

St Andrew's School for Girls



INFORMATION TECHNOLOGY PAPER I

Portfolio Assessment

July 2026

Time: 2 hours

100 Marks

Examiner: External

Moderator: L Cawdry

**EXAMINATION
NUMBER**

--

Instructions

1. This paper consists of 17 pages – please check it is complete
2. The paper is divided into two sections. All candidates must answer both sections.
3. Alert the invigilator immediately if you have hardware or software-related problems with your allocated PC. Do not waste time by trying to resolve the issue by yourself.
4. Ensure 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 questions.
5. Only answer what is asked for in each question.
6. If you cannot get a section of code to work, comment it out so that it will not be executed, and you can continue with the examination. If possible, try and explain the error to aid the marker.
7. Your programs must be coded to work with any data, 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.
8. Ensure that routines such as searches, sorts and selections are developed from first principles, and that you do not use the built-in features of a programming language for any of these routines.

9. Make sure that your examination number appears as a comment in every program you code and on every printed page you hand in.
10. At the end of the session, you must make printouts of ALL your classes. Please comment you name/exam number at the top of each class.
11. You should have one folder containing the following files. These files are to be used as data for the examination.

FireData.txt

Callouts.txt

QUESTION	1	2	3	4	5	TOTAL	%
MARKS	31	14	26	5	24	100	
MARKED							

SCENARIO

SA FireWatch is a national wildfire management system that coordinates firefighting resources across South Africa. Devastating fires break out annually across the Western Cape at locations like Table Mountain, Chapman's Peak, Jonkershoek and Tokai. These fires are managed by a network of fire stations.

Two types of firefighters are registered in the system:

- Permanent firefighters — full-time employees who are always available for deployment, provided they have not exceeded the maximum number of fires for the current period.
- Volunteer firefighters — part-time responders who can only be deployed if their training is complete AND their skill level is at least as high as the fire's severity AND they have not exceeded the maximum fires for the period.

The details of the firefighters are stored in a single text file called **FireData.txt**.

The program will make use of the following classes.

Firefighter Class

This class will represent the basic details of permanent firefighters.

- **name** - the full name of the firefighter
- **stationName** - the name of the station that the firefighter is registered at
- **callSign** – the firefighter call sign. The callsign for permanent firefighters is prefixed with the letter 'F'.
- **dateJoined**– the date the firefighter joined the station
- **firesAttended** – the number of fires the firefighter has attended in the current period
- **callouts** – an array to track the callout codes for all fires attended by the firefighter in the current period. The **firesAttended** variable is the index used for the **callouts** array.

VolunteerFirefighter Class

This class will represent the basic details of volunteer firefighters.

- **skillLevel** – indicating the maximum severity of a fire they are trained to fight
- **trainingComplete** – a boolean value indicating if the volunteer has completed their training

The details of the permanent and volunteer firefighters are stored in the file **FireData.txt** in no specific order.

A line of the file representing a permanent firefighter has the following format:

<name>;<stationName>;<callSign>;<dateJoined>

For example:

```
Kabelo Mokoena;Newlands;F12;2016-03-11
```

A line of the file representing a volunteer firefighter has the following format:

```
<name>;<stationName>;<callSign>;<dateJoined>;<skillsLevel>;<trainingComplete>
```

*the call sign starts with a “F” is the fire fighter is permanent and starts with a “V” is the fire fighter is a volunteer

For example:

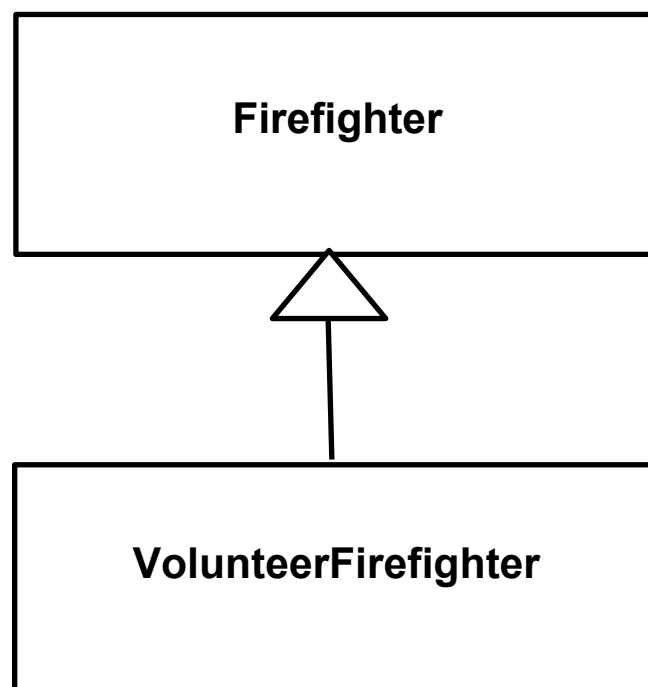
```
Lara October;Hout Bay Station;V07;2021-01-10;4;true
```

The first 6 firefighter records from **FireData.txt** are shown below.

```
Kabelo Mokoena;Newlands;F12;2016-03-11
Thabo Nkosi;Stellenbosch Station;F22;2014-06-24
Lara October;Hout Bay Station;V07;2021-01-10;4;true
James Smith;Newlands;V11;2023-02-15;3;false
Ayesha Khan;Constantia Station;F33;2012-07-01
Sipho Dlamini;Hout Bay Station;V21;2018-11-30;5;true
```

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

The relationship between the parent class **FireFighter** and the child class **VolunteerFireFighter** is shown in the diagram below.



Answer the questions that follow to create a program that will allow the user to manage the stations and firefighters to provide manpower to respond to callouts for fires.

QUESTION 1

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

Firefighter
Fields: - name: string - stationName: string - callSign: string - dateJoined: Date - firesAttended: integer - callouts[5]: string <u>+ MAXFIRES = 5: integer</u>
Methods: + Constructor(inName : string, inStation : string, inCallSign : string, inDate : string) + getName() : string + getCallSign() : string + getStationName() : string + getFiresAttended() : integer + isVeteran(): boolean + canDeploy(severity : integer, callout : string) : boolean + updateCallOut(callout:string) + toString() : string + getCalloutList() : string

- 1.1 Create a class called **Firefighter** with the fields **name**, **stationName**, **callSign**, **dateJoined**, **firesAttended** and **callouts** using the data types in the class diagram. These fields must not be accessible outside the class. (3)
- 1.2 Add the class constant **MAXFIRES** as shown in the class diagram. (2)
- 1.3 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.

Note: the parameter **inDate** is a string and must be converted to a date. The callouts array will be populated in a later question. Fires attended must be given an initial value of 0. (5)
- 1.4 Create accessor methods for the **name**, **callSign**, **stationName** and **firesAttended** fields. (4)

- 1.5 A firefighter is a veteran if they have been in service for 8 or more complete years from their `dateJoined` to the current date. Code a method called **isVeteran** that returns true if the firefighter qualifies. Do NOT hardcode today's date i.e. DO NOT hardcode the date you are completing this assessment. (4)
- 1.6 Create a method **updateCallout** that accepts a string parameter **callout**. The callout must be inserted into the array **callouts** at index `firesAttended` and the number of **firesAttended** must be increased by 1. (3)
- 1.7 A firefighter can be deployed as long as **firesAttended** is less than `MAXFIRES`. Code a method called **canDeploy** that accepts the fire severity and callout code as a parameter and returns true if `firesAttended < MAXFIRES`, and false otherwise. The method must call the **updateCallout** method to update the callout array and `firesAttended` variables if the firefighter can be deployed. (4)
- 1.8 Create a **toString** method that will return a string that combines the values of the **name**, **callsign** and **station** fields in the **Firefighter** class. The format of the string should be as follows:
- name**<tab>**callSign**<tab>**stationName**
- If the firefighter is a veteran, append `<space>'VETERAN'` to the string. For example:
- Riaan Botha F57 Paarl Station VETERAN
- Zanele Mthembu F29 Hout Bay Station (2)
- 1.9 Code a method called **getCalloutList** that returns a string listing the elements in the **callouts** array, if at least one fire has been attended. Each element must be separated by a comma. Note `firesAttended` is the variable used to track the number of elements in the **callouts** array.
- Sample output:
- N01, N02 (4)

[31]

QUESTION 2

Use the class diagram below to create a new class called **VolunteerFirefighter**. This class will be used to store the details of a volunteer firefighter and is a child class of the **Firefighter** class.

VolunteerFirefighter
Fields: - skillLevel : integer - trainingComplete: boolean
Methods: + Constructor(inName : string, inStation : string, inCallSign : string, inDate : string, inSkill : integer, inTrained : boolean) + canDeploy(severity : integer, callout : string) : boolean + toString() : string

- 2.1 Create a new class called **VolunteerFirefighter** which is a child class of **Firefighter** with the fields **skillLevel** and **trainingComplete** as shown in the class diagram. These fields must not be accessible outside the class. (3)
- 2.2 Write code to create a parameterised constructor method that will accept parameters as shown in the class diagram. Initialise the fields of the **FireFighter** class and the **VolunteerFirefighter** class. (3)
- 2.3 Code a **canDeploy** method that will override the parent's **canDeploy** method. A volunteer firefighter can be deployed if ALL THREE of the following conditions are met.
 - trainingComplete is true
 - skillLevel is greater than or equal to the fire's severity
 - firesAttended is less than MAXFIRESIf the volunteer can be deployed the firefighters callout details must be updated using the updateCallout method in the parent class. (5)
- 2.4 Code a **toString** method that will override the parent class's **toString** method. Add the skillLevel and trainingComplete fields to the parents **toString** and return a string in the following format (3)
name<tab>callSign<tab>stationName<space>'Skill:'<space>skillLevel<space>'Trained:'<space>trainingComplete

For example:

Lara October V07 Hout Bay Station Skill: 4 Trained : true

[14]

QUESTION 3

- 3.1 Create a class called **FireManager**. (1)
- 3.2 Create two fields for the **FireManager** class as described below.
An array called **ffArr** to store up to 50 **Firefighter** objects.
A counter called **ffSize** to count the objects added to **ffArr**.
The fields should not be accessible outside the class. (2)
- 3.3 Create a constructor method that will read the contents of the textfile **FireData.txt** containing information about the firefighters. The textfile contains data about **FireFighters** and **VolunteerFirefighters**. The method should do the following.
- Open the file **FireData** for reading.
 - Loop through the text file. In each iteration of the loop:
 - Read the line and split the data into separate parts.
 - If the call sign starts with 'V', read the additional volunteer fields and create a **VolunteerFireFighter** object. Otherwise, create a **FireFighter** object. Store each object in **ffArr** and increment **ffSize**.
 - Handle any file-related exceptions with an appropriate error message.
- (12)
- 3.4 Code a **toString** method. This method should return a string with:
- The heading 'Fire Fighters'.
 - Underneath the heading the details of all **FireFighters**, each on a new line as shown in Question 4.3.
- (4)
- 3.5 Code a method called **deleteFireFighter** that accepts a firefighter's call sign as a parameter and removes the matching firefighter from **ffArr**. After removing the firefighter object, all remaining elements must be shifted up so there are no gaps, and **ffSize** must be decremented. You may use a helper method to perform the shift. Develop this from first principles — do not use any built-in list/array removal features of the programming language. (7)

[26]

QUESTION 4

- 4.1 Write code to create a text-based user interface called **FireUI** that will allow simple input and output. (1)
- 4.2 Declare and instantiate a **FireManager** object called **fm**. (1)
- 4.3 Write code to call the appropriate method in the **FireManager** class to display a list of all the **FireFighter** objects. The sample output is shown below. (1)

Fire Fighters

```
Kabelo Mokoena F12 Newlands VETERAN
Thabo Nkosi F22 Stellenbosch Station VETERAN
Lara October V07 Hout Bay Station Skill: 4 Trained : true
James Smith V11 Newlands Skill: 3 Trained : false
Ayesha Khan F33 Constantia Station VETERAN
Sipho Dlamini V21 Hout Bay Station Skill: 5 Trained : true
Nomvula Jacobs F44 Stellenbosch Station VETERAN
Chris Brown V19 Newlands Skill: 2 Trained : true
Lerato Maluleke F51 Newlands VETERAN
Pieter van der Merwe F55 Paarl Station VETERAN
Fatima Adams F38 Constantia Station
Bongani Khumalo F61 Stellenbosch Station VETERAN
Riaan Botha F57 Paarl Station VETERAN
Zanele Mthembu F29 Hout Bay Station
Amahle Ngcobo V23 Newlands Skill: 4 Trained : true
Daniel Pretorius V31 Paarl Station Skill: 2 Trained : true
Thandeka Zulu V27 Constantia Station VETERAN Skill: 5 Trained :
true
Megan Naidoo V33 Stellenbosch Station Skill: 1 Trained : false
Sibusiso Mahlangu V25 Hout Bay Station Skill: 3 Trained : true
Carmen Daniels V35 Paarl Station Skill: 5 Trained : false
```

- 4.4 Write code to call the appropriate method in the **FireManager** class to remove the firefighter with call sign 'V11' (1)
- 4.5 Display the updated list of stations and firefighters (1)

[5]

QUESTION 5

- 5.1 In the **FireManager** class, code a method called **processCallOuts** that reads the file **CallOuts.txt** and allocates firefighters to be deployed to each fire and returns a report in the form of a string. Each line in the file has the following format:

```
<callOutCode>;<stationName>;<fireName>;<severity>
```

The severity is an integer from 1 (minor) to 5 (catastrophic). The contents of **CallOuts.txt** are shown below:

```
N01;Newlands;Table Mountain Fire;5
H01;Hout Bay Station;Chapmans Peak Fire;4
S01;Stellenbosch Station;Jonkershoek Fire;4
C01;Constantia Station;Tokai Fire;2
```

For each line in the file:

- Extract the callout code, station name, fire name and severity.
- Add a section heading: 'Deployments from ' + stationName + ' to ' + fireName.
- Search `ffArr` for all firefighters at that station.
- For each firefighter found, call the `canDeploy` method to determine if the firefighter can attend the fire. If true, add: name + ' deployed'. If false, add: name + ' not deployed' to the report.
- Return the complete deployment report as a string.
- Handle any file-related exceptions appropriately. (7)

See the sample out in Question 5.2

- 5.2 In the **FireUI** add code to call the **processCallOuts** method and display the results. Ensure a blank line after the output from Question 4.5. (1)

Sample output is shown below (after the delete in Question 5.4). This is not the full report.

```
Deployments from Newlands to Table Mountain Fire
Kabelo Mokoena deployed
Chris Brown not deployed
Lerato Maluleke deployed
Amahle Ngcobo not deployed
```

```
Deployments from Hout Bay Station to Chapmans Peak Fire
Lara October deployed
Sipho Dlamini deployed
Zanele Mthembu deployed
Sibusiso Mahlangu not deployed
```

- 5.3 Add a method to the **FireManager** class called **reportBusiest** that determines which firefighter(s) attended the most fires and returns a string. The report must show the highest number of fires attended and, for each firefighter who attended that many, their name, call sign and the list of callout codes for the fires they were deployed to. Your solution must cater for the possibility that more than one firefighter attended the maximum number of fires.

Use the following format. Line 2 may be repeated for multiple firefighters.

```
"Most fires attended:"<space>max
  name<space><"(">callsign<")>:"> CalloutList
```

- 5.4 In the **FireUI** add code to call the **reportBusiest** method and display the results. (8)

The correct output is shown below (after the delete in Question 4.4, partial):

```
Most fires attended: 5
Kabelo Mokoena (F12): N01, N02, N03, N04, N05
Lerato Maluleke (F51): N01, N02, N03, N04, N05
```

- 5.5 Add a method to the **FireManager** class called **getStationStats** that accepts the station name as a string parameter and tallies the number of volunteer and permanent firefighters registered at the station. The method must return a string detailing the statistic for the station in the following format. (1)

```
"Statistics for"<space>station
"Volunteers:" <space> countV
"Permanent:" <space> countF
```

- 5.6 In the **FireUI** add code to call the **getStationStats** method for "Stellenbosch Station" and display the results. (6)

The correct output is shown below (after the delete in Question 4.4, partial): (1)

```
Statistics for Stellenbosch Station
Volunteers: 1
Permanent: 3
```

[24]

TOTAL FOR EXAMINATION [100 MARKS]