4 A payroll program is to be written using an object-oriented programming language. An `Employee` class is designed. Two subclasses have been identified:

- `HourlyPaidEmployee` who is paid a monthly wage calculated from their hourly rate of pay and the number of hours worked during the month
- `SalariedEmployee` who is paid a monthly wage which is one 12th of their annual salary

(a) Draw an inheritance diagram for these classes.

[3]

(b) The design for the `Employee` class consists of:

- properties
  - `EmployeeName`
  - `EmployeeID`
  - `AmountPaidThisMonth`
- methods
  - `SetEmployeeName`
  - `SetEmployeeID`
  - `CalculatePay`

Write **program code** for the class definition of the superclass `Employee`.

Programming language ....

....

....

....

....

....

....

....

....

....

.... [5]

(c) **(i)** State the properties and/or methods required for the subclass `HourlyPaidEmployee`.

....

....

....

.... [4]

(ii) State the properties and/or methods required for the subclass `SalariedEmployee`.

....

....

....

.... [2]

(d) Name the feature of object-oriented program design that allows the method `CalculatePay` to be declared in the superclass `Employee`.

....

.... [1]

4 X-Games is an international extreme sports competition.

A program will store and process data about the teams in the competition.

- Each team is made up of members.
- Members can be added and removed from each team.
- Each member has a first name, last name, date of birth and gender.
- Each member can be an official or a competitor.
- Each official has a job title and may be first-aid trained.
- Each competitor takes part in one sport.

The program is written using object-oriented programming.

The program can output the full name and date of birth of any member. For example, “Nadia Abad 16/05/1995”

An introduction about a team member can be output using their name. For example, “Hello, I’m Nadia Abad”.

The program outputs a different version of the introduction for a competitor. This version includes the competitor’s sport. For example, “Hello, I’m Sally Jones and my sport is Skateboard Park.”

(a) Complete the following class diagram to show the attributes, methods and inheritance for the program.

You do not need to write the get and set methods.

Member
FirstName : STRING LastName : STRING DateOfBirth : DATE Gender : STRING

Constructor()
Introduction()
DisplayFullnameAndDateOfBirth()

Team
TeamName : STRING TeamList : ARRAY OF Member
Constructor() ....

Competitor
Sport : STRING

Constructor()
Introduction()

Official
....

Constructor()
DisplayJobTitle()

[3]

(b) Write **program code** for the `Member` class.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[5]

(c) Write **program code** for the `Competitor` class.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[5]

(d) Omar Ellaboudy is an official at X-Games. He is first-aid trained and his job title is Judge. He is male and was born on 17/03/1993.

Write **program code** to create an instance of an object with the identifier `BMXJudge`. All attributes of the instance must be fully initialised.

Programming language ....

Program code

....

....

....

....

....

....[3]

(b) The company wants to implement a program for the marking system. It will do this with object-oriented programming (OOP).

Many candidates take the examination. Each examination paper is given a `PaperID` that is made up of the centre (school) number followed by the candidate number.

Each examination paper is awarded a grade.

The following diagram shows the design for the `ExaminationPaper` class. This includes the attributes and methods.

```
                         ExaminationPaper
---------------------------------------------------------------------
FinalMark : INTEGER // maximum 2 digits, initialised to 0
Grade     : STRING  // "Pass", "Merit", "Distinction"
                    // or "Fail", initialised to "Fail"
PaperID   : STRING  // centre number followed by the
                    // candidate number, for example
                    // "ZZ00991001"
---------------------------------------------------------------------
Create()            // creates and initialises a new instance
                    // of the ExaminationPaper class using
                    // language-appropriate constructor
SetFinalMark()      // checks that the mark parameter has a
                    // valid value, if so, assigns it to
                    // FinalMark
SetGrade()          // sets Grade based on FinalMark
GetFinalMark()      // returns FinalMark
GetGrade()          // returns Grade
GetPaperID()        // returns PaperID
```

(i) The constructor receives the centre number and candidate number as parameter values to create `PaperID`. Other properties are initialised as instructed in the class diagram.

Write **program code** for the `Create()` constructor method.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [5]

(ii) Get and set methods are used to support the security and integrity of data in object-oriented programming.

Explain how get and set methods are used to support security and integrity.

....

....

....

....

....

.... [3]

(iii) Write **program code** for the following three get methods.

Programming language ....

`GetFinalMark()`

Program code

....

....

....

....

`GetGrade()`

Program code

....

....

....

....

`GetPaperID()`

Program code

....

....

....

....

[4]

© UCLES 2019

(iv) The method `SetFinalMark()` checks that its `INTEGER` parameter `Mark` is valid. It is then set as the final mark if it is valid. A valid mark is greater than or equal to 0 and less than or equal to 90.

If the mark is valid, the method sets the final mark and returns `TRUE`.
If the mark is not valid, the method does not set the final mark and returns `FALSE`.

Write **program code** for `SetFinalMark(Mark : INTEGER)`.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [5]

(v) Write **program code** for the method:

`SetGrade(DistMark, MeritMark, PassMark : INTEGER)`

Use the properties in the original class definition.

Grades are awarded as follows:

Grade	**Criteria**
Distinction	`>= DistMark`
Merit	`>= MeritMark`
Pass	`>= PassMark`
Fail	`< PassMark`

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(vi) Emily is a candidate who has taken the examination paper. The grades are awarded as follows:

Grade	**Criteria**
Distinction	`>= 80`
Merit	`>= 70`
Pass	`>= 55`

The procedure `Main()` performs the following tasks.

- allows the centre number, candidate number and mark to be input, with suitable prompts
- assigns an instance of `ExaminationPaper` to the variable `ThisPaper`
- sets the mark for the object
- sets the grade for the object
- outputs the grade for the object

Write **program code** for the `Main()` procedure.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(c) The examination paper will be taken by many candidates in centres around the world.

The program stores the objects of the `ExaminationPaper` class in a file. The company has decided to use a hash table, rather than a linked list to store the objects.

Explain why a hash table is more suitable than a linked list to store the objects.

....

....

....

....

....

....

....

.... [4]

7 A programmer is creating a computer game. The programmer has designed the class, `Character`, for the characters in the game.

The following class diagram shows the design for the `Character` class.

```
                              Character
----------------------------------------------------------------------------
Name : STRING       //  initialised in constructor to the parameter
                        value passed to the constructor
Skill : INTEGER     //  initialised in constructor to 0
Health : INTEGER    //  initialised in constructor to 50
Shield : INTEGER    //  initialised in constructor to a random value
                        between 1 and 25 (inclusive)
----------------------------------------------------------------------------
Constructor()       //  method used to create and initialise an object
GetName()           //  returns Name value
GetSkill()          //  returns Skill value
GetHealth()         //  returns Health value
GetShield()         //  returns Shield value
SetSkill()          //  increases Skill by the parameter value
SetHealth()         //  increases or decreases Health by the parameter
                        value
SetShield()         //  increases or decreases Shield value by the
                        parameter value
```

(a) Write **program code** for the `Constructor()` method. Use the appropriate constructor method for your chosen programming language.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

(b) Write **program code** for the `GetSkill()` method.

Programming language ....

Program code

....

....

....

....

.... [2]

(c) The method `SetSkill()` validates the parameter value and updates the value of `Skill`.

The method is passed an `INTEGER` parameter that must be between 10 and 25 (inclusive). A value outside of this is not valid.

If the parameter value is valid, the method will increase `Skill` by the parameter value. The maximum value that `Skill` can be increased to is 200. For example:

- `Skill` currently stores 180
- it is passed a valid parameter value of 25
- `Skill` will now store 200.

The method must return:

- `−1` if the parameter value is not valid
- `1` if the value of `Skill` is updated **and** `Skill` is less than 200
- `0` if the value of `Skill` is `200`.

Write **program code** for the `SetSkill()` method.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

.... [6]

(d) There are five characters in the game. All the character objects are stored in a 1D array.

Write **pseudocode** to declare the array, `CharacterArray`, to store the five character objects.

....

.... [2]

(e) The game has the character with the name Victory.

Write **program code** to create the character Victory as an instance of the class `Character`. The object needs to be stored in the first element of the array `CharacterArray`.

Programming language ....

Program code

....

....

.... [3]

Question 8 begins on the next page.

8 Files can be structured in serial, sequential or random format.

Tick (✓) **one** box in each row to show whether the statement applies to **Serial**, **Sequential** or **Random** format.

Statement	**Serial**	**Sequential**	**Random**
Uses a hashing algorithm			
No key field is used when storing data, for example, it is stored in chronological order			
Collisions can occur			
Least efficient for a very large number of records			
Most efficient for a very large number of records			

[3]

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

3 Ejaz is creating a program that will allow the user to create quizzes. He is using object-oriented programming (OOP).

There are two classes: `QuestionClass` and `QuizClass`.

The class attributes and methods are in the following tables. All attributes are declared as private.

QuestionClass	

`Question : STRING`
`Answer : STRING`
`Difficulty : INTEGER`
	`// stores the question`
	`// stores the correct answer`
	`// stores the difficulty as an integer`
	`// from 0(easy) to 10(hard)`

`Constructor(QuestionP, AnswerP, DifficultyP)`
`GetQuestion()`
`GetDifficulty()`
`GetAnswer()`
	`// creates an instance of QuestionClass`
	`// sets the attributes to the parameter`
	`// values`
	`// returns the question`
	`// returns the difficulty level`
	`// returns the answer`

QuizClass	

`Questions : ARRAY[0:19] OF QuestionClass`
`NumberOfQuestions : INTEGER`
	`// stores maximum 20 questions of`
	`// type QuestionClass`
	`// stores the number of questions`
	`// in this quiz`

`Constructor()`
`AddQuestion()`
`GetQuestion()`
`CheckAnswer()`
	`// creates an instance of`
	`// QuizClass`
	`// initialises NumberOfQuestions`
	`// to 0`
	`// adds the parameter question to`
	`// the array`
	`// increments NumberOfQuestions`
	`// returns the next question to be`
	`// asked`
	`// takes an answer as a parameter`
	`// and returns TRUE if correct`

(a) Write **program code** to define the class `QuizClass`. You are only required to write code for the attribute declarations and constructor.

If you are writing in Python, include attribute declarations using comments.

Use your programming language's constructor method.

Programming language ....

Program code

....

....

....

....

....

....

....

.... [4]

(b) The `QuizClass` method `AddQuestion()` takes a question object as a parameter and stores it in the next available location in the array `Questions`. It returns `TRUE` if it is successfully stored, and `FALSE` otherwise.

Write **program code** for the method `AddQuestion()`.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(c) The first quiz is created with the identifier `FirstQuiz`.

The first question in this quiz is: "What is 100 / 5 ?".

The answer is "20" and the difficulty level is 1.

Write **program code** to:

- declare an instance of `QuizClass` with the identifier `FirstQuiz`
- declare an instance of `QuestionClass` with the identifier `Question1`
- add `Question1` to the array in `FirstQuiz` using `AddQuestion()`.

Programming language ....

Program code

....

....

....

....

....

....

....

....

.... [5]

(d) The object `FirstQuiz` contains objects of type `QuestionClass`.

State the name of this OOP feature.

....

.... [1]

(e) Ejaz can use an interpreter and a compiler to translate program code during the development process. The program will be distributed without any access to the source code.

(i) State when Ejaz should use an interpreter and a compiler. Each answer must be different.

Interpreter ....

....

Compiler ....

....

[2]

(ii) Give the name of **two** facilities that Ejaz can use to debug his program.

1 ....

....

2 ....

....

[2]

(iii) Describe **one** feature of an editor that Ejaz can use when writing the program.

....

....

....

.... [2]

3 A college has two types of student: full-time and part-time.

All students have their name and date of birth recorded.

A full-time student has their address and telephone number recorded.

A part-time student attends one or more courses. A fee is charged for each course. The number of courses a part-time student attends is recorded, along with the total fee and whether or not the fee has been paid.

The college needs a program to process data about its students. The program will use an object-oriented programming language.

(a) Complete the class diagram showing the appropriate properties and methods.

Student
`StudentName: STRING`
....
....
....
`ShowStudentName()`
....
....
....

FullTimeStudent
`Address: STRING`
....
....
....
`Constructor()`
`ShowAddress()`
....
....

PartTimeStudent
....
....
....
....
....
....
....
....

[7]

(b) Write **program code**:

(i) for the class definition for the superclass `Student`.

Programming language ....

....

....

....

....

....

....

....

.... [2]

(ii) for the class definition for the subclass `FullTimeStudent`.

Programming language ....

....

....

....

....

....

....

....

.... [3]

(iii) to create a new instance of `FullTimeStudent` with:

- identifier: `NewStudent`
- name: A. Nyone
- date of birth: 12/11/1990
- telephone number: 099111

Programming language ....

....

....

....

....

....

.... [3]

4 A dictionary Abstract Data Type (ADT) has these associated operations:

- Create dictionary `(CreateDictionary)`
- Add key-value pair to dictionary `(Add)`
- Delete key-value pair from dictionary `(Delete)`
- Lookup value `(Lookup)`

The dictionary ADT is to be implemented as a two-dimensional array. This stores key-value pairs.

The pseudocode statement

```
DECLARE Dictionary : Array[1:2000, 1:2] OF STRING
```

reserves space for 2000 key-value pairs in array `Dictionary`.

The `CreateDictionary` operation initialises all elements of `Dictionary` to the empty string.

(a) The hashing function `Hash` is to extract the first letter of the key and return the position of this letter in the alphabet. For example `Hash("Action")` will return the integer value 1.
(Note: The ASCII code for the letter A is 65.)

Complete the pseudocode:

```
FUNCTION Hash (................................................) RETURNS ............................................

    DECLARE Number : INTEGER

    Number ← ..........................................................................................................

    ..................................................................................................................

ENDFUNCTION
```

[5]

(b) A company wants to simulate the use of a ticket machine. It will do this with object-oriented programming (OOP).

The following diagram shows the design for the class `TicketMachine`. This includes its attributes and methods.

```
                           TicketMachine
---------------------------------------------------------------------
Amount : INTEGER // total value of coins inserted in cents
State  : STRING  // "Idle", "Counting", "Cancelled"
                 // or "Accepted"
---------------------------------------------------------------------
Create()         // method to create and initialise an object
                 // if using Python use __init__
SetState()       // set state to parameter value
                 // and output new state
StateChange()    // insert coin or press button,
                 // then take appropriate action
CoinInserted()   // parameter is a string
                 // change parameter to integer
                 // and add coin value to Amount
ReturnCoins()    // output Amount, then set Amount to zero
PrintTicket()    // print ticket, then set Amount to zero
```

Write **program code** for the following methods.

Programming language ....

(i) `Create()`

....

....

....

....

....[3]

(ii) `SetState()`

....

....

....

....[2]

© UCLES 2016

(iii) `ReturnCoins()`

....

....

....

.... [2]

(iv) Each coin inserted must be one of the following: 10, 20, 50 or 100 cents.

Write **program code** for a function `ValidCoin(s : STRING)` that returns:

- `TRUE` if the input string is one of `"10", "20", "50"` or `"100"`
- `FALSE` otherwise

Programming language ....

....

....

....

....

....

....

....

....

....

....[3]

(v) Write **program code** for the method `CoinInserted()`

....

....

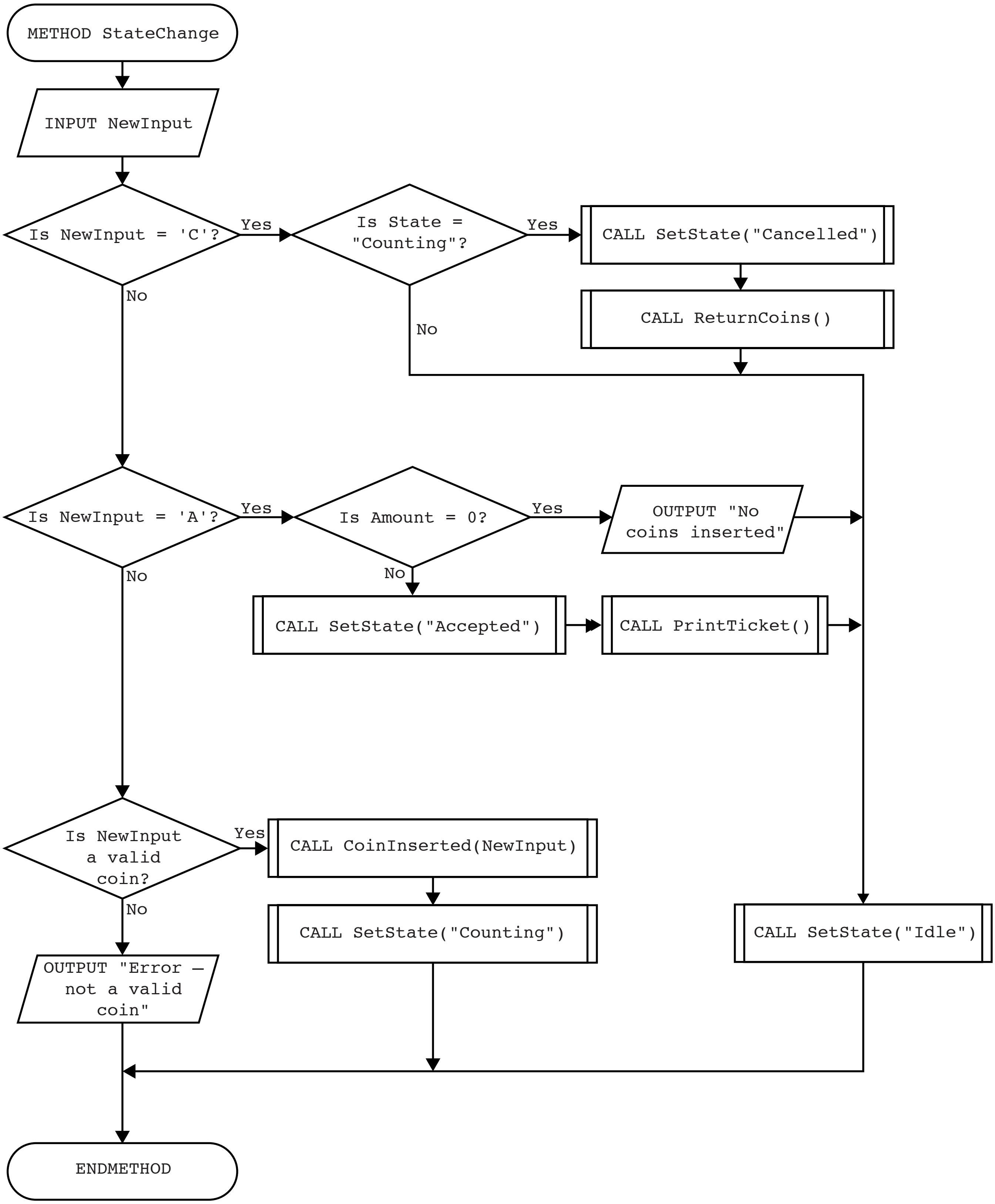
....

....

....[2]

© UCLES 2016

(vi) Convert the flowchart to **program code** for the method `StateChange()`.
Use the attributes and methods in the original class definition and the `ValidCoin()` function from **part (iv)**.



Programming language ....

....[12]

(vii) The company needs to write a program to simulate a parking meter. The program will create an object with identifier `ParkingMeter`, which is an instance of the class `TicketMachine`.

The main program design is:

```
instantiate ParkingMeter (create and initialise ParkingMeter)
loop forever (continually use ParkingMeter)
   call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language ....

....

....

....

....

....

....

....

....

....[4]

(c) It is possible to declare attributes and methods as either public or private.

A programmer has modified the class design for `TicketMachine` as follows.

```
TicketMachine
-------------------------
PRIVATE
   Amount : INTEGER
   State  : STRING
-------------------------
PUBLIC
   Create()
   StateChange()
PRIVATE
   SetState()
   CoinInserted()
   ReturnCoins()
   PrintTicket()
```

(i) Describe the effects of declaring the `TicketMachine` attributes as private.

....

....

....

....[2]

(ii) Describe the effects of declaring two methods of the class as public and the other four as private.

....

....

....

....[2]

6 A bank has a range of customer accounts, which includes current accounts and savings accounts.

All accounts have:

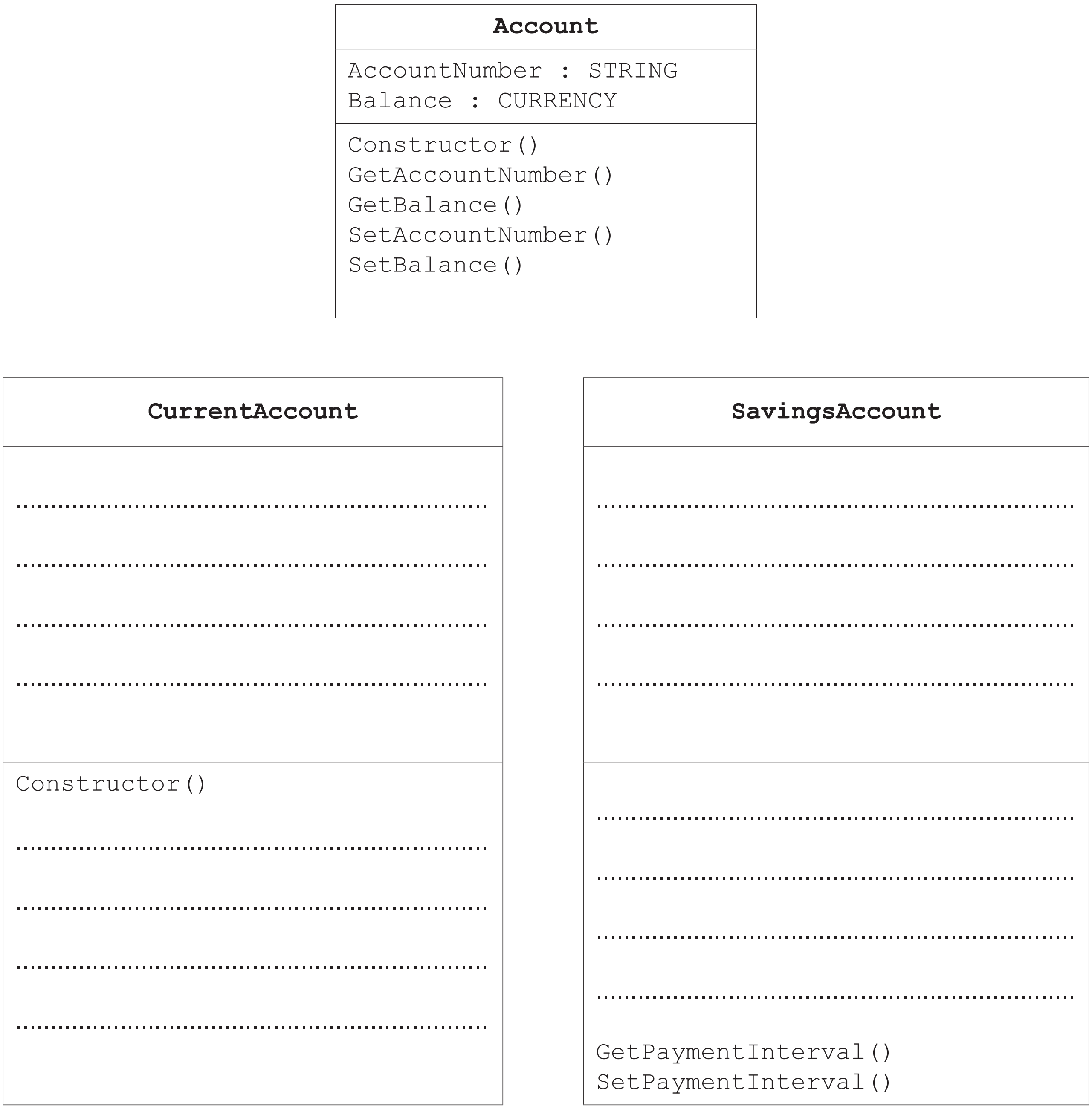
- an account number
- a balance (amount of money in an account).

A current account has a level (bronze, silver or gold). A monthly fee ($) is taken from each account.

Savings account customers pay a regular amount ($) into their account. The payment interval is a number of weeks (for example, 4).

An object-oriented program will be written to process data about the accounts.

(a) Complete the class diagram.



[3]

(b) Write **program code** to declare the `Account` class.

Programming language ....

Program code ....

....[5]

(c) Write **program code** to declare the `SavingsAccount` class. Do not write any get or set methods.

Programming language ....

Program code ....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[5]

4 A circus is made up of performers. There are three types of performer: clown, acrobat and aerial.

The following data are stored for each performer.

- First name
- Last name
- Secondary role (that can be edited)
- Stage name (that can be edited)
- Type of performer (PerfType)

The following statements apply to performers.

- An acrobat may or may not use fire in his or her act.
- An aerial performer can be one of two types: either catcher or flyer.
- Each clown has an item, such as a water-spraying flower or a unicycle.
- Each clown also has a musical instrument, such as a guitar or an oboe.

Each of the three types of performer has a method that will display all of the information about that performer in a specific format. For example:

> *Sally Superstar (real name Sally Smith) is an acrobat. Fire is part of Sally Superstar's act. When not performing, Sally Superstar is a set changer.*

(a) Complete the following class diagram to show the **attributes**, **methods** and **inheritance** for the program.

You do not need to write the get and set methods.

Performer

`FirstName : STRING`
`LastName : STRING`
`SecondaryRole : STRING`
`StageName : STRING`
`PerfType : STRING`

`Constructor()`
`EditSecondaryRole()`
`EditStageName()`

Acrobat
`UseFire : BOOLEAN`

`Constructor()`
`PerformerInfo()`

Clown
....
`Constructor()` ....

Aerial
....
`Constructor()` ....

[4]

(b) Write **program code** for the `Performer` class.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

© UCLES 2018

(c) The program will display the acrobat information as follows:

Sally Superstar (real name Sally Smith) is an acrobat. Fire is part of Sally Superstar's act.
When not performing, Sally Superstar is a set changer.

Write **program code** for the `Acrobat` class.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

© UCLES 2018

....

....

....

....

....

.... [8]

(d) Information about a performer is as follows:

> *Amazing Alex (real name Alex Tan) is an acrobat. Fire is part of Alex's act. When not performing, Amazing Alex is a popcorn seller.*

(i) Write **program code** to create an instance of an object with the identifier `Acrobat_1`.

All attributes of the instance should be fully initialised.

....

....

....

.... [3]

(ii) Explain **inheritance** with reference to the circus example.

....

....

....

....

....

....

.... [2]

(c) The company decides to implement a program for the software using object-oriented programming (OOP).

Each employee has a unique employee ID, name, address and date of birth. There are two types of employee: salary and apprenticeship.

Salaried employees are paid a fixed monthly payment. The hours a salary employee works in a month are recorded to calculate bonus payments. They may receive bonus payments and make pension payments (given in **part(b)**).

Apprenticeship employees are paid weekly. They receive an hourly rate of pay. Apprenticeship employees do not receive bonus payments or make pension payments.

(i) Complete the following class diagram for the program.

Employee

`EmployeeID : STRING`
`Name : STRING`
`Address : STRING`
`DateOfBirth : Date`

`Constructor()`
`GetEmployeeID()`
`GetName()`
`GetAddress()`
`GetDateOfBirth()`
`SetEmployeeID()`
`SetName()`
`SetAddress()`
`SetDateOfBirth()`

SalaryEmployee

`MonthlyPayment : CURRENCY`
`HoursThisMonth: REAL`
`PublicHoliday : BOOLEAN`
`Pension : BOOLEAN`

`Constructor()`
`GetMonthlyPayment()`
`GetHoursThisMonth()`
`GetPublicHoliday()`
`GetPension()`
`SetMonthlyPayment()`
`SetHoursThisMonth()`
`SetPublicHoliday()`
`SetPension()`

ApprenticeshipEmployee
....

....
....
`GetHourlyRate()`
`GetHoursThisWeek()`
`SetHourlyRate()`

[3]

(ii) Write **program code** for the `Constructor()` in the `Employee` class.

All properties are sent as parameters.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [4]

(iii) Write **program code** for the `GetEmployeeID()` method in the `Employee` class.

The get method returns the value of the `EmployeeID` property.

Programming language ....

Program code

....

....

....

....

....

.... [2]

© UCLES 2019

(iv) Write **program code** for the `SetEmployeeID()` method in the `Employee` class.

The set method takes the new value as its parameter.

Programming language ....

Program code

....

....

....

....

....

.... [2]

(v) Write **program code** for the `SetPension()` method in the `SalaryEmployee()` class.

- The method takes a new value for `Pension` as a parameter.
- If the parameter's value is valid (it is `TRUE` or `FALSE`), the method returns `TRUE` and sets the parameter's value.
- Otherwise the method returns `FALSE` and does not set `Pension`.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [4]

4 A programmer wants to create a mobile application to record the number of calories a person eats in a day.

The programmer has designed the class, `FoodItem`, to store details for each item of food.

The following diagram shows the design for the `FoodItem` class.

```
                              FoodItem
---------------------------------------------------------------------------
FoodID   : STRING  // initialised in constructor to parameter value
Name     : STRING  // initialised in constructor to an empty string
Calories : INTEGER // initialised in constructor to 0
---------------------------------------------------------------------------
Constructor()      // method used to create an instance of the
                   // FoodItem class and initialise its attributes
GetFoodID()        // returns FoodID
GetName()          // returns Name
GetCalories()      // returns Calories
SetFoodID()        // sets the FoodID to the parameter value
SetName()          // sets the Name to the parameter value
SetCalories()      // validates the parameter value to make sure
                   // it is a positive integer less than 2000, and
                   // then sets Calories to this value
```

(a) Write **program code** for the `Constructor()` method.

Use the appropriate constructor method for your chosen programming language.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [3]

(b) Write **program code** for the `GetCalories()` method.

Programming language ....

Program code

....

....

....

....

....

.... [2]

(c) The method `SetCalories()` validates the integer parameter value that is passed to it. It checks that the value is positive and is less than 2000.

The method sets `Calories` to the parameter value and returns `TRUE` if the parameter value is valid. It returns `FALSE` if the parameter value is not valid.

Write **pseudocode** for the `SetCalories()` method.

....

....

....

....

....

....

....

....

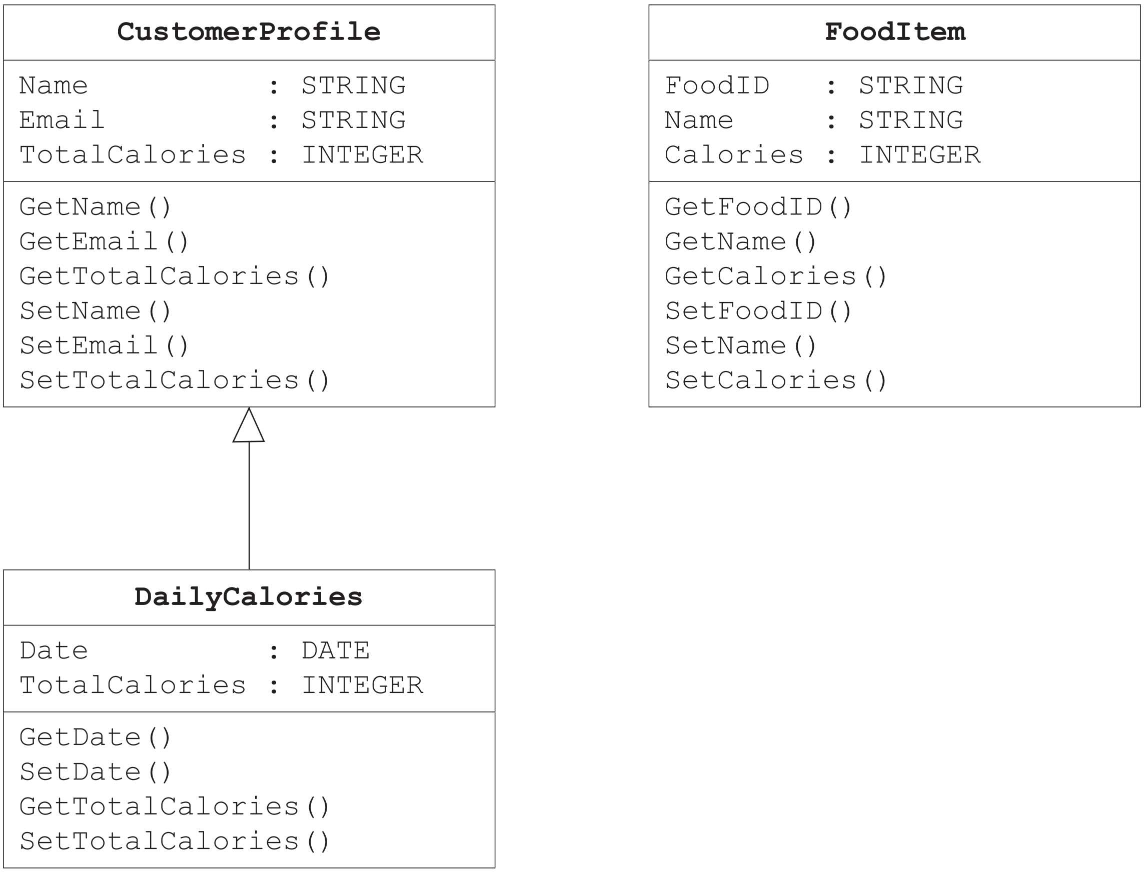
....

....

....

.... [4]

(d) The following is a class diagram for the application.



(i) The attributes of the class, `CustomerProfile`, are declared as private.

Explain why it is good practice to declare class attributes as private.

....

....

....

.... [2]

(ii) Explain what is meant by **inheritance**, using an example from the class diagram.

....

....

....

.... [2]

(iii) Explain what is meant by **polymorphism**, using an example from the class diagram.

....

....

....

.... [2]

(e) Object-oriented programming is an example of a programming paradigm. Another example is imperative programming.

Explain what is meant by the **imperative programming paradigm**.

....

....

....

.... [2]

(f) Testing is regularly performed during the development of software.

(i) Independent modules are combined to create the final program. Testing is performed to make sure they interact correctly.

Identify this type of testing.

.... [1]

(ii) Testing is performed to prove to the customer that the system works correctly and meets the requirements specified in the design.

Identify this type of testing.

.... [1]

(iii) Test plans are used when testing data. One item that would be included in a test plan is example test data.

Identify **two other** items that would appear in a test plan.

1 ....

2 ....

[2]

7 A treasure box is hidden within a computer game.

The box has a code that needs to be entered to allow the user into the box. The box contains up to 10 objects that are defined as being of the class `FieldObject`. The definition for the class `Box` is:

`Box`	
`Size : STRING`	`// small, medium or large`
`Contents : ARRAY[0 : 9] OF FieldObject`	`// the 10 items the box holds`
`Lock : STRING`	`// the code to unlock the box`

`Strength : INTEGER`	`// the strength of the box`
	`// decreases by 1 each time an`
	`// incorrect code is entered`

`Constructor()`	`// instantiates an object of the Box`
	`// class and assigns initial values`
	`// to the attributes`

`Unlock()`	`// checks if the code is correct to`
	`// unlock the box`

`GetContents()`	`// returns the array`
`SetSize()`	`// sets the size of the box`
`SetContents()`	`// sets the contents of the box`
`SetLock()`	`// sets the lock code`
`SetStrength()`	`// sets the strength`

(a) The constructor creates a new instance of a box. It takes the size of the box, one item of content and the lock code as parameters. The strength is initialised to `100`.

Write **program code** to create the class and constructor for `Box`.

Do **not** write the code for `Unlock()` or any of the set or get methods. Use the constructor for your chosen language.

Programming language ....

Program code ....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

(b) The player inputs the code to unlock the box. Each time they enter an incorrect code, the strength of the box decreases by 1. If the strength of the box becomes 0, the box automatically unlocks.

The class `Box` has a method `Unlock()` that:

- takes the code entered as a parameter and checks if it matches the code to unlock the box
- returns `TRUE` if the parameter matches the unlock code
- subtracts 1 from `Strength` if the parameter does not match the unlock code
- checks if the new value of `Strength` is less than 1
- returns `TRUE` if the new value of `Strength` is less than 1, otherwise it returns `FALSE`.

Write **program code** for the method `Unlock()`.

Programming language ....

Program code ....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

(c) The text file, `Progress.txt`, stores data about the player's previous progress.

The procedure `LoadGame()`:

- opens the text file in read mode
- takes the data from the file and stores the data in the variable `GameData`
- raises an exception with an appropriate message output if it cannot find the file.

Write **program code** for the procedure `LoadGame()`.

Programming language ....

Program code ....

....

....

....

....

....

....

....

....

....

....

....

....

.... [6]