Cambridge International A Level

Cambridge International Examinations
Cambridge International Advanced Level

COMPUTER SCIENCE **9608/43**

Paper 4 Further Problem-solving and Programming Skills **October/November 2015**

PRE-RELEASE MATERIAL

7213669108

This material should be given to candidates on receipt by the Centre.

READ THESE INSTRUCTIONS FIRST

Candidates should use this material in preparation for the examination. Candidates should attempt the practical programming tasks using their chosen high-level, procedural programming language.

This document consists of **9** printed pages and **3** blank pages.

DC (NH/SW) 111426

CAMBRIDGE International Examinations

This material is intended to be read by teachers and candidates prior to the November 2015 examination for 9608 Paper 4.

Reminders

The syllabus states:

- there will be questions on the examination paper which do not relate to this pre-release material
- you must choose a high-level programming language from this list:
    - Visual Basic (console mode)
    - Python
    - Pascal / Delphi (console mode)

The practical skills covered in Paper 2 are a precursor to those required in Paper 4. It is therefore recommended that the high-level programming language chosen for this paper is the same as that for Paper 2. This allows for sufficient expertise to be acquired with the opportunity for extensive practice.

Questions on the examination paper may ask the candidate to write:

- structured English
- pseudocode
- program code

A program flowchart should be considered as an alternative to pseudocode for the documenting of an algorithm design.

Candidates should be confident with:

- the presentation of an algorithm using either a program flowchart or pseudocode
- the production of a program flowchart from given pseudocode or the reverse

TASK 1

The taxis used by a taxi company are either cars or minibuses.

The unique registration and the charge (in $) per unit time are stored for all taxis.

All cars can carry a maximum of four passengers.

Data stored about minibuses also includes an extra charge (in $) per booking and the maximum number of passengers allowed.

The company needs software to process data about taxis.

The processing needs to include a calculation of the fare charged.

The software will be object-oriented.

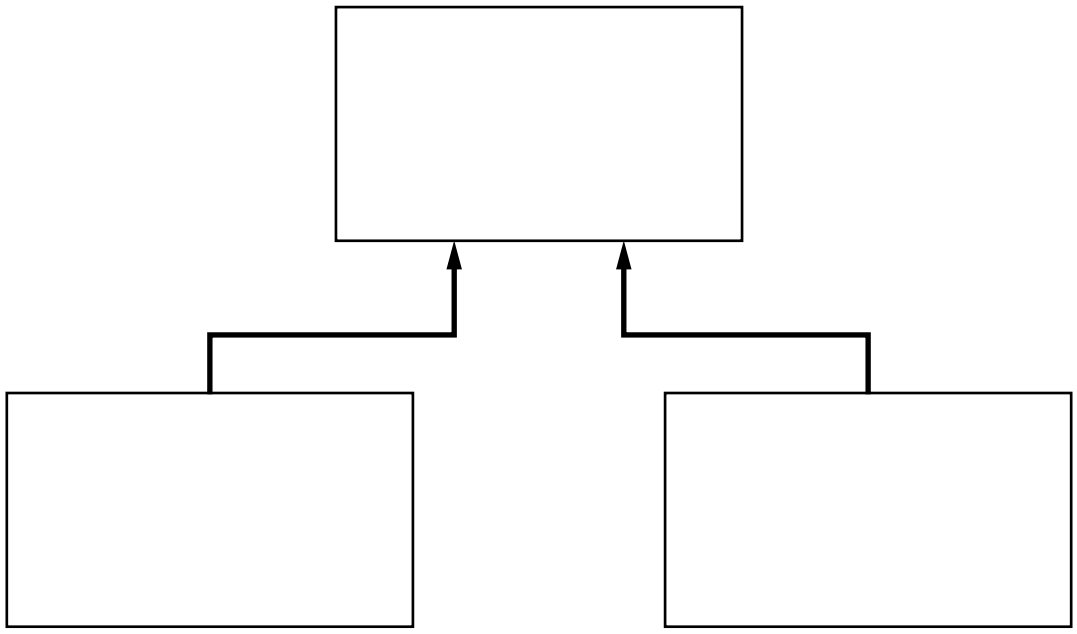
The superclass (also known as base class or parent class) `Taxi` is designed.

Two subclasses (also known as derived classes or child classes) have been identified:

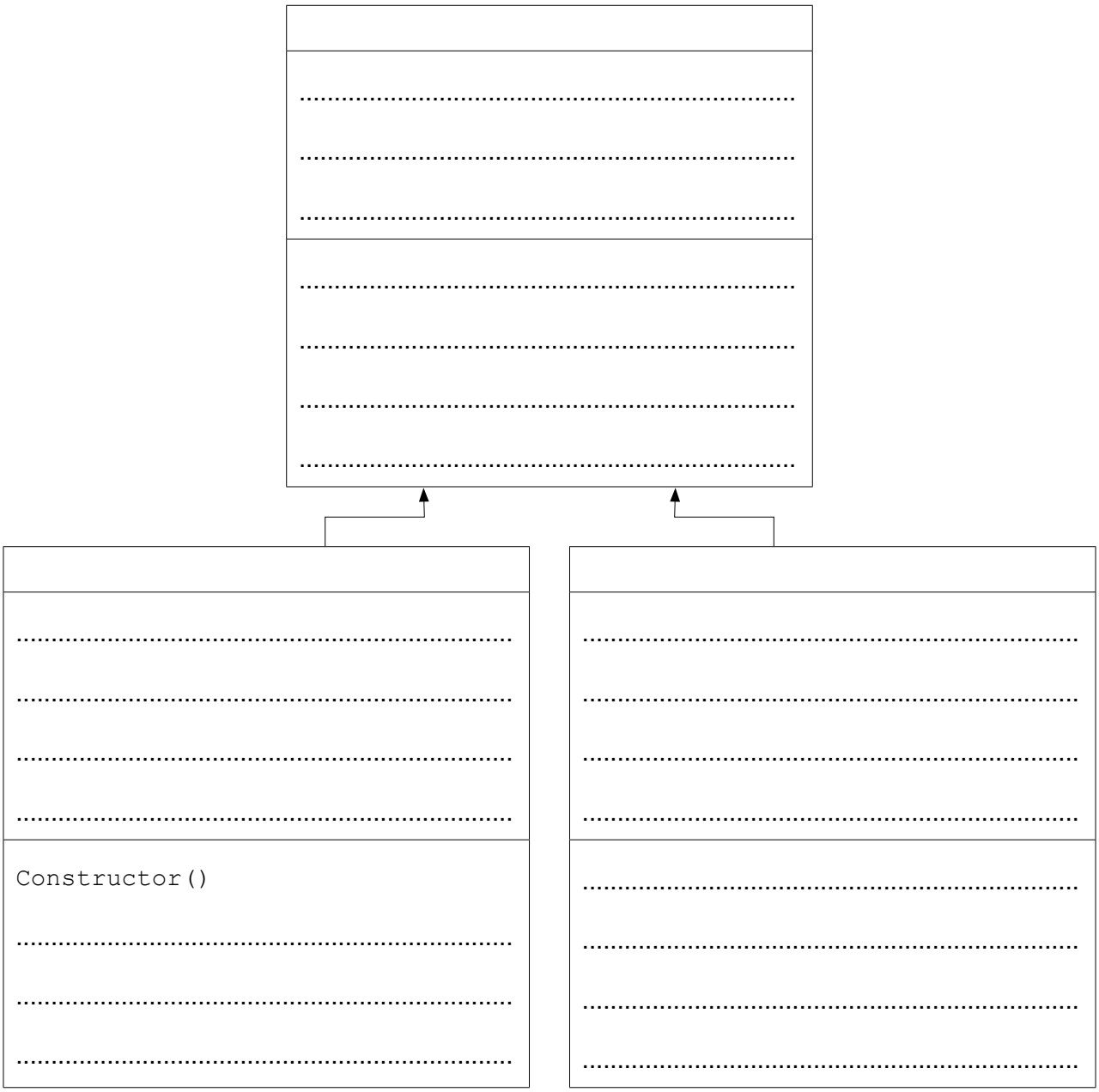
- `Car`
- `Minibus`

Key focus: Object-oriented Programming

TASK 1.1
Complete the **inheritance diagram**.



TASK 1.2
Complete the **class diagram** showing the appropriate properties and methods.



Note: a constructor is a method that creates a new instance of a class and initialises it.

TASK 1.3
Write **program code** for the class definitions. Make use of polymorphism and inheritance where appropriate.

TASK 1.4
Write **program code** to create a new instance of `Car`.

Suggested extension task
Write **program code** to display the properties of the object you created in Task 1.4.

TASK 2

Key focus: Project management using PERT and GANTT charts

A software program is to be written for a client.

A detailed program specification has been written. The program will consist of seven different menu options handled from a main module by calling seven different procedures. These modules can be coded independently.

All procedures and the main module are each estimated to take 3 hours to code and 2 hours to test.

Integration testing is expected to take 3 hours and Alpha testing 7 hours.

TASK 2.1

The project manager initially has one programmer available to write and test the program. The programmer works 10 hours a day. Calculate how many days it takes before the customer can start acceptance testing.

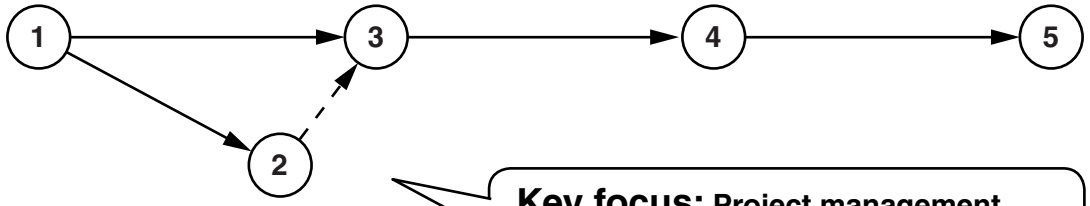
TASK 2.2

The customer is not happy about the proposed development time and the project manager considers hiring extra staff.

One proposal is to deploy one programmer and one software tester to the project.

The project manager needs a PERT chart to calculate the critical path.

Complete the diagram below.



Key focus: Project management using a PERT chart

Note: the arrow - - - → denotes a dummy activity.

Write the critical path.

Calculate the shortest time (in days) before the client can start acceptance testing.

TASK 2.3
Another proposal is to deploy four programmers, who each carry out their own module testing.

TASK 2.3.1
Draw a GANTT chart for this proposal.

Key focus: Project management using a GANTT chart

Activity

Hour number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

TASK 2.3.2
If the programmers each work 10 hours a day, calculate how many days it takes before the customer can start acceptance testing.

Suggested extension task
Try other proposals, such as only a software tester can carry out any testing.

Activity

Code P1																													
Test P1																													
Code P2																													
Test P2																													
Code P3																													
Test P3																													
Code P4																													
Test P4																													
Code P5																													
Test P5																													
Code P6																													
Test P6																													
Code P7																													
Test P7																													
Code P8																													
Test P8																													
Integration testing																													
Alpha testing																													
Hour number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

TASK 3

Data about books are stored in a random file of records.

- The key field of a book record is the ISBN (a 9-digit string and a check digit).
- Other book data are stored.
- A hashing function is used to calculate a record address.
  (The first 3 digits of the ISBN are used as the record address.)
- The random file initially consists of dummy records.
- Dummy records are shown by the ISBN set to 0000000000.

```
FUNCTION Hash(ISBN : STRING) RETURNS INTEGER
  Address ← LeftToInt(ISBN, 3)
  RETURN Address
ENDFUNCTION
```

Key focus: Random files

The `Hash` function assumes the existence of the function `LeftToInt`, defined below:

`LeftToInt(ThisString : STRING, n: INTEGER) RETURNS INTEGER`
returns an integer calculated from the `n`-digit string, starting from the left of the string `ThisString`.

An error is returned if:
- any of the first `n` characters of `ThisString` are non-digit characters
- the length of `ThisString` is less than `n`

For example: `LeftToInt("1575697620", 3)` returns the integer `157`

Note: Random files are also known as direct access files.

TASK 3.1
Write **program code** to implement the functions `LeftToInt` and `Hash`.

TASK 3.2
Write **program code** to implement the following pseudocode which initialises a random file:

```
TYPE BookRecord
   DECLARE ISBN    : STRING[10]  // need fixed-length records for
   DECLARE Title   : STRING[24]  // random files
   DECLARE OnLoan  : BOOLEAN
ENDTYPE

DECLARE DummyRecord : BookRecord

DummyRecord.ISBN ← "0000000000"

OPENFILE "BookFile" FOR WRITE  // create the file by sequentially
                               // writing dummy records to the new file
                               // prior to use as a random file
FOR Count ← 1 TO 1000
   PUTRECORD "BookFile", DummyRecord
ENDFOR
CLOSEFILE "BookFile"
```

TASK 3.3
Write **program code** to implement the following pseudocode to input 5 book records:

```
010  DECLARE NewBook : BookRecord
020
030  OPENFILE "BookFile" FOR RANDOM
040  FOR Count ← 1 TO 5
050     INPUT NewBook.ISBN
060     INPUT NewBook.Title
070     NewAddress ← Hash(NewBook.ISBN)
080     SEEK "BookFile", NewAddress
090     PUTRECORD "BookFile", NewBook
100  ENDFOR
110  CLOSEFILE "BookFile"
```

Test your program by inputting book details with ISBNs that will hash to different addresses.

TASK 3.4
You need to test that the records have been saved successfully.

Write program code to read records sequentially from the random file `BookFile` and output any non-zero records.

TASK 3.5

Key focus: Exception handling

If a program tries to open a non-existent file, a run-time error will occur.

To avoid this, you need to write exception handling code to give an error message instead.

Edit your **program code** from Task 3.3 to handle the case when `"BookFile"` does not exist.

TASK 3.6
Add a book record with an ISBN that will cause a collision (also known as a synonym).

Run your program from Task 3.4 again. What do you notice?

TASK 3.7
The pseudocode in Task 3.3 needs amending to handle a collision.

The following algorithm needs to be inserted between line `080` and `090`.

Write **pseudocode** for the following structured English algorithm:

Repeat until useable address found, or file is full:

    If there is a record with a non-zero ISBN at the address hashed

        Go to the next record address

    If the end of the file is reached, start at the beginning of the file

    If the hashed address is reached again, the file is full

Write **program code** for your algorithm and test it.

Suggested extension task
Write program code to extend your program from Task 3.5 so your program will successfully add and delete book records, even if their ISBNs cause collisions.

BLANK PAGE

BLANK PAGE

BLANK PAGE