Somerset College

INFORMATION TECHNOLOGY
GRADE 11
NOVEMBER PRACTICAL EXAMINATION 2019

Examiner: R November
Moderator: J Botha

Time: 3 hours (+10min reading time)
Marks: 120

Instructions:

1. Save your work in \\fileserver\SeniorStudentData\EXAM-IN\"your exam number" (your exam folder).
2. This examination contains 10 pages including this cover page. Check that your paper is complete.
3. If you cannot get a piece of code to work then you may comment out that code so that it will not affect the execution of your program. The commented code will be marked as part of your answer.
4. Only answer the question that is asked. If the question does not ask for data validation then no marks are awarded for it and you are not required to perform that validation.
5. Code all your programs in Java.
6. Make sure that your name and today's date appear as comments at the top of the program.
7. Please add a comment in your program for each question and sub-question that you attempt.
8. Marks will be awarded for good programming style. "Brute Force" (hard coding) will not be awarded any marks.
9. All sorting, searching, inserting and deleting of array elements must be done using first principals. No marks will be awarded for the use of Array class methods.
10. The data and scenarios used in this assessment are all fictional and does not relate to any real event or person. Any such connections to real persons and/or events are a mere coincidence.
11. The text file "SAORI.txt" should be in your exam folder. Please check if it is there and that you can open the file.

SECTION A **STRUCTURED QUERY LANGAUGE**

Scenario:

You have been contacted by a group of urban farmers who are individuals who use their backyards, gardens, rooftops or community spaces to grown their own vegetables. They would like to improve tracking of which farmers grow which crops and to store information about which types of crops to grow and when to harvest them. They have prepared a small database to help you get started. Below are the fields and tables of the database "UrbanFarmers.accdb"

tblCrops

tblCrops

Field Name	Data Type	Description (Optional)
CropID	AutoNumber	The primary key field to uniquely identify each crop
CropName	Short Text	The name of the crop
PlantingMonth	Short Text	The month in which to plant the seeds
HarvestTime	Number	Time to harvest from seed (in weeks)
ProfitPerCrop	Currency	Estimated price in Rand per kg of crop

tblCrops

CropID	CropName	PlantingMonth	HarvestTime	ProfitPerCrop
1	Beans	March	9	R 15.00
2	Beets	September	8	R 5.00
3	Broccoli	February	13	R 20.00
4	Cabbage	April	14	R 8.00
5	Carrots	April	15	R 6.00

tblFarmers

tblFarmers

Field Name	Data Type	Description (Optional)
FarmerID	AutoNumber	The primary key field to uniquely identify each farmer
Owner	Short Text	The name of the farm owner
FarmName	Short Text	The name of the farm
DateStarted	Date/Time	The date the farm was started
Acres	Number	The area of the farm in acres
Organic	Yes/No	If the farmer focuses on growing organic crops
PreferedCrop	Number	The crop that the farmer prefers to grow

tblFarmers

FarmerID	Owner	FarmName	DateStarted	Acres	Organic	PreferedCrop
1	Cello Longworth	Oakey Dokey Orchard	2015/09/19	2	☐	1
2	Bartel Izhak	New Spring Nursery	2015/03/14	0.2	☐	13
3	Alexis Andrzej	Big Bear Orchard	2016/12/20	2.72	☑	13
4	Corrianne Grimmert	Sweet Dreams Gardens	2017/04/27	0.04	☑	1
5	Tully Andryunin	Willowbranch Ranch	2014/04/10	1.61	☐	6

QUESTION 1

Use the database "UrbanFarmers.accdb" to test your queries then paste your completed query into the document "SQLAnswerSheet.docx".

1.1) Write a query that will list all the details of the crops in descending order of harvest time. (3)

CropID	CropName	PlantingMonth	HarvestTime	ProfitPerCrop
10	Onions	January	30	R 4.00
12	Potatoes	September	18	R 4.00
15	Watermelon	November	15	R 4.00
5	Carrots	April	15	R 6.00
4	Cabbage	April	14	R 8.00
6	Corn	August	13	R 5.00

1.2) List the name of the crop and the planting month of all crops that can be planted in January, February or March. (2)

CropName	PlantingMonth
Beans	March
Broccoli	February
Onions	January
Radishes	February
Spinach	January

1.3) Write a query that will list the farm owner, farm name and date started for all the farmers whose farm name contains the word "farm" or garden and are organic. (3)

Owner	Farmname	DateStarted
Corrianne Grimmert	Sweet Dreams Gardens	2017/04/27
Randolph Antonoczyk	Blackmeadow Farm	2015/03/23
Karlan Ogborne	Badger Hill Farm	2014/05/30
Gae Joules	Daisy Chain Farm	2018/08/01

1.4) List the 5 farmers with the greatest amount of acres. Show only the farm name and the amount of acres (2)

FarmName	Acres
Oakdale Ranch	2.97
Dinky Creek	2.94
Strawberry Mountain Mea	2.94
Pitchfork Farmstead	2.93
Riverlands	2.88
	0

1.5) Write a query that will calculate and display the average amount of profit from all the crops. Name this field avgProfit. (2)

avgProfit
R10.00

1.6) Display all details of the crop with the highest profit. (4)
**Note that the output below could change if other crops are added in future*

CropID	CropName	PlantingMonth	HarvestTime	ProfitPerCrop
3	Broccoli	February	13	R 20.00
(New)			0	R 0.00

1.7) Write a query that will calculate and display how many farmers started per year. Display only the year as a field called "Years" as well as a field called "NumberOfFarmers". (4)

Years	NumberOfFarmers
2013	1
2014	20
2015	19
2016	12
2017	10
2018	14
2019	9

1.8) Create a query that will show how many acres have been allocated to each crop by the farmers but only if the total amount of acres for the crop exceeds 5 acres. You must show only the preferred crop and the total amount of acres as a field called "TotalAcres". (4)

PreferedCrop	TotalAcres
1	8.04
2	5.69
3	21.26
4	15.1
5	8.35
6	11.04
7	12.2
9	8.2
11	10.82
13	9.72
14	12.2

1.9) Write a query that will change the profit per crop in the tblCrops by 25%. (3)

1.10) Some of the farmers would like their farm names changed. Each farm name that ends in a "Ranch" must have the word replaced with "Range". (4)

e.g Willowbranch Ranch will become Willowbranch Range

For this query you may assume that the word "Ranch" will always be the last word in the farm name.

1.11) Add the following record into the tblCrops (4)

Crop name	Pumpkin
Planting month	October
Harvest time	18
Profit per crop	R24

1.12) Create a query that will display the farm name, crop name, profit per crop, acres and a field named "ProfitPerAcre" that will be calculated in the following way: (5)

Profit Per Acre = Profit per crop × Acres × 1000

FarmName	CropName	ProfitPerCrop	Acres	ProfitPerAcre
Oakey Dokey Orchard	Beans	R 15.00	2	30000
New Spring Nursery	Radishes	R 16.00	0.2	3200
Big Bear Orchard	Radishes	R 16.00	2.72	43520
Sweet Dreams Gardens	Beans	R 15.00	0.04	600
Willowbranch Ranch	Corn	R 5.00	1.61	8050
Rusty Bucket Acres	Peas	R 17.00	2.32	39440
Paradise Farmstead	Radishes	R 16.00	1.5	24000

[40]

SECTION B OBJECT-ORIENTATED PROGRAMMING

Scenario:

In the world of farming, the tractor is one of the most essential pieces of equipment. No modern farm can operate at a profit without the help of these machines and the various specialised attachments that they come with. Some of these farmers have heard of the great job you did with their urban farming counterparts and would like your help in creating a program to help manage the various attachments that one can buy for the different brands of tractors. The farms have prepared two files for you. The first is a text file containing information about all the various attachments that can be connected to the tractors and the second a list of all the different tractors owned by the farmers. Below are 5 lines from each of the files:

`Attachments.txt`

```
Loader#F-RTS-2007#175
Tiller#F-KHD-2001#191
Seeder#F-UMY-2017#150
Specialized#R-KHD-1997#118
Specialized#B-KHD-1991#185
```

Each line of the file contains information about an Attachment in the following format:

<type> # <model> # <minimum power requirement in horse power (HP)>

The model has information on the location of connection (F)ront (R)ear or (B)oth, brand and model year separated by a dash.

`Tractors.txt`

```
UMY#2017#290
BME#2011#230
MDE#2001#130
KHD#1999#110
UMY#2010#220
```

Each line of the tractor.txt file contains the data for a tractor in the format:

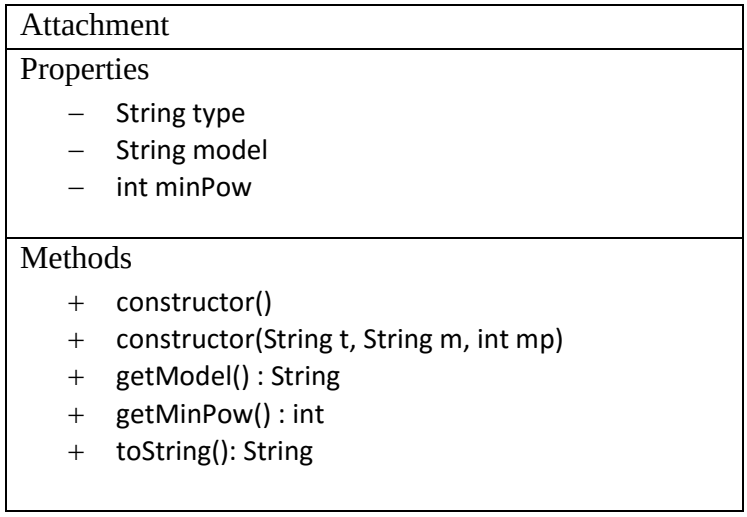
<brand> # <tractor year> # <maximum power output in horse power HP>

As you no doubt can see, the brand has been abbreviated. The following table is an expanded list for each of the brands.

Brands	
Mary Deere	MDE
Big Mike	BME
New Cape Tractors	NCT
Ubuntu Machinery	UMY
Rhino Tractors	RTS
Kaiju Heavy Duty	KHD

QUESTION 2

Using the UML class diagram below create a new Java class named Attachment.java. This class will be used to store all the information about a tractor attachment.



2.1) Create a **new class** named **Attachment** with the type, model and minPow properties as shown in the UML class diagram above. (4)

2.2) Create a **default constructor** for the class (1)

2.3) Code a **parameterised constructor** for the class that will accept three parameters representing the type , model and minimum power requirements for an Attachment and assign them to the appropriate properties in the class. (3)

2.4) Create **getter methods** for the model and minPow properties (2)

2.5) Code a **toString** method for the class that will return a String containing all the properties in the class in the following format: (3)

<type> Attachment – (<model>) [<minPow>HP req]

e.g. `Seeder#F-UMY-2017#150`

`Seeder Attachment – (F-UMY-2017) [150 HP req]`

[13]

QUESTION 3

Use the class diagram below to create a new class called Tractor. This class will be used to store the information of a tractor.

Tractor
Properties – String brand – int year – int horsePower – Attachment attachment
Methods + constructor(String b, int y, int h) + getBrand() : String + getYear(): int + getHorsePower() : int + getAttachment() : Attachment + setAttachment(Attachment a) + toString(): String

3.1) Code a new class named **Tractor** and add the properties as they appear in the diagram above. (2)

3.2) Create the **constructor** method as indicated in the diagram. Make sure to assign the parameters to the appropriate properties. (2)

3.3) Add **getter** methods for each of the properties of the class. (2)

3.4) Code a **setter** method for the attachment property. (2)

3.5) Create a **toSting** method to return the properties of the class as a String in the following format: (4)

<brand> , <year> - <horsepower> HP
Attachment: <attachment toString()>

If there is no attachment stored for the class then display "-not connected-"

e.g `UMY#2017#290`

```
UMY , 2017 – 290 HP
Attachment: -not connected-
```

with attachment `Seeder#F-UMY-2017#150`

```
UMY , 2017 – 290 HP
Attachment: Seeder Attachment – (F-UMY-2017) [150 HP req]
```

[12]

QUESTION 4

You will now need to create a TractorManager class that will be used to read the files, create objects and populate arrays. This class will handle the tractor data first and in another question you will be adding the attachments to the different tractors. It is recommended that you read through the scenario again as it will help you visualise the problem solutions more efficiently.

4.1	Create the class **TractorManager**.	(1)
4.2	Add two **private instance variables** to the class. One must be an **array** capable of storing **150 Tractor objects**. The other a **counter** to keep track of the number of objects added to the array.	(3)
4.3	Code a **constructor** method for the class that will accept one **String parameter** representing the name of the file containing the information of the tractors to be read. Your method should open the file, read each line, extract the data for a tractor and create the Tractor objects then add it to the array and then increment the counter.	(10)
4.4	Create a **toString** method for the class that will return a Sting containing all the toString information for each Tractor object in the array separated by a new line character.	(4)
4.5	Code a method named **sortByBrand** that will sort all the objects in the array in alphabetical order by their brand. This method should not accept any parameter and will not return any data.	(6)
4.6	Add a method called **deleteTractor**. This method should accept a number that represents the position of the object to be deleted. The method should check that the position is a valid index element of the array. This deletion should not cause any memory leaks and all elements should be reorganised so that there is efficient use of the space. This method should not return any data.	(5)

[29]

QUESTION 5

Now it is time to test your classes and the methods that you have created.

5.1)	Create a class called **TrackerUI**. This class should contain a main method.	(1)
5.2)	In an appropriate place in the class **instantiate** a new TractorManager object	(1)
5.3)	Using the **methods** you created in Question 4 do the following:	
5.3.1)	**Display** a list of all the tractors.	(1)
5.3.2)	**Sort** all the tractor by brand, and display the list of tractors again.	(1)
5.3.3)	**Delete** the first 5 tractors.	(2)

[6]

QUESTION 6

6.1) As you would no doubt have noticed none of the tractors have any attachment at the moment. You are required to create a method named **connectAttachments** in the **TractorManager** that will read through the entire text file of attachments and find a suitable attachment for each tractor. The using the appropriate method set the attachment to the tractor Before an attachment can be connected to a tractor it must meet the following requirements (18)

- Tractors can only have attachments of the same brand.
- The tractor must meet or exceed the minimum power requirements of an attachment.
- The year of the tractor must be greater or equal to the year model of the attachment.
- The farmer will always choose to equip an attachment that can be connected to either the front or the rear of the tracker and that uses as much power as the tractor produces. If the preference cannot be met then the attachment that can use the most power from the tractor should be selected and attached.

For this question you are allowed to create any other helper methods in any of the classes. However you may only add additional properties or variables to the **TractorManager** class.

Note that the file Attachments.txt contains 141 lines each representing a different type of attachment for a tractor.

Marks are awarded according to the efficiency of your solution.

6.2) Call the **connectAttachments** method in the appropriate class and redisplay all the tractor information. (2)

[20]

TOTAL MARKS [120]