```
using System;

namespace ValueTypeCaseStudy
{
  class Program
  {
    static void Main()
    {
    // =====================================
    // Exercise 1: Student Attendance
    // =====================================
    int totalClasses = 100;
    int attendedClasses = 83;

    double attendancePercentage =
      (double)attendedClasses / totalClasses * 100;

    int attendanceTruncated = (int)attendancePercentage;
    int attendanceRounded = (int)Math.Round(attendancePercentage);

    Console.WriteLine("Exercise 1");
    Console.WriteLine(attendancePercentage);
    Console.WriteLine(attendanceTruncated);
    Console.WriteLine(attendanceRounded);
    Console.WriteLine();
```

Data Types Used

- int is used for total classes and attended classes because attendance is counted in whole numbers.
- double is used for attendance percentage because percentages can have decimal values.
- int is used for final display because university rules require an integer.

Rounding vs Truncation

Truncation removes the decimal part directly.
Example: 83.9 → 83

Rounding follows math rules.
If the decimal is 0.5 or more, the value increases.
Example: 83.9 → 84

Impact

Truncation can reduce the attendance value and may make a student ineligible.
Rounding gives a fairer result.

```
    // =====================================
    // Exercise 2: Online Exam Results
    // =====================================
    int m1 = 78;
    int m2 = 85;
    int m3 = 92;

    double averageMarks = (m1 + m2 + m3) / 3.0;
    double averageRoundedTwoDecimals =
      Math.Round(averageMarks, 2);
```

```
int scholarshipEligibleAverage =
    (int)Math.Round(averageMarks);

Console.WriteLine("Exercise 2: Exam Results");
Console.WriteLine(averageRoundedTwoDecimals);
Console.WriteLine(scholarshipEligibleAverage);
Console.WriteLine();
```

Data Types Used

- int is used for subject marks because marks are whole numbers.
- double is used for the average because averages can have decimals.
- int is used for scholarship eligibility because only whole numbers are required.

Precision Loss

Precision loss happens when converting a decimal number into an integer.

Example:

Actual average: 84.99
After truncation: 84 (decimal part lost)
After rounding: 85 (more accurate)

Rounding reduces precision loss compared to truncation.

```
// ====================================
// Exercise 3: Library Fine
// ====================================
decimal finePerDay = 10.5m;
int overdueDays = 3;

decimal totalFine = finePerDay * overdueDays;
double fineForAnalytics = (double)totalFine;

Console.WriteLine("Exercise 3");
Console.WriteLine(totalFine);
Console.WriteLine(fineForAnalytics);
Console.WriteLine();
```

Data Types Used

- decimal is used for fine per day and total fine because money calculations require accuracy.
- int is used for overdue days because days are whole numbers.
- double is used for analytics logging where small precision differences are acceptable.

Conversion Explanation

The total fine is calculated in decimal and then explicitly converted to double for analytics.

```
// ====================================
// Exercise 4: Banking Interest
// ====================================
 decimal accountBalance = 20000m;
 float annualInterestRate = 5f;
```

```
  decimal monthlyInterest =
    accountBalance * (decimal)annualInterestRate / 100;

  accountBalance =
    accountBalance + monthlyInterest;

  Console.WriteLine("Exercise 4: Bank Interest");
  Console.WriteLine(accountBalance);
  Console.WriteLine();
```

Data Types Used

- decimal is used for account balance because it represents money.
- float is used for interest rate because it comes from an external API.

Safe Conversion

C# does not allow direct calculation between decimal and float.
So the interest rate is explicitly converted to decimal.

Why Implicit Conversion Fails

Implicit conversion fails because decimal is more precise than float.
Explicit conversion prevents calculation errors.

```
  // ====================================
  // Exercise 5: E-Commerce Pricing
  // ====================================
  double cartTotal = 999.99;
  decimal taxRate = 0.18m;
  decimal discount = 100m;

  decimal finalAmount = (decimal)cartTotal;
  finalAmount = finalAmount + (finalAmount * taxRate);
  finalAmount = finalAmount - discount;

  Console.WriteLine("Exercise 5");
  Console.WriteLine(finalAmount);
  Console.WriteLine();
```

Data Types Used

- double is used for cart total because it is accumulated from multiple calculations.
- decimal is used for tax, discount, and final payable amount because financial values require high precision.

Conversion Strategy

The cart total is explicitly converted from double to decimal before applying tax and discount.

Precision Risk

Using double for money can cause rounding errors.
Using decimal avoids these errors.

```
  // ====================================
  // Exercise 6: Weather Monitoring
```

```
// =====================================
short sensorValue = 300;

double temperatureCelsius = sensorValue - 273;
int displayTemperature = (int)Math.Round(temperatureCelsius);

Console.WriteLine("Exercise 6");
Console.WriteLine(displayTemperature);
Console.WriteLine();
```

Data Types Used

- short is used for sensor readings because sensor values are small.
- double is used to store temperature in Celsius for accurate calculation.
- int is used for dashboard display because only whole numbers are needed.

Overflow and Casting

Explicit casting is used when converting double to int.
Rounding is applied to avoid incorrect display values.

```
// =====================================
// Exercise 7: University Grading
// =====================================
double finalScore = 82.5;
byte grade;

if (finalScore >= 80)
   grade = 1;
else
   grade = 2;

Console.WriteLine("Exercise 7");
Console.WriteLine(grade);
Console.WriteLine();
```

Data Types Used

- double is used for final score because it can contain decimals.
- byte is used for grades because grades have small numeric values.

Validation and Casting

The score is checked using conditions before assigning a grade.
Direct casting from double to byte is avoided to prevent invalid values.

```
// =====================================
// Exercise 8: Mobile Data Usage
// =====================================
long dataUsageInBytes = 5368709120;

// convert bytes to MB
double dataUsageInMB = dataUsageInBytes / (1024.0 * 1024);

// convert bytes to GB
double dataUsageInGB = dataUsageInBytes / (1024.0 * 1024 * 1024);
```

```
 // round monthly usage to nearest integer
 int roundedMonthlyUsage = (int)Math.Round(dataUsageInGB);

 Console.WriteLine("Exercise 8: Mobile Data Usage");
 Console.WriteLine(dataUsageInMB);
 Console.WriteLine(dataUsageInGB);
 Console.WriteLine(roundedMonthlyUsage);
 Console.WriteLine();
```

Data Types Used

- long is used to store data usage in bytes because the value can be very large.
- double is used to display usage in MB and GB.
- int is used for monthly summary after rounding.

Implicit Conversion and Rounding

Bytes are converted to MB and GB using division.
The monthly summary is rounded to the nearest integer for easy understanding.

```
 // ====================================
 // Exercise 9: Warehouse Inventory
 // ====================================
 int itemCount = 450;
 ushort maxCapacity = 500;

 bool withinLimit = itemCount <= maxCapacity;

 Console.WriteLine("Exercise 9");
 Console.WriteLine(withinLimit);
 Console.WriteLine();
```

Data Types Used

- int is used for current item count.
- ushort is used for maximum capacity because capacity cannot be negative.

Signed vs Unsigned Risk

Comparing signed (int) and unsigned (ushort) values must be done carefully.
Using correct types avoids overflow and comparison errors.

```
 // ====================================
 // Exercise 10: Payroll Salary
 // ====================================
 int basicSalary = 25000;
 double allowance = 5000.75;
 double deduction = 1200.50;

 decimal netSalary = basicSalary;
 netSalary = netSalary + (decimal)allowance;
 netSalary = netSalary - (decimal)deduction;

 Console.WriteLine("Exercise 10");
 Console.WriteLine(netSalary);
 }
}
```

```
}
```

Data Types Used

- int is used for basic salary.
- double is used for allowances and deductions because they may contain decimals.
- decimal is used for net salary to ensure accurate money calculation.

Conversion Flow

The basic salary is converted to decimal.
Allowances and deductions are explicitly converted before calculation.

Justification

Using decimal for final salary prevents precision loss and rounding issues.