ENGINEERING TRIPOS PART IIA

Friday 27 April 2007 9.00 to 10.30

Module 3F6

SOFTWARE ENGINEERING AND DESIGN

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

All questions carry the same number of marks.

*The **approximate** percentage 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

1 Figure 1 shows a UML class diagram for software to send commands from a controller on earth to a remote planetary explorer.

(a) Draw a UML sequence diagram showing what happens when the `do_snap()` function is called in the `Controller` object. [25%]

(b) Because the radio link between earth and the remote explorer introduces a significant time delay, it is decided to upgrade the software in order to allow multiple commands to be sent at once. A facility must also be added to the software to allow a sequence of commands to be repeated a set number of times (e.g. "Turn Right 30^o, Snap a picture" could be repeated 12 times to obtain a 360^o panoramic image).

Show what changes will be required to the software in order to support this additional functionality, drawing any modified parts of the class diagram and identifying any design patterns used. [50%]

(c) In order to check a series of commands before they are sent to the explorer, it is decided to build a simulator into the controller software. This will allow the commands to be run on the simulator which will display a 3D model of the explorer moving around on a 3D model of the real explorer's surroundings.

Show how the Abstract Factory design pattern could be used to implement this. Give your answer as a UML class diagram. You do not need to show the operation of the physics of the simulator or its graphical output. [25%]

(cont.

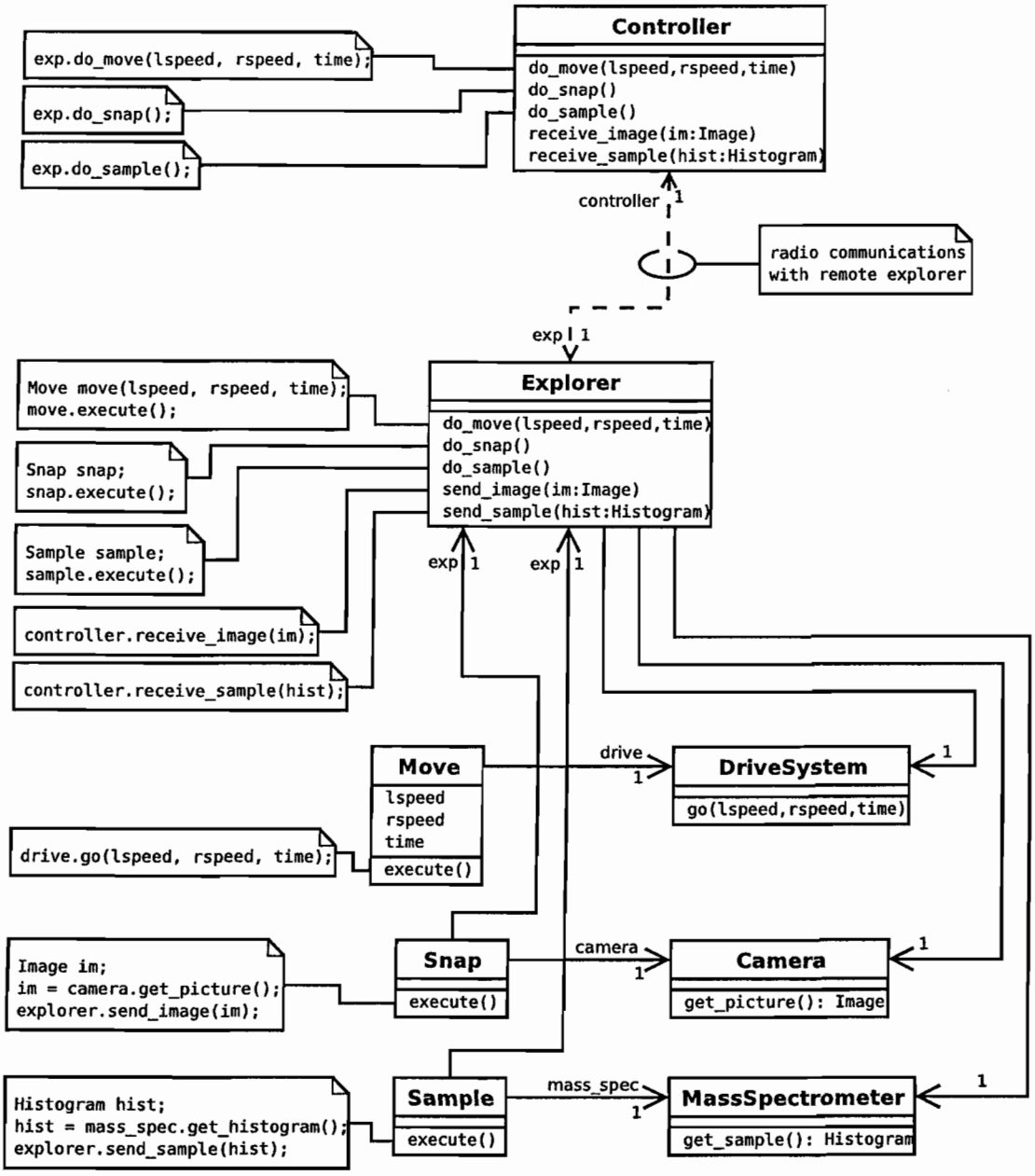

Fig. 1

(TURN OVER

2 (a) Explain how object oriented programming and design patterns can be used to:

(i) Decouple the programming interface from its implementation

(ii) Decouple the user interface from a program's internal state [20%]

(b) A traffic management authority wants a computer system to monitor and assist in managing traffic flow around a city. Part of the specification is shown in Fig. 2.

Using good design principles, draw a UML class diagram which shows the main classes that will be needed for this software, the relationships between these classes and the main attributes and operations that they should support. Identify any design patterns used. You do not need to give any pseudocode in your answer. [55%]

(c) It is now desired to use the traffic information to assist in the routing of emergency vehicles around the city. When a route is planned for an emergency vehicle, the vehicle will be sent the road sensor measurements for any road segments on its route. This will be used to warn drivers of heavy traffic or to plan an alternative route if the traffic becomes too heavy.

Draw a UML class diagram which shows what changes to your design are necessary to accomodate this additional feature. You do not need to show the internals of the route planning software which can be represented as a single class in your answer. [25%]

(cont.

Specification:

Software is required to monitor and manage traffic flow within a city. The software will obtain sensor data from road sensors and use this information to control traffic lights as well as provide a graphical display for monitoring purposes.

The system is connected to several road sensors. Once every five minutes, each road sensor will relay the average speed and the average number of vehicles on a particular stretch of road.

The traffic management system will receive the data from these road sensors and will use it to update a graphical display showing the traffic conditions on a computerised map of the city. Each road segment for which a sensor exists will be shown as a coloured line. The thickness of the line will represent the number of vehicles and the colour of the line will represent the average speed.

The system must be able to identify congested road segments. A road segment is said to be "Congested" if the average number of vehicles on the segment exceeds a specified limit and at the same time the average speed drops below a limit. These limits will be specified independently for each road segment.

When a road segment is declared "Congested" the graphical display should change to represent this.

Traffic lights in the city are split up into several zones. Each zone contains several traffic lights. The traffic lights in each zone are controlled by one of a number of policies for that zone. The policies determine the timing of the lights.

Each zone is associated with a number of road segments so that if any of them become congested, the zone is notified and can switch to using another policy. Some road segments may be associated with more than one zone, while others may not be associated with any zones.

Fig. 2

(TURN OVER

3 (a) In distributed systems, what is an Interoperable Object Reference (IOR) and what types of information does it contain? [15%]

(b) In a national immigration control system called BioCheck, every airport passenger entry channel has a computer connected to a passport scanner and fingerprint reader. Each passenger supplies a print of their left and right index fingers and their passport details. This data is then checked against a central database to verify that the prints match the supplied details. A CORBA interface definition module for the remote finger-print verification facility is shown in Fig. 3.

Given an IOR, list the steps that a client computer in an airport must take to access the `init` function in the remote BioCheck facility. [15%]

(c) Modify the BioCheck module to use the Factory design pattern to create Verifier objects and state the advantages of this organisation compared to the original. [20%]

(d) The processing of finger prints performed by BioCheck is computationally intensive. Describe by means of UML, how the implementation of your Factory BioCheck module can be designed to support load balancing. [25%]

(e) Subsequent to the original release, BioCheck discover that verification accuracy could be significantly improved by enlarging finger prints to a 128 x 128 pixel array. Show how the CORBA IDL interface could be extended to accommodate this without breaking existing applications. [25%]

(cont.

```
module BioCheck {

   typedef short FingerPrint[64][64];
   // a square array of greyscale pixel values

   enum Hand {lefthand, righthand};

   interface Verifier {
      long init(in string name, in string passNum);
      // init verifier with name and passport number
      // this call returns the transaction id

      void register(in long transaction_id, in Hand h, in FingerPrint fp);
      // register fingerprint from index finger of hand h, this must
      // be called once for each hand

      boolean verify(in long transaction_id);
      // check that registered fingerprints match passport info
   };

};
```

Fig. 3

(TURN OVER

4 (a) List three different types of code review. Describe the features that they have in common and the features that distinguish them. [25%]

(b) What information should a reviewer be provided with in advance of a review meeting? [15%]

(c) Figure 4 shows a fragment of code used in an automated car park system in which the single entrance barrier has a large 3 digit display and every parking bay has a sensor which can detect when a car arrives in the bay and when it leaves. When a car arrives at the barrier it waits until a bay number is displayed and the barrier lifts. The car then proceeds directly to the indicated bay and parks. The code is multi-threaded and entirely interrupt-driven. Hence, the three routines shown can be called at any time.

Assume that you have been asked to walk-through the code for the following use cases:

(i) car park is empty;

(ii) car park is full;

(iii) two cars leave simultaneously.

Write a code review report for these use cases. [20%]

(d) What additional failure mode would be introduced if there were multiple entry barriers? [20%]

(e) In use it is discovered that occasionally drivers ignore the instructions and park in some other vacant bay. How does the design fail in this case and how could the problem be resolved? [20%]

(cont.

```
// ------------- Car park automation system ---------------

const int NumBays = 100;  // Bays are numbered 0 to 99
int freeBays;             // number of free bays
short map[NumBays];       // bay i is free if map[i]==0
Signal notfull;           // caller blocks when carpark full

// called when car reaches entrance barrier.  Returned bay
// number is displayed and driver is allowed to proceed
int getNextFreeBay()
{
   if (freeBays==0) notfull.wait();     // if full, wait
   i = 0;                               // find next free bay
   while (i<=NumBays && map[i]) ++i;
   map[i] = 1; --freeBays;     // mark allocated bay as taken
   return i;                   // return allocated bay to user
}

// called when sensor in parking bay detects car arriving
void car_arriving(int bay)
{
   // do nothing
}

// called when sensor in parking bay detects car leaving
void car_leaving(int bay)
{
   map[bay] = 0; ++freeBays;
   notfull.send();
}
```

Fig. 4

END OF PAPER