32	*i*	The index where the subject is to be removed from.

Examples

```
Student sd = new Student("2608", "Bradley", "Wiley", 13, "Bronze");
sd.AddSubjectToStudent(new Subject("FRE3", "French", "B", "A"));
sd.AddSubjectToStudent(new Subject("CHM3", "Chemistry", "A*", "A"));
sd.AddSubjectToStudent(new Subject("ENG3", "English", "C", "B"));
sd.RemoveSubject(2);
```

The subject at index position 2 (English) will be removed.

ToString()

This method overrides the default 'ToString()' representation of the Student class.

Declaration

```
public override string ToString()
```

Returns

Type	Description
System.String	The string representation of the Student object.

Overrides

System.Object.ToString()

Examples

```
Student sd = new Student("7142", "Ralph", "Dibney", 11, "Silver");
sd.ToString();
```

Implements

System.IComparable<T>

Class Subject

This class defines a `Subject` object.

Implements IComparable to allow Subject objects to be compared.

Author: Yashwant Rathor

Inheritance

- System.Object
  - Subject

Implements

System.IComparable<Subject>

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: SchoolReportSystem.model.classes

Assembly: SchoolReportSystem.dll

Syntax

```
public class Subject : IComparable<Subject>
```

Constructors

Subject(String, String, String, String)

This custom constructor method is responsible for creating a Subject object.

Declaration

```
public Subject(string ID, string name, string actualGrade, string expectedGrade)
```

Parameters

Type	Name	Description
System.String	*ID*	The ID of the subject.
System.String	*name*	The name of the subject.
System.String	*actualGrade*	The actual subject grade.
System.String	*expectedGrade*	The expected subject grade.

Examples

```
Subject sub = new Subject("PHY4", "Physics", "B", "A*");
```

This creates a Subject object with "PHY4" as the subject ID, "Physics" as the subject name, "B" as the actual grade and "A*" as the expected grade.

Properties

GetActualGrade

This method retrieves the subject's actual grade.

Declaration

```
public string GetActualGrade { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetActualGrade.

Examples

```
Subject sub = new Subject("PHY4", "Physics", "B", "A*");
sub.GetActualGrade;
```

This will return "B".

GetExpectedGrade

This method retrieves the subject's expected grade.

Declaration

```
public string GetExpectedGrade { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetExpectedGrade.

Examples

```
Subject sub = new Subject("PHY4", "Physics", "B", "A*");
sub.GetExpectedGrade;
```

This will return "A*".

GetSubjectID

This method retrieves the subject ID.

Declaration

```
public string GetSubjectID { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetSubjectID.

Examples

```
Subject sub = new Subject("PHY4", "Physics", "B", "A*");
sub.GetSubjectID;
```

This will return "PHY4".

GetSubjectName

This method retrieves the subject name.

Declaration

```
public string GetSubjectName { get; }
```

Property Value

Type	Description
System.String	Returns the value of GetSubjectName.

Examples

```
Subject sub = new Subject("PHY4", "Physics", "B", "A*");
sub.GetSubjectName;
```

This will return "Physics".

Methods

CompareTo(Subject)

This method compares one Subject object with another Subject object.

Declaration

```
public int CompareTo(Subject other)
```

Parameters

Type	Name	Description
Subject	*other*	A different Subject object.

Returns

Type	Description
System.Int32	Returns 0 if the objects are the same and -1 or 1 if they are different.

Examples

```
Subject sub = new Subject("MAT3", "Maths", "A", "B");
Subject sub2 = new Subject("MAT3", "Maths", "A", "B");
sub.CompareTo(sub2);
```

This returns `0`, as the objects have the exact same subject ID.

ToString()

This method overrides the default 'ToString()' representation of the Subject class.

Declaration

```
public override string ToString()
```

Returns

Type	Description
System.String	The string representation of the Subject object.

Overrides

System.Object.ToString()

Examples

```
Subject sub = new Subject("BIO3", "Biology", "C", "B");
sub.ToString();
```

Implements

System.IComparable<T>

Namespace SchoolReportSystem.model.validation

Classes

SchoolValidation

This important class contains utility methods that validate some of the constructor arguments for the objects created by the `School` class.

Author: Yashwant Rathor

StudentValidation

This important class contains utility methods that validate some of the constructor arguments for the objects created by the `Student` class.

Author: Yashwant Rathor

SubjectValidation

This important class contains utility methods that validate some of the constructor arguments for the objects created by the `Subject` class.

Author: Yashwant Rathor

Class SchoolValidation

This important class contains utility methods that validate some of the constructor arguments for the objects created by the School class.

Author: Yashwant Rathor

Inheritance

- System.Object
  - SchoolValidation

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.model.validation

Assembly: SchoolReportSystem.dll

Syntax

```
public static class SchoolValidation
```

Methods

IsSchoolNameValid(String)

This method checks if the name field of a School object is valid and it returns a tuple of boolean and string, which depends on the validity.

Declaration

```
public static (bool, string) IsSchoolNameValid(string name)
```

Parameters

Type	Name	Description
System.String	*name*	The school name to be validated.

Returns

Type	Description
System.ValueTuple<System.Boolean, System.String>	A tuple of a boolean true/false dependent on the validity of the name string and

Class StudentValidation

This important class contains utility methods that validate some of the constructor arguments for the objects created by the `Student` class.

Author: Yashwant Rathor

Inheritance

- System.Object
  - StudentValidation

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.model.validation

Assembly: SchoolReportSystem.dll

Syntax

```
public static class StudentValidation
```

Methods

CheckStudentIDRange(Int32, String)

This method returns true/false depending on the whether digits of the ID string are valid for the year.

Declaration

```
public static bool CheckStudentIDRange(int year, string studentID)
```

Parameters

Type	Name	Description
System.Int32	*year*	The student's academic year.
System.String	*studentID*	The student's ID.

Returns

Type	Description
System.Boolean	A boolean true/false is returned, which depends on the ID value being in range or not.

IsStudentIDValid(Int32, String)

This method checks if the ID field of a Student object is valid and returns a tuple of boolean and string, which depends on the validity.

Declaration

```
public static (bool, string) IsStudentIDValid(int year, string ID)
```

Parameters

Type	Name	Description
System.Int32	*year*	The student's academic year.
System.String	*ID*	The student's ID.

Returns

Type	Description
System.ValueTuple<System.Boolean, System.String>	A tuple of a boolean true/false dependent on the validity of the ID string and a h

IsStudentNameValid(String, String)

This method checks if the forename or surname fields of the Student object are valid; it returns a tuple of boolean and string, which depends on the validity.

Declaration

```
public static (bool, string) IsStudentNameValid(string name, string type)
```

Parameters

Type	Name	Description
System.String	*name*	The name string to be validated.
System.String	*type*	The type of name (forename/surname).

Returns

Type	Description
System.ValueTuple<System.Boolean, System.String>	A tuple of a boolean true/false dependent on the validity of the name string and

Class SubjectValidation

This important class contains utility methods that validate some of the constructor arguments for the objects created by the `Subject` class.

Author: Yashwant Rathor

Inheritance

- System.Object
  - SubjectValidation

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.model.validation

Assembly: SchoolReportSystem.dll

Syntax

```
public static class SubjectValidation
```

Methods

IsSubjectGradeValid(String, String)

This method checks if the subject grade is valid and returns a tuple of boolean and string, which depends on the validity.

Declaration

```
public static (bool, string) IsSubjectGradeValid(string grade, string type)
```

Parameters

Type	Name	Description
System.String	*grade*	The subject grade string to be validated.
System.String	*type*	The type of subject grade (actual/expected).

Returns

Type	Description
System.ValueTuple<System.Boolean, System.String>	A tuple of a boolean true/false dependent on the validity of the grade string and

IsSubjectIDValid(String)

This method checks if the ID field of a Subject object is valid and returns a tuple of boolean and string, which depends on the validity.

Declaration

```
public static (bool, string) IsSubjectIDValid(string ID)
```

Parameters

Type	Name	Description
System.String	*ID*	The subject ID to be validated.

Returns

Type	Description
System.ValueTuple<System.Boolean, System.String>	A tuple of a boolean true/false dependent on the validity of the subject ID and a

IsSubjectNameValid(String)

This method checks if the subject name is valid and returns a tuple of boolean and string, with values that depend on the validity of the subject name. As there a potential of different types of subject names depending on different conditions, each one is sectioned in a seperate part of the main IF statement.

Declaration

```
public static (bool, string) IsSubjectNameValid(string name)
```

Parameters

Type	Name	Description
System.String	*name*	The subject name to be validated.

Returns

Type	Description
System.ValueTuple<System.Boolean, System.String>	A tuple of a boolean true/false dependent on the validity of the subject name ar

Namespace SchoolReportSystem.view

Classes

DataClass

This class is responsible for creating instances of the `ImportData` and `ExportData` classes. It allows the two instances to be accessed by the `ReportBuilder` class for importing/exporting data to and from the system.

Author: Yashwant Rathor

Class DataClass

This class is responsible for creating instances of the ImportData and ExportData classes. It allows the two instances to be accessed by the ReportBuilder class for importing/exporting data to and from the system.

Author: Yashwant Rathor

Inheritance

- System.Object
  - DataClass

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.view

Assembly: SchoolReportSystem.dll

Syntax

```
public class DataClass
```

Constructors

DataClass()

The default constructor responsible for creating a DataClass object.

Declaration

```
public DataClass()
```

Properties

GetExportData

This method allows the ExportData class to be accessed.

Declaration

```
public ExportData GetExportData { get; }
```

Property Value

Type	Description
ExportData	Returns the instance of the ExportData class.

GetImportData

This method allows the ImportData class to be accessed.

Declaration

```
public ImportData GetImportData { get; }
```

Property Value

Type	Description
ImportData	Returns the instance of the ImportData class.

Namespace SchoolReportSystem.view.import

Classes

ImportData

This class is responsible for providing a main method which executes both main methods from the `ImportStudentData` and `ImportSubjectData` classes.

Author: Yashwant Rathor

ImportStudentData

This class is responsible for providing a main method, which allows `Student` data to be read from a raw text file and stored in the `model`.

Author: Yashwant Rathor

ImportSubjectData

This class is responsible for providing a main method, which allows `Subject` data to be read from a raw text file and stored in the `model`.

Author: Yashwant Rathor

Class ImportData

This class is responsible for providing a main method which executes both main methods from the ImportStudentData and ImportSubjectData classes.

Author: Yashwant Rathor

Inheritance

- System.Object
  - ImportData

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.view.import

Assembly: SchoolReportSystem.dll

Syntax

```
public class ImportData
```

Constructors

ImportData()

The default constructor responsible for creating a DataClass object.

Declaration

```
public ImportData()
```

Methods

LoadAllData(School)

This method is responsible for loading all student and subject data into the model classes.

Declaration

```
public void LoadAllData(School school)
```

Parameters

Type	Name	Description
School	*school*	The School object to have all data stored inside it.

Class ImportStudentData

This class is responsible for providing a main method, which allows `Student` data to be read from a raw text file and stored in the `model`.

Author: Yashwant Rathor

Inheritance

- System.Object
  - ImportStudentData

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.view.import

Assembly: SchoolReportSystem.dll

Syntax

```
public class ImportStudentData
```

Methods

LoadStudentData(School)

This method effectively stores the raw student data in the appropriate model classes.

Declaration

```
public void LoadStudentData(School sl)
```

Parameters

Type	Name	Description
School	*sl*	The School object where the data will be stored to.

Class ImportSubjectData

This class is responsible for providing a main method, which allows `Subject` data to be read from a raw text file and stored in the `model`.

Author: Yashwant Rathor

Inheritance

- System.Object
  - ImportSubjectData

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.view.import

Assembly: SchoolReportSystem.dll

Syntax

```
public class ImportSubjectData
```

Methods

LoadSubjectData(AcademicYear)

This method effectively stores the raw subject data in the appropriate model classes.

Declaration

```
public void LoadSubjectData(AcademicYear ay)
```

Parameters

Type	Name	Description
AcademicYear	*ay*	The AcademicYear object where the data will be stored to.

Namespace SchoolReportSystem.view.export

Classes

ExportData

This class is responsible for providing a main method, which allows data to be read from the `model` and stored into reports (text files).

Author: Yashwant Rathor

Class ExportData

This class is responsible for providing a main method, which allows data to be read from the `model` and stored into reports (text files).

Author: Yashwant Rathor

Inheritance

- System.Object
  - ExportData

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.view.export

Assembly: SchoolReportSystem.dll

Syntax

```
public class ExportData
```

Methods

StoreReportData(School)

This method effectively stores student data from the model classes into individual reports.

Declaration

```
public void StoreReportData(School sl)
```

Parameters

Type	Name	Description
School	*sl*	The School object where the data will be retrieved from.

Namespace SchoolReportSystem.utility

Classes

ExportMethods

This class contains some useful utility methods used to support the `ExportData` class.

Author: Yashwant Rathor

HelperMethods

This class contains some useful utility methods that are used by other classes in the program.

Author: Yashwant Rathor

ImportMethods

This class contains some useful utility methods to support the `ImportData` class.

Author: Yashwant Rathor

Class HelperMethods

This class contains some useful utility methods that are used by other classes in the program.

Author: Yashwant Rathor

Inheritance

- System.Object
  - HelperMethods

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.utility

Assembly: SchoolReportSystem.dll

Syntax

```
public static class HelperMethods
```

Methods

CapitaliseFirstLetterOfEachWord(String)

This utility method converts all strings into capital case, with the starting letter of each word being converted to upper case.

Declaration

```
public static string CapitaliseFirstLetterOfEachWord(string s)
```

Parameters

Type	Name	Description
System.String	*s*	The string to have the first letter of each word capitalised.

Returns

Type	Description
System.String	The string has the first letter of each word capitalised; any leading/trailing whitespaces are removed.

CheckArray(String[], String)

This utility method checks whether an array has any elements that are null, empty or only consist of whitespaces.

Declaration

```
public static void CheckArray(string[] array, string error)
```

Parameters

Type	Name	Description
System.String[]	*array*	The array to have each element checked.
System.String	*error*	The error message to display if an exception needs to be thrown

DynamicSeparator(String, String)

This utility method returns a pattern of characters corresponding to the length of the supplied string.

Declaration

```
public static string DynamicSeparator(string s, string pattern)
```

Parameters

Type	Name	Description
System.String	*s*	The string which requires a pattern separator.
System.String	*pattern*	The string pattern to be produced for the dynamic seperator.

Returns

Type	Description
System.String	The dynamic seperator with the input pattern and the length equal to the input string.

Class ImportMethods

This class contains some useful utility methods to support the ImportData class.

Author: Yashwant Rathor

Inheritance

- System.Object
  - ImportMethods

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.utility

Assembly: SchoolReportSystem.dll

Syntax

```
public static class ImportMethods
```

Methods

CapitaliseAlphaNumeric(String)

This method returns a single 'alphanumeric' string into capital case by capitalising each letter.

Declaration

```
public static string CapitaliseAlphaNumeric(string s)
```

Parameters

Type	Name	Description
System.String	*s*	The alphanumeric string to have each letter capitalised.

Returns

Type	Description
System.String	The alphanumeric string has each letter capitalised and any leading/trailing whitespaces are removed.

CapitaliseFirstLetter(String, Boolean)

This method returns a single word string into capital case by capitalising the first letter of the word.

Declaration

```
public static string CapitaliseFirstLetter(string s, bool special)
```

Parameters

Type	Name	Description
System.String	*s*	The string to have the first letter capitalised.
System.Boolean	*special*	A true/false value dependant on whether there are special characters in the string or not.

Returns

Type	Description
System.String	The string has the first letter capitalised and any leading/trailing whitespaces are removed.

CheckNoSubjectsForYear(Int32)

This useful method returns the correct number of subjects for an academic year.

Declaration

```
public static int CheckNoSubjectsForYear(int year)
```

Parameters

Type	Name	Description
System.Int32	*year*	The school year.

Returns

Type	Description
System.Int32	The total number of subjects for that year.

CheckSubjectIDForStudent(Int32, String)

This method checks whether the subject ID is applicable for a student, dependant of what year they are in.

Declaration

```
public static bool CheckSubjectIDForStudent(int year, string subjectID)
```

Parameters

Type	Name	Description
System.Int32	*year*	The student's current year.
System.String	*subjectID*	The subject's ID.

Returns

Type	Description
System.Boolean	A boolean true/false is returned, which depends on the subject ID value being applicable to the student's year or not

ShortenSubjectName(String)

This method returns a shortened variant of the string representing the subject name.

Declaration

```
public static string ShortenSubjectName(string subjectName)
```

Parameters

Type	Name	Description
System.String	*subjectName*	The subject name to be shortened.

Returns

Type	Description
System.String	The subject name is abbreviated, depending on certain conditions.

StoreStudentDetails(AcademicYear, String[])

This method stores valid student data into a Student object, which is then stored within the AcademicYear object.

Declaration

```
public static void StoreStudentDetails(AcademicYear ay, string[] array)
```

Parameters

Type	Name	Description
AcademicYear	*ay*	The AcademicYear object to have a student added to it.
System.String[]	*array*	The array, representing items on a line of the raw data text file.

StoreSubjectDetails(Student, String[])

This method stores valid subject data into a Subject object, which is then stored within the Student object.

Declaration

```
public static void StoreSubjectDetails(Student sd, string[] array)
```

Parameters

Type	Name	Description
Student	*sd*	The Student object to have a subject added to it.
System.String[]	*array*	The array, representing items on a line of the raw data text file.

Class ExportMethods

This class contains some useful utility methods used to support the ExportData class.

Author: Yashwant Rathor

Inheritance

- System.Object
  - ExportMethods

Inherited Members

System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.ToString()

Namespace: SchoolReportSystem.utility

Assembly: SchoolReportSystem.dll

Syntax

```
public static class ExportMethods
```

Methods

AddSpaces(Int32)

This method simply returns a string of whitespaces with a quantity solely dependant on the int 'no' parameter.

Declaration

```
public static string AddSpaces(int no)
```

Parameters

Type	Name	Description
System.Int32	*no*	The number of whitespace characters required.

Returns

Type	Description
System.String	Whitespaces with a quantity equal to the 'no' parameter.

FixSpacing(List<Subject>, Subject, String)

This method returns a string of whitespaces that depend on the size of the field type string for a student.

Declaration

```
public static string FixSpacing(List<Subject> subjects, Subject s, string type)
```

Parameters

Type	Name	Description
System.Collections.Generic.List<Subject>	*subjects*	The list of subjects that need to be iterated through.
Subject	*s*	The current subject which requires one of its fields to be spaced out neatly with a
System.String	*type*	The type of string that needs to whitespaces inserted next to it.

Returns

Type	Description
System.String	The number of whitespaces required to space out the subject's field string.

GenerateOutput(School, AcademicYear)

This method executes the 'OuputInformation' method with differing parameters, depending on the year number of the AcademicYear object.

Declaration

```
public static void GenerateOutput(School sl, AcademicYear ay)
```

Parameters

Type	Name	Description
School	*sl*	The School object that needs to be outputted.
AcademicYear	*ay*	The AcademicYear object that will have student reports built from.

GetCurrentDate()

This method returns the current date in the 'DD/MM/YYYY' format.

Declaration

```
public static string GetCurrentDate()
```

Returns

Type	Description
System.String	The current date in the specified string format.

GetCurrentTime()

This method returns the current time in the 'HH:MM:SS' format.

Declaration

```
public static string GetCurrentTime()
```

Returns

Type	Description
System.String	The current time in the specified string format.

OutputFeedback(String, Int32, Int32, Int32)

This method returns a message dependent on the performance of the student in terms of their target grades for each subject.

Declaration

```
public static string OutputFeedback(string fullName, int above, int equal, int below)
```

Parameters

Type	Name	Description
System.String	*fullName*	The student's full name.
System.Int32	*above*	The number of subjects with grades that exceed the target
System.Int32	*equal*	The number of subjects with grades that meet the target.
System.Int32	*below*	The number of subjects with grades that lie below the target.

Returns

Type	Description
System.String	A message that represents feedback on the overall performance of a student, in relation to their target grades.

OutputInformation(School, AcademicYear, String)

This method writes the report data for each student to their own text file.

Declaration

```
public static void OutputInformation(School sl, AcademicYear ay, string qual)
```

Parameters

Type	Name	Description
School	*sl*	The School object that needs to be outputted.
AcademicYear	*ay*	The AcademicYear object that will have student reports built from.
System.String	*qual*	The current adademic qualification that the student is enrolled on.