

SA's Leading Past Year

Exam Paper Portal



You have Downloaded, yet Another Great
Resource to assist you with your Studies 😊

Thank You for Supporting SA Exam Papers

Your Leading Past Year Exam Paper Resource Portal

Visit us @ www.saexampapers.co.za



**SA EXAM
PAPERS**



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2014

MARKS: 150

TIME: 3 hours

This question paper consists of 22 pages.

INSTRUCTIONS AND INFORMATION

1. This paper is divided into THREE sections. Candidates must answer ALL THREE sections.
2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.
3. This paper is set in programming terms that are not specific to any particular programming language (Delphi/Java (using the Netbeans IDE)).
4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements only.
5. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.
6. 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.
7. Routines such as search, sort and selection must be developed from first principles. You may not use the built-in features of a programming language for any of these routines.
8. All data structures must be defined by you, the programmer. You may not use components provided within the interface to store and later retrieve data.
9. You must save your work regularly on the disk you have been given, or the disk space allocated to you for this examination session.
10. Make sure that your examination number appears as a comment in every program that you code as well as on every event indicated.
11. If required, print the programming code of all the programs/classes that you completed. You will be given half an hour printing time after the examination session.
12. At the end of this examination session you must hand in a disk/CD/DVD/flash disc with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

13. The files you need to complete this question paper have been given to you on a disk/CD/DVD/flash disk or the disk space allocated to you in the form of a password-protected executable file:

- Delphi learners must use the file **DelphiDataENG.exe**
- Java learners must use the file **JavaDataENG.exe**

Do the following:

- Double click on the file
- Click on the extract button
- Enter the following password: **Transport@(!\$**

List of files provided in the folder DelphiDataENG/JavaDataENG (once extracted):

Delphi files

Question1:

Question1_P.dpr
Question1_P.res
Question1_U.dfm
Question1_U.pas

Question2:

Delivery_U.pas
DeliveryInfo.txt
Question2_P.dpr
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question3:

Question3_P.dpr
Question3_P.res
Question3_U.dfm
Question3_U.pas

Java (Netbeans) files

Question1:

Question1.form
Question1.java

Question2:

Delivery.java
DeliveryInfo.txt
Question2.form
Question2.java

Question3:

Question3.form
Question3.java

SCENARIO:

SuperTrans Courier Services is a national transport company with branches throughout South Africa. You are requested to assist with some of the software applications the company intends to implement shortly.

SECTION A**QUESTION 1: GENERAL PROGRAMMING SKILLS****INSTRUCTIONS:**

Delphi programmers	Java programmers
<ul style="list-style-type: none"> The project Question1 is provided in the DelphiDataENG folder. Open the incomplete project file Question1_P.dpr in the Question1 folder. Add your examination number as a comment in the first line of the main form unit (Question1_U) file. 	<ul style="list-style-type: none"> The project Question1 is provided in the JavaDataENG folder. Open the incomplete class called Question1.java contained in the folders Source Packages (src), Question1Package. Add your examination number as a comment in the first line of the class (Question1).

Do the following:

- Compile and execute the program. The interface displays five different sections labelled Question 1.1 to Question 1.5. The program currently has no functionality. An example of the interface is given below:

The screenshot displays a software interface with five distinct sections, each with its own set of controls and labels:

- Question 1.1:** Contains two dropdown menus for 'Select the place of departure' and 'Select the destination', both currently set to 'Cape Town'. Below them is a text input field for 'Enter the number of kilometres'. At the bottom left is a 'Confirm delivery' button, and to its right is a 'Delivery label' text input field.
- Question 1.2:** Features a list box for 'Select the category' with options A1 (0kg - 1kg), A2 (>1kg - 2kg), A3 (>2kg - 5kg), and A4 (>5kg). To the right is a checkbox labeled 'Tick if speed post is required' with the text 'Speed post' below it. At the bottom left is a 'Delivery cost' button, and to its right is a text input field.
- Question 1.3:** Located at the bottom left, it contains a 'Delivery box number' button and a text input field.
- Question 1.4:** Located at the top right, it has a 'UPC bar code' text input field containing '639382000393' and a 'Validate bar code' button. Below these is a large empty text area.
- Question 1.5:** Located at the bottom right, it contains a 'Select a city' dropdown menu set to 'Cape Town' and a 'View and save deliveries' button. Below these is a large empty text area.

- Complete the code for each section of QUESTION 1 as described in QUESTIONS 1.1 to QUESTION 1.5 below.

1.1 Button – [Confirm Delivery]

Obtain the following data from the relevant components:

- Place of departure from the **Place of Departure** combo box
- Destination from the **Destination** combo box
- Number of kilometres from the **Kilometres** text box

Create a line of text as output that indicates the place of departure, the destination and the number of kilometres as shown in the example below. Place the constructed line of text in the label component provided.

Example of possible input:

Select the place of departure	Johannesburg
Select the destination	Durban
Enter the number of kilometres	635

Required output:

Delivery label
Johannesburg to Durban : 635 km

(5)

1.2 Button – [Delivery cost]

The following components are provided:

- A list box that indicates the weight categories of deliveries in terms of codes (A1–A4) in the following format:

<code><space><(range in kg)>

A1 (0kg - 1kg)
A2 (>1kg - 2kg)
A3 (>2kg - 5kg)
A4 (>5kg)

The following tariffs per weight category apply:

Category	Tariff per kilometre
A1	R0,60
A2	R1,00
A3	R1,25
A4	R1,65

- A check box that indicates whether speed post must be used. A standard amount of R100,00 is charged for speed post.

When the user clicks on the **Delivery cost** button the number of kilometres entered in QUESTION 1.1, the selected weight category and whether speed post is required or not must be used to calculate the delivery cost.

Example of the output if the number of kilometres is 635, the weight of the item to be delivered is in the A2 category and speed post is required:

The screenshot shows a window titled "Question 1.2". Inside, there are two main sections. The left section is titled "Select the category" and contains a list box with four options: "A1 (0kg - 1kg)", "A2 (>1kg - 2kg)", "A3 (>2kg - 5kg)", and "A4 (>5kg)". The "A2 (>1kg - 2kg)" option is selected and highlighted in blue. The right section is titled "Tick if speed post is required" and contains a checkbox labeled "Speed post" which is checked. At the bottom of the window, there is a button labeled "Delivery cost" and a text box displaying the calculated cost "R735.00".

(10)

1.3 Button – [Delivery box number]

Items to be delivered need to be placed in specific delivery boxes. The correct delivery box for each individual item must be determined using the criteria below:

- There are five delivery boxes numbered from 1 to 5.
- All **speed post items** will be placed in delivery box **4**.
- All the other delivery items will be **randomly** placed in the remaining delivery boxes (1, 2, 3 or 5).

Display the number of the delivery box into which the item must be placed.

Example of output if speed post was requested in QUESTION 1.2 (on the next page):

Question 1.2

Select the category

A1 (0kg - 1kg)
A2 (>1kg - 2kg)
A3 (>2kg - 5kg)
A4 (>5kg)

Tick if speed post is required

☒ Speed post

Delivery cost

R735.00

Question 1.3

Delivery box number

4

Example of output when speed delivery is NOT required in QUESTION 1.2:

Question 1.2

Select the category

A1 (0kg - 1kg)
A2 (>1kg - 2kg)
A3 (>2kg - 5kg)
A4 (>5kg)

Tick if speed post is required

☐ Speed post

Delivery cost

R635.00

Question 1.3

Delivery box number

3

NOTE: Due to the nature of the random function the value for the delivery box displayed in the screenshot above may differ from the value displayed by your program.

(9)

1.4 Button – [Validate bar code]

A Universal Product Code (UPC) bar code is printed on items to be delivered. The picture on the next page shows an example of a UPC bar code:



The bar code number consists of twelve digits. For example, the bar code number shown in the picture above is 639382000393. The last digit of a UPC is called a check digit. The check digit is used by a scanner to determine whether a bar code is valid or not.

Use the algorithm below to write code for the **Validate bar code** button. The code needs to verify the check digit and display a message to indicate whether the bar code is valid or not.

Algorithm:

1. `barCode` \leftarrow text box value
2. `sumOddPositions` \leftarrow 0
3. `sumEvenPositions` \leftarrow 0
4. loop from first digit to second last digit of `barCode`
5. if the logical position of the digit in the bar code is even
 `sumEvenPositions` \leftarrow `sumEvenPositions` + digit at position
 else
 `sumOddPositions` \leftarrow `sumOddPositions` + digit at position
6. `sum` \leftarrow `sumOddPositions` * 3 + `sumEvenPositions`
7. `checkDigit` \leftarrow 10 – (sum modulus 10)
8. if `checkDigit` = last digit of `barCode`
 display the check digit and a suitable message indicating that the bar code is valid
else
 display a suitable message indicating that the bar code is invalid

Example:

If the bar code is 639382000393 then:

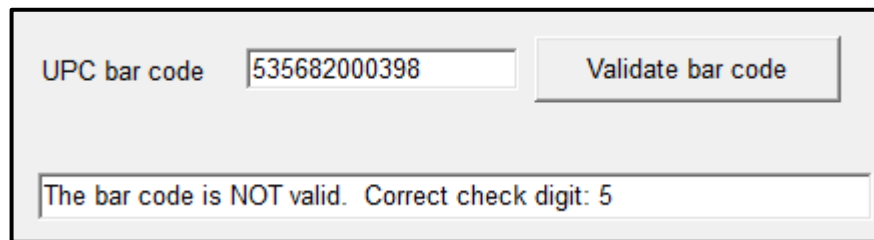
$$\text{sumOddPositions} = 6+9+8+0+0+9 = 32$$

$$\text{sumEvenPositions} = 3+3+2+0+3 = 11$$

Example of a possible output for a valid bar code:

UPC bar code	<input type="text" value="639382000393"/>	<input type="button" value="Validate bar code"/>
<div>The bar code is valid. Check digit: 3</div>		

Example of a possible output for an invalid bar code:



(14)

1.5 Button – [View and save deliveries]

All deliveries for December 2014 are stored in the given array called **arrDecDeliveries**. The format of each entry in the array is as follows:

<date><space><place of departure><space>to<space><destination>

Example:

2014-12-01 Durban to Cape Town

The user has to select a city from the provided combo box. All the deliveries that were made during December 2014 to or from the selected city must be displayed in the provided output area and also written to a text file.

Write code to do the following:

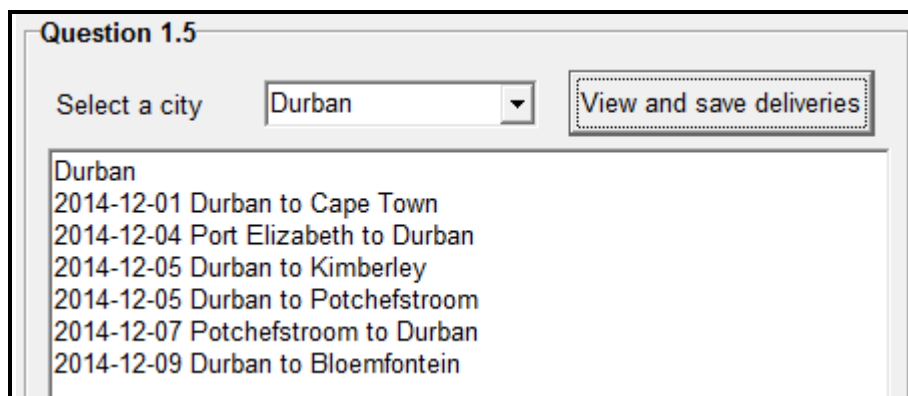
- Create a text file where the name of the file is made up of the text 'December2014' combined with the name of the city that was selected.

Example:

If Durban was selected, the name of the text file must be **December2014Durban.txt**.

- Use the data stored in the **arrDecDeliveries** array and display the deliveries to and from the selected city in the output area provided. Use the name of the city as a heading.
- Store the deliveries to and from the selected city in the text file that was created – one delivery per line.

Example of the contents of the output area if the city of Durban is selected:



Question 1.5

Select a city:

Durban
2014-12-01 Durban to Cape Town
2014-12-04 Port Elizabeth to Durban
2014-12-05 Durban to Kimberley
2014-12-05 Durban to Potchefstroom
2014-12-07 Potchefstroom to Durban
2014-12-09 Durban to Bloemfontein

Example of the contents of the **December2014Durban.txt** text file:

```
2014-12-01 Durban to Cape Town
2014-12-04 Port Elizabeth to Durban
2014-12-05 Durban to Kimberley
2014-12-05 Durban to Potchefstroom
2014-12-07 Potchefstroom to Durban
2014-12-09 Durban to Bloemfontein
```

(12)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

TOTAL SECTION A: 50

SECTION B**QUESTION 2: OBJECT-ORIENTATED PROGRAMMING**

SuperTrans Courier Services owns five trucks. The trucks are classified as light, medium or heavy-duty trucks. The trucks are used on four different routes (RN1 to RN4) to deliver cargo.

INSTRUCTIONS:

Delphi programmers	Java programmers
<p>The project Question2 is provided in the DelphiDataENG folder which contains:</p> <ul style="list-style-type: none"> ○ A main form unit called Question2_U.pas ○ An incomplete unit file called Delivery_U.pas ○ A text file (DeliveryInfo.txt) that contains information on deliveries <ul style="list-style-type: none"> • Open the incomplete project file called Question2_P.dpr in the Question2 folder. • View (Ctrl + F12) the unit file Delivery_U.pas and add your examination number as a comment in the first line of both files Question2_U.pas and Delivery_U.pas. 	<p>The project Question2 is provided to you in the JavaDataENG folder which contains:</p> <ul style="list-style-type: none"> ○ A GUI class file called Question2.java ○ An incomplete object class file called Delivery.java ○ A text file (DeliveryInfo.txt) that contains information on deliveries <ul style="list-style-type: none"> • Open the incomplete classes called Question2.java and Delivery.java in the folders Source Packages (src), Question2Package. • Add your examination number as a comment in the first line of both classes Question2.java and Delivery.java.

Do the following:

- Compile and execute the program. The program currently has no functionality. An example of the interface is shown on the next page:

SuperTrans Courier Services	
Create and display new delivery object Select truck number <input type="text" value="Tr1"/> <input type="button" value="Get data from file"/> New delivery number <input type="text"/> Start odometer reading <input type="text"/> Enter end odometer reading <input type="text"/> <input type="button" value="New delivery"/> <input type="button" value="Display delivery"/> <div style="border: 1px solid black; height: 50px; width: 100%;"></div>	Fuel used Actual fuel used <input type="text"/> <input type="button" value="Check fuel used"/> <input type="text"/> Toll fees Route to be followed <input type="text"/> <input type="button" value="Calculate toll fees"/> <input type="text"/>

- Complete the code for this program as specified in QUESTION 2.1 and QUESTION 2.2 below.

2.1 The given incomplete object class (**TDelivery/Delivery**) contains the following code:

- The declaration of five attributes which describes a **delivery** object
- The declaration of a two-dimensional array to be used to determine toll fees
- A **toString** method

The attributes of a **delivery** object are the following:

Names of attributes		Description
Delphi	Java	
fDeliveryNum	deliveryNum	Number assigned to a specific delivery, for example 1, 2, et cetera
fTruckNum	truckNum	Truck number, for example Tr1, Tr2, Tr3, Tr4 or Tr5
fFuelUsed	fuelUsed	Fuel used during the delivery
fOdoStart	odoStart	Odometer reading at the start of the delivery
fOdoEnd	odoEnd	Odometer reading on completion of the delivery

Complete the code in the given **delivery** class (**TDelivery/Delivery**) as described in QUESTION 2.1.1 to QUESTION 2.1.4 below:

2.1.1 Write code for a constructor method to receive the delivery number, truck number, odometer reading at the start of the delivery and odometer reading on completion of the delivery as parameter values. Assign these values to the relevant attributes of the object class. (3)

2.1.2 Write a mutator method and an accessor method for the **fFuelUsed/fuelUsed** attribute. (4)

2.1.3 Write a method called **calculateDistance** to calculate and return the distance travelled based on the start and end odometer readings for the delivery. (3)

2.1.4 Different toll fees must be paid on different toll routes. The routes used are RN1, RN2, RN3 and RN4. Toll fees on these routes are dependent on the type of truck used. The company's trucks are classified as follows:

- Light-duty trucks: Tr1, Tr2
- Medium-duty truck: Tr3
- Heavy-duty trucks: Tr4, Tr5

A two-dimensional array called **tollFees** contains the toll fees for the different routes for different types of trucks and is supplied as part of the given code. The contents of the array can be represented as follows:

	Light-duty truck	Medium-duty truck	Heavy-duty truck
RN1	R105,50	R135,00	R210,00
RN2	R35,00	R54,00	R82,00
RN3	R85,00	R129,00	R205,00
RN4	R112,00	R170,00	R219,00

Rows: Represents the routes RN1 to RN4

Columns: Represents the types of trucks

Write a method called **determineTollFees** to determine and return the toll fees to be paid for the delivery. The method must receive the route (RN1, RN2, RN3 or RN4) as a parameter. Use the two-dimensional array called **tollFees** to look up the toll fee for the route and type of truck that was used for the delivery.

NOTE: It is compulsory to use the given two-dimensional array in your solution to look up the toll fee. (10)

- 2.2 A text file named **DeliveryInfo.txt** contains an unknown number of lines of information on previously completed deliveries. Each line of information contains data on a single delivery in the following format:

<delivery number>#<truck number>#<odometer reading on completion of the delivery>
--

Example of some of the data in the text file named **DeliveryInfo.txt**:

1#Tr1#121110
2#Tr2#8010
3#Tr3#15021
4#Tr4#700
5#Tr1#121453
6#Tr3#15653
:

The data of the first delivery can be interpreted as follows:

- Delivery **1** identifies the delivery.
- Truck **Tr1** was used to make the delivery.
- The reading on the odometer on completion of the delivery was **121110**.

Do the following to complete the code for each button in the main form unit (Delphi)/GUI class (Java) as described below:

2.2.1 Button – [Get data from file]

- Select the specific truck to be used from the provided **Truck number** combo box.
- Use the given text file **DeliveryInfo.txt** to determine the following:
 - **New delivery number**
The delivery number for the new delivery will follow on the number of the last delivery stored in the text file. If the information of 20 deliveries are stored in the text file, the number of the new delivery will be 21.
 - **Start odometer reading**
The odometer reading for the last delivery that was made by the selected truck must be used as the start odometer reading for the new delivery.

Example:

If truck Tr4 is selected and the odometer reading that was captured in the text file on completion of the last delivery for Tr4 is 1648, then the value of 1648 must be used as the odometer reading at the start of the new delivery.

- Display the **new delivery number** and **start odometer reading** in the text boxes provided.
- Display a suitable message in a dialog box and terminate the program if the text file cannot be accessed.

Example of the content of the relevant text boxes when truck Tr4 is selected:

The screenshot shows a Java Swing window titled "Create and display new delivery object". It contains a dropdown menu for "Select truck number" with "Tr4" selected. Below it is a button labeled "Get data from file". There are three text input fields: "New delivery number" containing "21", "Start odometer reading" containing "1648", and "Enter end odometer reading" which is empty. At the bottom are two buttons: "New delivery" and "Display delivery", followed by a large empty rectangular area.

NOTE: If you are unable to read the required information from the given text file, type in the number and the start odometer reading for the delivery into the text boxes to continue with the rest of the program.

(14)

2.2.2 Button – [New delivery]

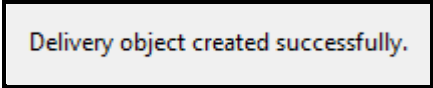
To instantiate a new **Delivery** object, the user needs to first enter the end odometer reading for the delivery in the textbox provided.

NOTE: The end odometer reading entered by the user must be greater than the previous odometer reading of the truck, because it is the start odometer reading for this delivery. No validation is necessary.

NOTE: The **Delivery** object variable has already been declared globally as part of the given code.

Write code for the **New delivery** button to do the following:

- Use the new delivery number, the truck number, the odometer reading at the start and the odometer reading at the end of the delivery to instantiate the **Delivery** object.
- Display a message indicating that the object has been instantiated successfully.



Delivery object created successfully.

- Obtain the distance travelled by calling the **calculateDistance** method and calculate the estimated litres of fuel used for the delivery. Use the following information:

One litre of fuel is used for every five kilometres travelled.

Use the method you have written in QUESTION 2.1 to set the **fFuelUsed/fuelUsed** attribute to this calculated value.

NOTE: The following buttons must be enabled:

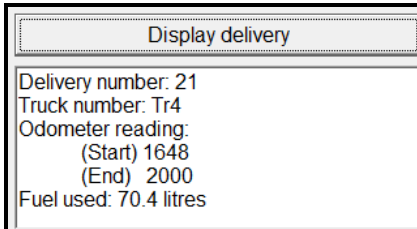
- Check fuel used
- Calculate toll fees

(10)

2.2.3 Button – [Display delivery]

Display the object in the output area provided using the **toString** method.

Example of the output for truck Tr4 with an end odometer reading of 2000:



Display delivery

Delivery number: 21
Truck number: Tr4
Odometer reading:
 (Start) 1648
 (End) 2000
Fuel used: 70.4 litres

(2)

2.2.4 Button – [Check fuel used]

The driver of the truck is required to refill the fuel tank at the end of each delivery. Due to the nature of the routes, the amount of fuel used to refill the tank may sometimes differ from the fuel calculated for the delivery.

Write code to do the following:

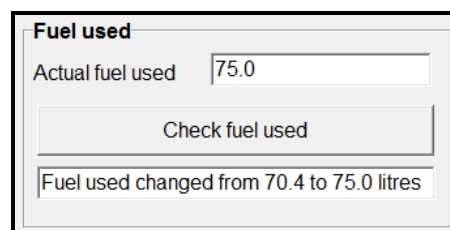
- Enter the amount of fuel used to refill the tank in the text box provided.

HINT: The value of 75 can be used to test the code.

- Obtain the calculated amount of fuel used from the relevant attribute in the object class.
- Calculate the difference between the fuel used to refill the tank and the calculated fuel used as stored in the attribute of the **Delivery** object.
 - If the difference is less than 10%, the **fFuelUsed/fuelUsed** attribute must be updated by setting its value to the new value as entered in the text box.

An appropriate message must be displayed in the text box provided, indicating whether the value of the **fFuelUsed/fuelUsed** attribute has been updated or not.

Example of the output if the delivery was done by truck Tr4 and the actual fuel used is 75,0:



- If the difference is greater than or equal to 10%, an error message should be displayed and the value of the **fFuelUsed/fuelUsed** attribute must not be updated.

Example of the output if the delivery was done by truck Tr4 and the actual fuel used is 85,0:

(9)

2.2.5 Button – [Calculate toll fees]

To determine the toll fees for the route used for the new delivery, the route number (for example RN3) must be entered in the text box provided. This route number must be sent as a parameter to the **determineTollFees** method.

The toll fees must be displayed in the label component provided. The amount must be displayed as currency (rand) to TWO decimal places.

Example of the output if the delivery was done by truck Tr4 on route RN3:

(5)

- Ensure that your examination number is entered as a comment in the first line of the class as well as the form.
- Save all the files.
- Print code you created for both the classes, if printouts are required.

TOTAL SECTION B: 60

SECTION C**QUESTION 3: PROBLEM-SOLVING**

SuperTrans Courier Services provides a daily speed service from Cape Town to Johannesburg. The storage space in the truck used for the deliveries is divided into two shelves for fragile and non-fragile items respectively. The maximum capacity of the shelf reserved for fragile items is 20 items while a maximum of 30 items can be placed on the shelf reserved for non-fragile items. The loading zone at the company controls the clearing of a load, loading of items into the truck and checking the status of the truck load at any given time.

An incomplete program has been provided to manage the loading of trucks.

INSTRUCTIONS:

Delphi programmers	Java programmers
<ul style="list-style-type: none"> The project Question3 is provided in the DelphiDataENG folder. Open the incomplete project file Question3_P.dpr in the Question3 folder. Add your examination number as a comment in the first line of the main form unit (Question3_U) files. 	<ul style="list-style-type: none"> The project Question3 is provided in the JavaDataENG folder. Open the incomplete class called Question3.java in the folders Source Packages (src), Question3Package. Add your examination number as a comment in the first line of the class (Question3).

Do the following:

- Compile and execute the program. The program currently has no functionality. An example of the interface is given below.

<p>Loading zone</p> <div style="display: flex; justify-content: space-between;"> <div> <input checked="" type="radio"/> Fragile <input type="radio"/> Non-fragile </div> <div>Load item</div> </div> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> <div>Loading code</div> <div><input type="text"/></div> </div> <div style="text-align: center; margin-top: 10px;"> <div style="border: 1px solid black; padding: 5px; width: 100%;">Check load status</div> <div style="border: 1px solid black; padding: 5px; width: 100%;">Clear load</div> </div>	<p>Loading progress display area:</p> <p>=====</p> <p>Fragile items:</p> <p>Non-fragile items:</p>
--	---

The output area of the GUI labelled 'Loading progress display area' represents the storage area of the truck. An asterisk (*) is used to represent an item placed on a shelf.

Example of the display of a fully loaded truck with 20 fragile and 30 non-fragile items:

```

Loading progress display area:
=====
Fragile items:      *****
Non-fragile items: *****
  
```

- Write code to complete QUESTION 3.1 to QUESTION 3.3 as explained in the instructions below:

3.1 Button – [Load item]

Write code to do the following when an item is loaded:

- Create a loading code.
- Add the item, represented by an asterisk (*), to the correct shelf in the loading progress display area.

Loading code:

The loading code is compiled using the letter 'F' for fragile items and the letters 'NF' for non-fragile items, followed by the sequence number of the item on the shelf.

For example:

F1 refers to item 1 on the shelf for fragile items.

NF6 refers to item 6 on the shelf for non-fragile items.

Add item to loading progress display area:

If there is space on the shelf, create and display the loading code in the text box provided and update the loading progress display area to show the new item.

If the item cannot be loaded, the loading code must be left empty and a dialog box must be used to display the following message: '**Loading of item cannot be processed – No loading space**'


Example of output of the 'Loading progress display area' if the item loaded is the first non-fragile item. In this example five fragile items have already been loaded (on the next page):

Loading zone <input type="radio"/> Fragile <input checked="" type="radio"/> Non-fragile Loading code <input type="text" value="NF1"/>		Loading progress display area: <hr/> Fragile items: ***** Non-fragile items: *
<input type="button" value="Load item"/>		

Example of an attempt to load a fragile item when the shelf for fragile items is full:

Loading zone <input checked="" type="radio"/> Fragile <input type="radio"/> Non-fragile Loading code <input type="text"/> <input type="button" value="Load item"/> <input type="button" value="Check load status"/> <input type="button" value="Clear load"/>		Loading progress display area: <hr/> Fragile items: ***** Non-fragile items: *****
--	--	---

Information [X]

 Loading of item cannot be processed - No loading space

(20)

3.2 **NOTE:** If you could not complete QUESTION 3.1, use the data below to complete QUESTION 3.2:

Number of fragile items: 4
 Number of non-fragile items: 13

Button – [Check load status]

The delivery can be made if the truck has a minimum load of:

- 50% fragile items (10 fragile items), and
- 50% non-fragile items (15 non-fragile items)

When this button is clicked, a load status report must display a summary of the number of both fragile and non-fragile items, using the following column headings:

Item type	Number of items	Percentage loaded
-----------	-----------------	-------------------

- If the percentage of items loaded is 50% or more for both fragile and non-fragile items, the message **'The delivery may progress'** must be displayed.

- If the percentage of items loaded is less than 50% for fragile OR non-fragile items, then the message '**The delivery may not progress**' must be displayed along with the outstanding number of items for each type required to make up the minimum load.

NOTE: Marks will be awarded for column formatting and for displaying the percentage loaded to TWO decimal places.

Example 1: If the load status of the truck is 4 fragile items and 10 non-fragile items, the load status report should be as follows:

```
Load status report:
=====
Item type      Number of items      Percentage loaded
Fragile         4                      20.00
Non-fragile    10                      33.33

The delivery may not progress.
Number of fragile items still required: 6
Number of non-fragile items still required: 5
```

Example 2: If the load status of the truck is 12 fragile items and 17 non-fragile items, the load status report should be as follows:

```
Load status report:
=====
Item type      Number of items      Percentage loaded
Fragile        12                      60.00
Non-fragile    17                      56.67

The delivery may progress.
```

(17)

3.3 Button – [Clear load]

Initialise all variables and data structures to prepare for a new load. Also clear all text from the output area.

(3)

- Enter your examination number as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

TOTAL SECTION C: 40
GRAND TOTAL: 150