UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2002

MSc in Computing Science
for Internal Students of the Imperial College of Science, Technology and Medicine

PAPER M225

SOFTWARE ENGINEERING

Monday 22 April 2002, 14:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

***Section A** (Use a separate answer book for this Section)*

1 **This question is about state-based specifications.**

a i) Describe the difference between *state schemas* and *operation schemas*, and explain the standard use of the four symbols ?, !, Δ, and Ξ in specifying operation schemas.

ii) Consider the following schemas, and write the schema disjunction *Div = DivOK ∨ DivError* in full, without using any schema inclusion.

```
┌─ Number ──────
│ r:R
├──────────────
│
└──────────────

┌─ DivOK ──────────────────────
│ ΔNumber
│ m?: N
│ report! = {Ok, DivbyZero}
├──────────────────────────────
│ m? > 0 ∧ r' = r/m?
│ report! = OK
└──────────────────────────────

┌─ DivError ───────────────────
│ ΞNumber
│ m?: N
│ report! = {Ok, DivbyZero}
├──────────────────────────────
│ m? = 0
│ report! = DivbyZero
└──────────────────────────────
```

b A car rental shop uses a system for handling its own car renting process. The system keeps track of the cars that are rented and the cars that are available. It allows a customer to rent one car or to return a rented car. To rent a car a customer identifies the car that he/she wants to rent and the number of days, and receives in return a confirmation message and the outstanding cost to pay. A car can be rented only when it is available. To return a car, the customer again provides the car to be returned and the number of rented days. When a car is returned, the system automatically updates the account of the shop. The system also allows the addition to the shop of new cars for rental. The following schema *CarRentalShop* defines the basic parameters of the system.

```
┌─ CarRentalShop ──────────────────────────────────────────────────
│ fullstock: FCARS        (the set of all the cars owned by the shop)
│ CC: CARS → N            (it gives the cubic capacity (cc) of each car)
│ price: CARS → R         (it gives the rental price per day of each car)
├──────────────────────────────────────────────────────────────────
│ ∀x: CARS. (price(x) = CC(x)/100)
└──────────────────────────────────────────────────────────────────
```

i) Write a state schema *SystemState* for the overall state of the system, covering the cases when the system is open and when it is closed. The system is closed when there are no more cars available and open otherwise. In both cases, the schema includes *CarRentalShop*, uses the variables *account* (the shop's account), *CarsAvailable* (the set of cars available) and *CarsRented* (the set of cars rented). Include relevant constraints on the variables.

ii) Write operation schemas for the following operations using a defensive approach. All the schemas include the inputs *car?* and *days?*, and the output *report!*.
Marks will be awarded for correct use of Δ and Ξ.

RentCar returns the outstanding cost to pay, if the requested car is available. It returns the error "CarNotAvail", when the car is not available and the system is open, and the error "NoCarsAvail" when the system is closed.

ReturnCar updates *account* if the input *car?* is currently being rented. Otherwise it returns an error.

iii) Write the operation schema *AddNewCar*, which takes the input *newcar?* and updates the basic parameters of the system.

The two parts carry, respectively, 25% and 75% of the marks.

2 This question is about object-oriented specifications using ObjectZ

a i) Define what the features of a class schema are, and describe the difference between a *state schema* and an *initial schema* of a given class schema.

ii) Consider the following two operation schemas of a class schema "company". Write in full the operation expression *Employment = comp1. SackEmployee || comp2.RecruitEmployee*, where *comp1* and *comp2* identify two objects of the class schema company.

```
__ RecruitEmployee ________
| Δ(staff)
| name?: seq char
|__________________________
| staff' = staff ∪ {name?}
|__________________________
```

```
__ SackEmployee ________
| Δ(staff)
| name!: seq char
|__________________________
| name! ∈ staff
| staff' = staff - {name!}
|__________________________
```

b A travel agency uses a system for controlling flight bookings. The system uses a class *Passenger* and a class *Seat*. Each seat has a *flight number* (of type *seqchar*) and a *status*, which can be either free or booked. Each passenger has a *name* (of type *seqchar*), a *maximum number* of seats that he/she can book, and an associated finite set of booked seats, denoted by the attribute *seatsbooked* (of type *FSeat*). Initially, passengers have no seats booked and seats have the status free. The system includes as attributes a finite set of *passengers* and a finite set of *seats*. The booking operation takes as input a passenger and an individual seat. The seat is booked for that passenger when the seat is free and the number of seats booked by that passenger is less than the maximum number of seats the passenger is allowed to book. If one of these two conditions is false, the booking operation is denied.

Write an ObjectZ class schema for each of the following classes, using the local control approach to specify the association between a passenger and the seats booked by the passenger. Include in each class schema an appropriate visibility list, state schema, initial schema and the operation schema(s) as described below:

i) The class *Passenger* includes two operations *BookedOK* and *BookingDenied*. The first operation adds the new seat to the set of seats booked by the passenger if the seat is currently free and the passenger hasn't yet reached the allowed maximum number of seats he/she can book. The operation *BookingDenied* expresses only the cases when the seat cannot be booked.

ii) The class *Seat* includes one operation called *Reserved* that changes the status of the seat from *free* to *booked*. It should also include an operation for providing as output the "self" identity.

iii) The class *BookingSystem* includes one operation called *BookingaSeat*, which takes two input variables of type Passenger and Seat, respectively. This operation either books the seat for the passenger and makes the seat reserved, or it performs the operation *BookingDenied*.

The two parts carry, respectively, 30% and 70% of the marks.

3 This question is about object-oriented specification and program verification

a i) Briefly describe the meanings of *pre-condition* and *post-condition* statements.

ii) Describe the use of the subscript 0 (for variable names, e.g. x_0) in the pre-conditions of a method, and the use of the variable *result* in the post-conditions of a method.

b The pay office of a company uses an automatic system for handling the salaries of its employees. One module of this system, called *Wage*, deals with the weekly employee time card. This includes a fixed number of *standard* working hours per week, which is equal to 40 and a fixed regular pay *rate* of £8.25 (for the standard hours). It also includes an operation, called *calcGrossWage*, for calculating the weekly gross wage. This operation takes as input the number of hours worked in a week, by an employee, and returns the *grossWage* as output. Note that employees are paid one and an half the regular pay rate for all hours worked over 40.

i) Write a class schema *Wage* in ObjectZ that specifies the description given above. Include appropriate attributes and the operation *calcGrossWage*. This operation has the input variable *hours?* (of type **N**) and one output variable *grossWage!* (of type **R**). Only the operation *calcGrossWage* should be visible, all other attributes should be private to the class.

ii) Map the ObjectZ class schema *Wage* you have given in part (i) into a Java class *Wage*, by including the declaration of the data fields of this Java class, the method header and its pre- and post-conditions.

iii) Implement the method of the Java class *Wage* you have given in part (ii), including suitable mid-conditions, and argue carefully that the code you give satisfies the post-condition of the method.

The two parts carry, respectively, 25% and 75% of the marks.

Section B *(Use a separate answer book for this Section)*

4 In this question you may define any additional classes and methods you feel are useful.

a Explain the concepts of high-cohesion and low-coupling and show using examples how they are used in software design.

b An art gallery exhibits paintings and sculptures. Both are linked to displacement sensors, which are triggered if the objects are moved. All exhibits (paintings and sculptures) are shown in rooms and every room has one or several motion sensors, which can also be triggered. Each room has an alarm, which is notified when either motion or displacement sensors are triggered. The alarm is connected to a central security system, to which all alarms are notified. When notified of an alarm, the security system creates an alarm report containing the room number in which the alarm occurred and a description of the exhibits that are affected. When a displacement sensor is triggered, only the exhibit linked to that sensor is considered to be affected. When a room motion sensor is triggered, all exhibits in that room are considered to be affected. The security system then appends the alarm report to a log, identifies the closest security guard to the scene of the incident and notifies her on her pager. Assume that objects representing security guards have an attribute indicating the guard's current location.

i) Draw a class diagram of the system's architecture. Indicate the nature and cardinality of all relationships, any abstract classes and indicate which attributes and methods you would expect each object to provide.

ii) In no more than 15 lines comment on your design and discuss any alternatives.

c i) Give a collaboration diagram indicating the invocations occurring when a sensor connected to a painting is triggered. Indicate sequence numbers and transient links if any.

ii) Discuss your design in no more than 15 lines showing what decisions you have made to reduce coupling and increase cohesion.

The three parts carry, respectively, 20%, 40%, 40% of the marks.