ENGINEERING TRIPOS PART IIA

Friday 6 May 2011 2.30 to 4

Module 3F6

SOFTWARE ENGINEERING AND DESIGN

*Answer not more than **three** questions.*

All questions carry the same number of marks.

*The **approximate** number of marks allocated to each part of a question is indicated in the right margin.*

There are no attachments.

STATIONERY REQUIREMENTS
Single-sided script paper

SPECIAL REQUIREMENTS
Engineering Data Book
CUED approved calculator allowed

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

Version: 3

1 (a) In object oriented programming, describe the purpose of virtual functions. [10%]

(b) Consider how an object oriented operating system could handle file access, so that files on different media use the same interface. The UML diagram in Fig. 1 describes the `File` class and the associated class hierarchy.

The design includes a proposed class `EncryptedDisk` which transparently treats an encrypted file on disk as a normal file.

Discuss the proposed design for `EncryptedDisk` and give a design which provides encryption for all file types and will extend easily to any new file types. Include a UML diagram for your design. [30%]

(c) Extend the design to allow network support so that the system can read files on a different computer, and give a UML diagram for the new design. [30%]

(d) Give a sequence diagram for the act of reading an encrypted file from a USB device, over the network. [30%]

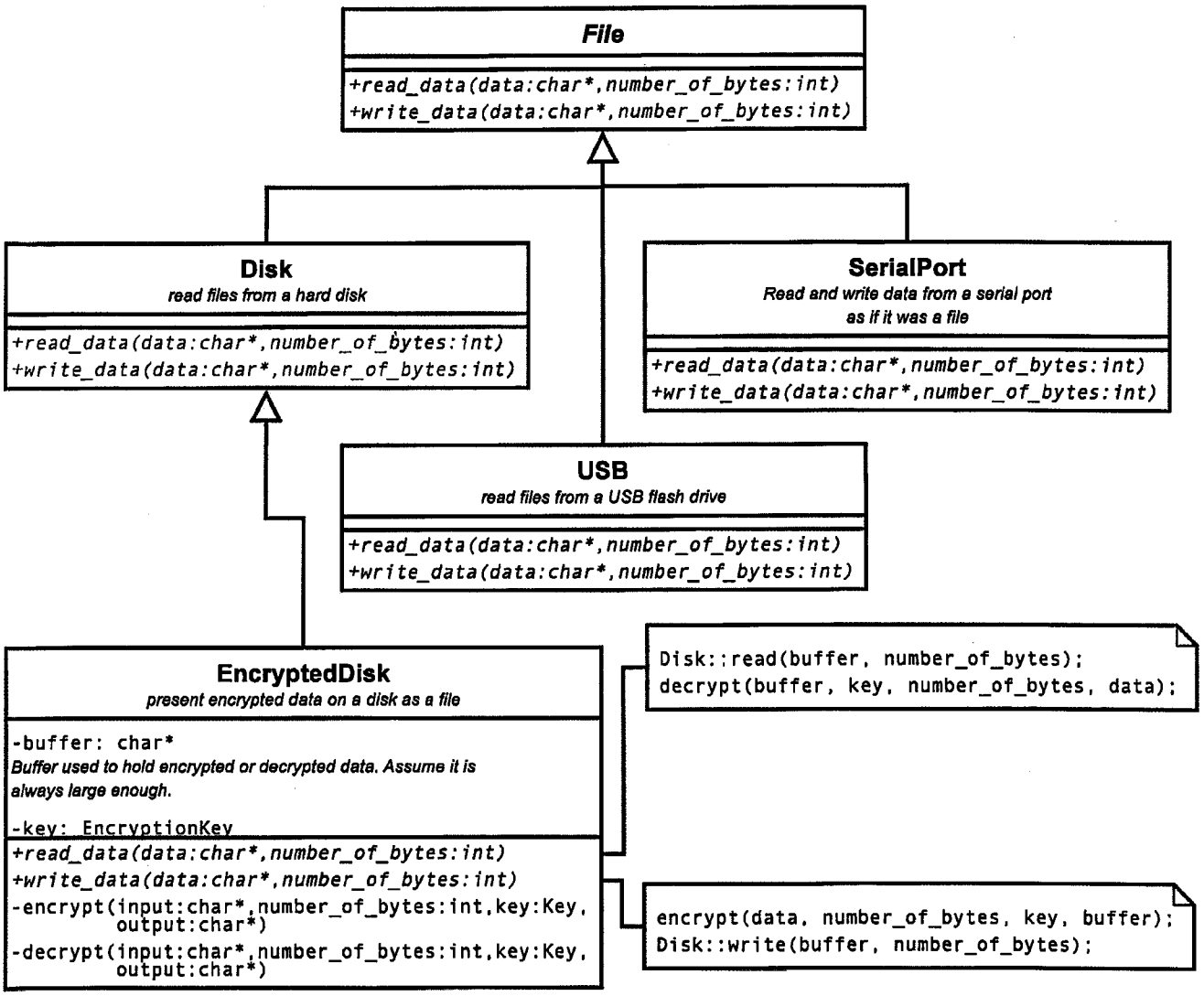
File
+read_data(data:char*,number_of_bytes:int)
+write_data(data:char*,number_of_bytes:int)
Disk
read files from a hard disk
+read_data(data:char*,number_of_bytes:int)
+write_data(data:char*,number_of_bytes:int)
SerialPort
Read and write data from a serial port as if it was a file
+read_data(data:char*,number_of_bytes:int)
+write_data(data:char*,number_of_bytes:int)
USB
read files from a USB flash drive
+read_data(data:char*,number_of_bytes:int)
+write_data(data:char*,number_of_bytes:int)
EncryptedDisk
present encrypted data on a disk as a file
-buffer: char*
Buffer used to hold encrypted or decrypted data. Assume it is always large enough.
-key: EncryptionKey
+read_data(data:char*,number_of_bytes:int)
+write_data(data:char*,number_of_bytes:int)
-encrypt(input:char*,number_of_bytes:int,key:Key, output:char*)
-decrypt(input:char*,number_of_bytes:int,key:Key, output:char*)
Disk::read(buffer, number_of_bytes);
decrypt(buffer, key, number_of_bytes, data);
encrypt(data, number_of_bytes, key, buffer);
Disk::write(buffer, number_of_bytes);

Fig. 1

2 (a) In concurrent programming, describe what a monitor class is, and what the role of signals are. [15%]

(b) Explain the terms 'critical section' and 'mutual exclusion' and show how semaphores can be used to implement mutual exclusion. [15%]

(c) A bounded buffer class allows messages to be passed from one thread or process to another. In C++ a skeleton of the class implementing the bounded buffer is:

```
class BoundedBuffer{
    public:
        void send(Message m);
        Message receive();
};
```

Using only semaphores, provide implementations for `send()` and `receive()`, and list any private variables which the `BoundedBuffer` class needs. The thread calling `receive()` must wait if there are no pending messages. The thread calling `send()` must wait if the buffer is full. [50%]

(d) An operating system does not provide semaphores and mutexes to users, but it does provide message passing using the `BoundedBuffer` class. Show how mutual exclusion for a block of code can be implemented using the `BoundedBuffer` class. [20%]

3 (a) In CORBA based distributed systems, what is an IOR, what does it contain and why? [15%]

(b) It has been decided to use CORBA to design a simple graphical windowing system which allows a computer display to be used by programs running on another computer. The computer running the display is referred to as the display server.

The display server must provide an interface which allows the client to create `Window` objects of a certain size. The `Window` object must provide a method, `setpixel` to allow the client to set a pixel to a particular value.

Give the CORBA IDL that will be needed to provide this functionality. [25%]

(c) The design needs to be extended so that the remote program can get a record of any mouse and keyboard events that occur in the window. The proposed design modifies the IDL to be:

```
enum Type {MouseButtonPress, KeyPress};
struct Event
{
    Type event_type;
    short key_or_button_number;
    short mouse_position_x, mouse_position_y;
};

interface Window
{
    ...
    sequence<Event> get_events();
};
```

What problems would be encountered when a large number of windows are open with this design. What would an alternative be which did not suffer from the same problem? Give the CORBA IDL necessary for the improved design. [30%]

(d) The system can be extended so that clients can open the same window on several display servers simultaneously. Using good design principles, show how the system can be extended by giving a UML diagram for the extended system, and describe how the `setpixel` method works when the client has a single window open on multiple displays. [30%]

4 (a) In database systems, give an example of how concurrency can cause the database state to become inconsistent. [15%]

(b) Figure 2 shows the intended actions present in three concurrent database transactions, T1, T2 and T3 operating on four resources, A, B, C and D. The database is designed to provide ACID transactions by protecting resources using locks. Therefore for a resource R, the action R.`read` automatically attempts to obtain the shared lock $R.S$, and the action R.`write` attempts to obtain the exclusive lock $R.X$. Assume that locks are held by a transaction until it commits or aborts.

Identify the first point that a deadlock occurs, and draw a resource allocation graph. [25%]

(c) Explain a strategy that can be used to recover from a deadlock. [10%]

(d) An alternative to automatically locking the database during transactions is the optimistic lock-free strategy. Explain how the lock-free optimistic strategy operates and maintains the ACID properties of transactions. [30%]

(e) Describe the sequence of actions which would happen in part 4(b) in the optimistic case and state which accounts would be updated and which transactions would have to be restarted. [20%]

Time	Action	Transaction	Time	Action	Transaction
1	A.read	1	9	C.read	1
2	D.read	3	10	A.write	2
3	B.read	1	11	B.read	3
4	C.read	3	12	commit	2
5	A.read	2	13	A.write	1
6	B.read	2	14	D.write	3
7	D.read	2	15	commit	1
8	C.write	3	16	commit	3

Fig. 2

END OF PAPER