



ST PETER'S
COLLEGE

**SUBJECT: INFORMATION TECHNOLOGY
PRELIMINARY EXAMINATION : PAPER I**

GRADE: 12
MARKS: 150
TIME: 3 HOURS

DATE: 22 JULY 2025
EXAMINERS: J. NOCTON-SMITH
S. RONNE
MODERATOR: IT CLUSTER
EXAM PROFILE: Exam 0__

FULL NAME: _____

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY:

1. Log into the exam profile allocated to you.
2. This question paper is to be answered using Object-Oriented Programming principles. Your program must make sensible use of methods and parameters.
3. This paper is divided into two sections and consists of 14 pages. You must answer both sections.
4. This paper is set in programming terms that are not specific to any programming language (Delphi/Java) or database (Access/MySQL).
5. Make sure that you answer the questions in the manner described because marks will be awarded for your solution according to the specifications that are given in the question. Only answer what is asked in each question. For example, if the question does not ask for data validation, then no marks are awarded for it, and therefore no code needs to be written.
6. If you cannot get a section of code to work, comment it out so that it will not be executed and so that you can continue with the examination. If possible, try to explain the error to aid the marker.
7. Make sure that routines such as searches, sorts and selections are to be developed from first principles, and that you do not use the built-in features of a programming language for any of these routines.
8. All data structures must be defined and declared by you, the programmer. You may not use components provided within the interface to store and later retrieve data.
9. Read the whole question paper before you choose a data structure. You may find that there could be an alternative method of representing the data which will be more efficient considering the questions that are asked in the paper.
10. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination.
11. If there is a technical interruption that prevents you from writing your examination, e.g., a power failure, when you resume writing your examinations, you will only be given the time that was remaining when the interruption began. No extra time will be given to catch up on work that was not saved.
12. Make sure that your name appears as a comment in every unit that you code as well as on every page of hard copy that you hand in.
13. Print a code listing of all the programs/classes that you code. Printing must be done after the examination. You will be given half an hour to print after the examination is finished. Your teacher will tell you what arrangements have been made for the printing of your work.
14. You should have the following two folders (in bold) and files. These files are to be used as data for the examination.

Section A:

- RailwayDB.accdb
- SQLAnswerSheet.rtf

Section B:

- RailData.txt

15. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper. You are advised to look at the supplied data files carefully.

QUESTION	1	2	3	4	5	6	7	TOTAL	%
MARKS ALLOCATED	50	13	15	29	4	15	24	150	
MARKS AWARDED									

SECTION A

SQL

50 MARKS

SCENARIO:

Continental Railways operates a fleet of high-speed trains that travel across various routes in multiple countries. To manage operations, they maintain a relational database called RailwayDB. This database stores information about the trains, the routes they operate on, and customer bookings for those routes.

The company uses the data to monitor performance, calculate route income, generate timetables, and prepare reports for management. Some trains operate internationally, while others are restricted to domestic routes. Bookings are tracked for each travel date and linked to the appropriate route.

The database contains the three tables, namely **tblTrains**, **tblRoutes** and **tblBookings**:

tblTrains — stores details about each train, such as its name, maximum speed, and whether it operates internationally.

Design View:

tblTrains		
Field Name	Data Type	Description (Optional)
TrainID	Number	A unique auto-numbered ID number for each train
TrainName	Short Text	The name of the train
Country	Short Text	The country where the train operates
MaxSpeed	Number	The maximum train speed in km/hour
International	Yes/No	Whether the train operates in other countries

A sample of data:

tblTrains					
TrainID	TrainName	Country	MaxSpeed	International	
1	LimeGreen Sho	Saudi Arabia	315	☒	
2	DarkGreen You	Chile	270	☒	
3	MistyRose Leg	Holy See (Vatic	261	☒	
4	SeaShell Past	Syrian Arab Rep	274	☒	
5	SlateGray Glass	Germany	346	☐	

tblRoutes — contains information about origin, destination, distance, duration, and cost for each route. Each route is assigned to a train.

Design View:

tblRoutes		
Field Name	Data Type	Description (Optional)
RouteID	Number	A unique auto-numbered identification number for each route
TrainID	Number	The ID of the train operating on this route. This is a foreign key to tblTrains
Origin	Short Text	The station where the route originates/starts
Destination	Short Text	The final destinatio station on this route
DistanceKM	Number	The distance in kilometers
DurationMin	Number	The duration in minutes stored as an integer value
Cost	Currency	The cost stores the price of a one-way ticket for the route, in South African Rand

A sample of data:

RouteID	TrainID	Origin	Destination	DistanceKM	DurationMin	Cost
1	38	Jasonton	Faithchester	984	180	R761.00
2	68	Harrisbury	South Rogerbu	1941	340	R571.00
3	76	New Austinbur	Brendastad	1718	300	R1,743.00
4	17	Port Adamville	West Aaron	1141	260	R954.00
5	43	Amvport	North Jeffrev	1462	350	R1,275.00

tblBookings — tracks customer bookings, linking each passenger to a route and recording the travel date.

Design View:

tblBookings X		
Field Name	Data Type	Description (Optional)
BookingID	Number	A unique auto-numbered identification number for each booking
RouteID	Number	The ID of the route. This is a foreign key to tblRoutes
PassengerName	Short Text	The full name of the passenger
TravelDate	Date/Time	The travel date of the booking

A sample of data in tblBookings:

tblBookings X				
BookingID	RouteID	PassengerName	TravelDate	Click to Add
1	40	Charles Wilson	18/05/2025	
2	15	Stanley Hayes	13/07/2024	
3	91	Taylor Martin	15/03/2025	
4	82	Michelle Stone	29/01/2024	
5	80	Laurie Tate	27/10/2025	

QUESTION 1

Note: Do not rename columns/use an alias unless explicitly told to do so in a question.

- 1.1
- Display the details of all trains operating in Germany that are international trains.
The correct output is shown below:

TrainID	TrainName	Country	MaxSpeed	International
48	Orchid Begin	Germany	324	☒
64	Purple Image	Germany	303	☒

(4)

- 1.2
- Display the origin , destination, and distance for the 3 longest routes.
The correct output is shown below:

Origin	Destination	DistanceKM
Piercefurt	Pittmanland	2497
West Dannyshi	Maymouth	2489
Robbinsfort	Thomastown	2468

(3)

- 1.3 Display the train name and cost for all routes where the cost is greater than the average cost of all routes. Only display trains with “blue” anywhere in the names. A sample of output is below:

TrainName	Cost
PowderBlue Half	R1,615.00
MediumBlue Hot	R1,697.00
MediumBlue Hot	R1,320.00
MediumSlateBlue	R1,554.00
LightSteelBlue Scer	R1,906.00
DodgerBlue Everyc	R1,496.00
MediumBlue Billio	R1,636.00
SkyBlue Area	R1,613.00
SkyBlue Drop	R1,501.00

(6)

- 1.4 Display the average MaxSpeed for international trains for each country. Only show the countries where the average MaxSpeed is greater than 300. The correct output is shown below:

Country	AVG MaxSpe
Australia	350
Cayman Islands	349
Antigua and Barbuda	349
Ethiopia	347
Norfolk Island	341
Bangladesh	336
Suriname	335
Turkey	331
Korea	331
Taiwan	329
Comoros	329
Senegal	327.5
Uzbekistan	327
Vietnam	327
Madagascar	325
Saudi Arabia	315
Marshall Islands	314
Germany	313.5
Norway	306

(7)

- 1.5 Some of the durations in the table are unrealistic when compared to the train maximum speed and route distance. You will create a query to calculate the time in minutes it would take a train to complete its route if it were to travel at the given maximum speed for the whole route.

The formula to calculate the duration in minutes is

$$\text{duration} = \text{DistanceKM}/\text{MaxSpeed} * 60$$

This value must be rounded to 0 decimal places.

Display the name of the train, the given duration, and the calculated duration.

Only show the records where the calculated duration exceeds the duration for that route.

The correct output is shown below:

TrainName	DurationMin	Calculated D
Magenta System	150	151
Cyan Work	170	171
IndianRed Heat	80	82
IndianRed Work	80	81
Lime Article	140	142
Snow Near	190	191
Gainsboro Us	120	121

(7)

- 1.6 Display the origin and destination for all routes that have not been booked. A sample of only the first 10 records as output is displayed below:

Origin	Destination
New Austinbur	Brendastad
Amyport	North Jeffrey
Piercefurt	Pittmanland
Fullerchester	Crystalberg
North Vanessa	Thompsonland
Port Noah	North Samanth
West Dannyshi	Maymouth
Mendezchester	South Johnview
East Kimberly	North Carlaton
East Tonifort	Amyton

(4)

- 1.7 Display the passenger's name, travel date and the days since the travel date for all bookings that are older than six months (you may assume 180 days for 6 months).

The days must remain accurate and must not be rounded up but truncated instead. Order the data by travel date.

The sample output is shown below as at 21/06/2025. Your days calculated will differ. Only the first 10 rows are shown.

PassengerName ▾	TravelDate ▾	DAYS ▾
Kelly Garrett	09/01/2024	529
Jessica Petersen	17/01/2024	521
Travis Carr	21/01/2024	517
Julia Parker	22/01/2024	516
Cheryl Edwards	25/01/2024	513
Michelle Stone	29/01/2024	509
Connor Ramos	04/02/2024	503
Sheila Silva	10/02/2024	497
Jay Parker	11/02/2024	496
Jennifer King	13/02/2024	494

(7)

- 1.8 Display the train name, origin, destination, and travel date for all bookings for June and July 2025. Combine the origin and destination into one field labelled route. The origin and destination must be separated by a '-.' The correct output is shown below.

TrainName ▾	Route ▾	TravelDate ▾
Bisque Word	Jasonton - Faithchester	14/07/2025
HoneyDew Evening	Port Jennifer - Ballardshire	22/07/2025
SeaGreen Medical	Lake Harryborough - North Steve	04/07/2025
FireBrick Economy	West Greggbury - Hollandhaven	12/07/2025
IndianRed Word	West Timothyside - East Seanborough	20/07/2025
DarkKhaki Significant	Seanport - Thompsonburgh	02/07/2025
IndianRed Word	West Timothyside - East Seanborough	11/07/2025
MediumBlue Billion	South Amanda - Jennifermouth	12/06/2025
IndianRed Word	South Tonyaside - Marcusfurt	16/07/2025
SpringGreen Never	New Deanna - Derrickton	24/07/2025
DarkGreen Your	Angelland - West Amandafurt	03/07/2025
Wheat Black	West Ryan - Maryport	19/06/2025

(9)

- 1.9 Create a query to increase the cost of all routes longer than 800km by R 50.

(3)

SUBTOTAL: 50 MARKS

SECTION B**OBJECT-ORIENTED PROGRAMMING****100 MARKS****SCENARIO**

Continental Railways is a company that manages a network of high-speed trains and routes across several countries. The company maintains detailed records about its fleet of trains, the routes they operate, and key route attributes such as distance, duration, and origin-destination cities.

To ensure operational efficiency and customer satisfaction, Continental Railways requires a system that can:

- Store and retrieve information about trains and their assigned routes
- Generate detailed reports on current train and route data
- Identify and classify express routes
- Modify route details when changes occur (e.g. duration adjustments)
- Generate updated timetables and route summaries after modifications

The system should use object-oriented principles, allow data to be read from an external text file, and support common operations such as filtering, searching, and statistical reporting.

The program will make use of the following classes.

Train Class

This class will represent the basic details of the trains

- **name** - the name of the train stored as a string
- **country** - the country where the train operates stored as a string
- **maxSpeed** – the maximum speed of the train in kilometres per hour
- **international** – a value indicating if the train travels between countries. This is stored as a boolean value.

The details of the trains are stored in the first 13 lines of a text file called **RailData.txt**. A line of the file representing the data of a train has the following format.

<name>;<country>;<maxspeed>;<international>

For example, the first line is:

```
Silver Falcon;Germany;300;true
```

Silver Falcon is the name of a train that operates in Germany.
It has a maximum speed of 300 kilometres per hour and is an international train.

The first 13 lines of the text file are shown below:

```
Silver Falcon;Germany;300;true
Eastern Arrow;Japan;320;false
Desert Wind;UAE;280;true
La Bala;Spain;310;false
Atlantic Express;USA;330;true
Northern Light;Canada;305;true
Solar Runner;Italy;295;false
Eastern Spirit;China;310;true
Ocean Glide;UK;290;false
Kyle Express;South Africa;326;true
Brett Express;Canada;317;false
Meghan Express;Germany;342;false
Savannah Arrow;South Africa;315;true
```

Route Class

This class will represent the basic details of the routes.

- **name** - the name of the train operating on the route stored as a string
- **origin** - the name of the station where the route starts stored as a string
- **destination** – the name of the station where the route end stored as a string
- **distance** – the distance of the route in kilometres stored as an integer
- **duration** – the duration of the trip in minutes stored as an integer

The details of the routes are stored after the thirteen lines of train data in the file **RailData.txt**. A line of the file representing the data of a route has the following format:

<name>;<origin>;<destination>;<distance> ;<duration>

For example, the first line is:

```
Silver Falcon;Berlin;Paris;1050;290
```

The train operating on this route is Silver Falcon. The route starts in Berlin and ends in Paris. The distance covered is 1050 kilometres in 290 minutes.

The first 5 lines of Route data from the text file are shown below:

```
Silver Falcon;Berlin;Paris;1050;290
Desert Wind;Abu Dhabi;Riyadh;900;260
Eastern Arrow;Tokyo;Osaka;515;135
La Bala;Madrid;Barcelona;621;160
Atlantic Express;New York;Chicago;1145;300
```

NOTE: The file **RailData.txt** contains more records than those shown above.

QUESTION 2

Use the class diagram below to create a new class called **Route**. This class will be used to store the details of a route.

Route
- name : string - origin : string - destination : string - distance : integer - duration : integer + EXPRESS SPEED THRESHOLD = 300 : integer
+ Constructor (inName : string, inOrigin : string, inDestination : string, inDistance : integer, inDuration : integer) + getName() : string + getOrigin() : string + getDestination() : string + getDuration() : integer + adjustDuration(perc : real) + toString() : string

- 2.1

Create a class called **Route** with the fields **name**, **origin**, **destination**, **distance**, and **duration** using the data types in the class diagram. These fields must not be accessible outside the class.

(3)
- 2.2

Add the class constant **EXPRESS_SPEED_THRESHOLD** as shown in the class diagram.

(2)
- 2.3

Write code to create a parameterised constructor method that will accept parameters for all fields as shown in the class diagram and assign these values to corresponding fields.

(3)
- 2.4

Create accessor methods for the **name**, **origin**, **destination**, and **duration** fields.

(2)
- 2.5

It is sometimes necessary to modify the duration of a route. Create a method **adjustDuration** that accepts a percentage value as a real variable. This is used to increase the duration of the route by the percentage. Use the following formula:

$$duration = duration + duration * (perc/100.0)$$

The value must be rounded to the nearest integer value.

(3)
- 2.6

Create a **toString** method that will return a string that combines the values of the **route**, **origin**, and **destination** fields in the **Route** class. The format of the string should be as follows:

"Route: "**name**<tab>"Origin: "**origin**<tab>"Destination: "**destination**

For example :

Route: Desert Wind Origin: Abu Dhabi Destination: Riyadh

(2)

QUESTION 3

Use the class diagram below to create a new class called **Train**. This class will be used to store the details of a train. The diagram below indicates the fields and methods that are required.

Train
- name : string - country : string - maxSpeed : integer - international : boolean
+ Constructor (inName : string, inCountry : string, inSpeed :integer, inInternational : boolean) + getName() : string + getCountry() : string + getMaxSpeed() : integer + isExpress() : boolean + toString() : string

- 3.1

Create a class called **Train** with the fields **name**, **country**, **maxspeed** and **international** using the data types in the class diagram. These fields must not be accessible outside the class.

(3)
- 3.2

Write code to create a parameterised constructor method that will accept parameters for all **fields** as shown in the class diagram and assign these values to corresponding fields.

(3)
- 3.3

Create accessor methods for the **name**, **country** and **maxSpeed** fields.

(2)
- 3.4

A train is an express train if it exceeds the express speed threshold (see Q3.2) and it is an international train. Code a method called **isExpress**. This method should return whether the train is express or not.

(3)
- 3.5

Create a **toString** method that will return a string that combines the values of the **name**, **country** and **maxspeed** fields in the **Train** class. The format of the string should be as follows:

name<tab>country<tab>maxSpeed

(2)

For example,

Silver Falcon

Germany

300

QUESTION 4

- 4.1 Create a class called **RailwayManager**. (1)
- 4.2 Create four fields for the **RailwayManager** class as described below.
An array called **trainArr** to store up to 100 **train** objects.
A counter called **trainSize** to count the objects added to **trainArr**.
An array called **routeArr** to store up to 100 **route** objects.
A counter called **routeSize** to count the objects added to **routeArr**.
The fields should not be accessible outside the class. (3)
- 4.3 Create a constructor method that will read the contents of the text file **RailData.txt** containing information about the different trains and routes. The first 13 lines contain data about **Trains**. (You may hardcode the number 13 in your solution). The remaining lines contain data about **Routes**. The method should do the following.
- Check if the file **RailData.txt** exists.
 - Display a suitable error message if the file does not exist.
 - Open the file for reading.
 - Loop through the first 13 lines of the text file. In each iteration of the loop:
 - Read the line and split the data into separate parts.
 - Create a **Train** object and store the object in the next available position in the **trainArr** array.
 - Update the counter variable **trainSize**.
 - Loop through the remaining lines. In each iteration of the loop:
 - Read the line and split the data into separate parts.
 - Create a **Route** object and store the object in the next available position in the **routeArr** array.
 - Update the counter variable **routeSize**.
- (14)
- 4.4 Create code to create a **toString** method. This method should return a string with:
- The heading 'Trains'
 - Underneath the heading the details of all **Trains**, each on a new line.
 - A blank line.
 - The heading 'Routes'
 - Underneath the heading the details of all **Routes**, each on a new line as shown in Question 5.3.
- (6)
- 4.5 Create a method named **listExpressTrains**. This method should return the name of all express trains for a given country. The country will be passed as a parameter to the method. Use the country in the heading. (5)

For example:

```
Express Trains in Canada  
Northern Light
```

[29]

QUESTION 5

- 5.1 Write code to create a text-based user interface called **RailwayUI** that will allow simple input and output. (1)
- 5.2 Declare and instantiate a **RailwayManager** object. (1)
- 5.3 Write code to call the appropriate method in the **RailwayManager** class to display a list of all the **Train** and **Route** objects. The sample output is shown below. (1)

Trains

```
Silver Falcon    Germany 300
Eastern Arrow   Japan   320
Desert Wind      UAE     280
La Bala Spain    310
Atlantic Express      USA     330
Northern Light Canada 305
Solar Runner     Italy   295
Eastern Spirit China 310
Ocean Glide      UK      290
Kyle Express     South Africa 326
Brett Express    Canada 317
Meghan Express   Germany 342
Savannah Arrow South Africa 315
```

Routes

```
Route: Silver Falcon Origin: Berlin Destination: Paris
Route: Desert Wind Origin: Abu Dhabi Destination: Riyadh
Route: Eastern Arrow Origin: Tokyo Destination: Osaka
Route: La Bala Origin: Madrid Destination: Barcelona
Route: Atlantic Express Origin: New York Destination: Chicago
Route: Northern Light Origin: Toronto Destination: Vancouver
Route: Solar Runner Origin: Rome Destination: Naples
Route: Eastern Spirit Origin: Beijing Destination: Shanghai
Route: Ocean Glide Origin: London Destination: Manchester
Route: Savannah Arrow Origin: Johannesburg Destination: Durban
Route: Silver Falcon Origin: Munich Destination: Vienna
Route: Desert Wind Origin: Dubai Destination: Muscat
Route: Eastern Arrow Origin: Kyoto Destination: Hiroshima
Route: La Bala Origin: Valencia Destination: Seville
Route: Atlantic Express Origin: Houston Destination: Dallas
Route: Northern Light Origin: Montreal Destination: Calgary
Route: Solar Runner Origin: Venice Destination: Florence
Route: Eastern Spirit Origin: Wuhan Destination: Guangzhou
Route: Ocean Glide Origin: Bristol Destination: Glasgow
Route: Savannah Arrow Origin: Cape Town Destination: Port Elizabeth
```

- 5.4 Write code to call the appropriate method in the **RailwayManager** class to display all the express trains in Canada. The correct output is given below. (1)

```
Express Trains in Canada
Northern Light
```

[4]

QUESTION 6

6.1 In the **RailwayManager** class, code a helper method called **sortTrainsByCountry** to sort the **trainArr** by country. (6)

6.2 In the **RailwayManager** class , code a method **getCountrySummary** that will return a string listing each country and the number of trains for each country. See sample output below. (8)

Hint: you will need to make use of the sort method coded in question 6.1

```
Country Summary
Canada: 2
China: 1
Germany: 2
Italy: 1
Japan: 1
South Africa: 2
Spain: 1
UAE: 1
UK: 1
USA: 1
```

6.3 In the **RailwayUI** add code to call the **getCountrySummary** method and display the results. (1)
[15]

QUESTION 7

7.1 In the **RailwayManager** class, code a method **updateRouteDuration** to update the duration of all routes for a specific train by a given percentage. The method should accept the name of the **Train** and the percentage as parameters. The percentage is a real value. (4)

7.2 In the **RailwayManager** class, code a method **printTimeTable** that will print a timetable for a train for a specific week. The method must accept the train, origin, start date and time of the first train as parameters. The start date must be a **LocalDate** variable and the first train time a **LocalTime** variable. You can assume that the start date will always be a Monday.

The rules for the timetable are as follows:

Monday to Friday -

- Trains leave every 4 hours after the first train.
- There are no trains departing after 7pm (19:00)
- The arrival time is calculated using the duration.

The method must return the timetable as a string using the following format:

- **Heading**
"Timetable for "<TRAIN NAME> " Departing from "<ORIGIN >
- **Header for days of week**
"Mon: " <start date> "to Fri: " <start date + 5 days>
- **Train times**
"Depart:" <departure time> " Arrive: " <arrival time>

Example for

Train: Eastern Spirit

Origin: Wuhan

Start Date: 2025-06-16

First Train: 6:15

Timetable for EASTERN SPIRIT Departing from WUHAN

Mon: 2025-06-16 to Fri: 2025-06-21

Depart: 06:15 Arrive: 10:50

Depart: 10:15 Arrive: 14:50

Depart: 14:15 Arrive: 18:50

Depart: 18:15 Arrive: 22:50

(15)

7.3 In the **RailwayUI** add code to call the method **updateRouteDuration**. Use 'Eastern Spirit' as the train parameter and 10.0 as the percentage parameter. (1)

7.4 Create a **LocalDate** variable **startDate** and a **LocalTime** variable **firstTrain**. Initialise **startDate** to **2025-07-14** and **firstTrain** to **06:15** (2)

7.5 In the **RailwayUI** add code to call the method **printTimetable** and display the results.
Call the method using the parameter values "Eastern Spirit" for the train, "Wuhan" for the origin station and the variables **startDate** and **firstTrain** created in 7.4 (2)

[24]

SUBTOTAL SECTION B: 100 MARKS

GRAND TOTAL: 150 MARKS

END OF ASSESSMENT