UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2003

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

PAPER M225

SOFTWARE ENGINEERING

Wednesday 7 May 2003, 15:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

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

1 Throughout this question state clearly any additional assumptions made.

a i) Explain why low-cohesion and high-coupling are undesirable in software design?

ii) Give three examples of guidelines that can be used in assigning responsibilities to classes in order to reduce coupling and increase cohesion.

b SWD is a software consultancy company that undertakes systems development contracts for various clients. Each contract has a start date and an end date. SWD employs Analysts, Architects and Project Managers which are allocated to the various contracts for a duration which may be less than the duration of the contract. Duration is specified as a start date and an end date. Contracts can be software design contracts, implementation contracts, or systems development contracts. Implementation contracts are outsourced to an external provider and systems development contracts may comprise several subcontracts. Each employee has a daily rate which determines how much SWD should invoice clients for their work. Assume that an employee cannot be simultaneously assigned to two contracts. SWD needs to design a system to manage employees and contracts. In this system a *ContractManager* class maintains references to all the contracts while a *HRManager* class maintains references to all the employees. The system must be able to: *i)* determine which employees are free between a given start date and end date and *ii)* periodically generate invoices for those contracts where the end date has passed. The invoices must specify the names of the employees, the period (start date and end date) during which they have worked on the contract and the total cost of their involvement.

i) Draw a class diagram showing a possible software design for the system described above, including all the necessary attributes and methods.

ii) Briefly explain how the following tasks would be implemented in your system, showing which classes would be responsible for which tasks and why this responsibility would be attributed to them.

- Creating an instance of a contract.
- Identifying the period for which a specific employee has worked on a specific contract.
- Listing all the employees working on a particular contract
- Printing all the invoices

c When a new contract is created, the system must find a Project Manager available for the entire duration of the contract and assign him/her to the contract for the entire duration of the contract.

i) Draw a collaboration diagram indicating the invocations which occur in the system. Indicate any transient links.

ii) Briefly discuss any alternatives.

The three parts a, b &c carry, respectively, 20%, 50%, 30% of the marks.

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

2 This question is about state-based schemas.

a A firm deals with employees and projects. There are three projects in the firm called P1, P2, P3. Employees can be assigned to these projects. Employees have ranks, salaries, next of kin and ages. The only ranks in the firm are "programmer" and "manager". The following constraints are specified on employees and project allocations:

- Each employee must always have exactly one non-zero salary, exactly one age, exactly one rank, and at least one next of kin.
- No employee can be under 18 years.
- No programmer is assigned to project P3.
- The initial salaries of a manager and a programmer are £40,000 and £30,000, respectively.

Assume that there is a sort "PERSON" and that salaries and ages are non-negative integers.

Devise state-based (and, where necessary, initial) schemas for the employees and the firm including the project assignments as defined above. Make sensible use of schema inclusions and make sure that all the constraints are specified formally.

b Devise operation schemas for the above firm to specify the operation of assignment of an employee to a project. Give two versions of these, one with preconditions (non defensive) and one with exception handling (defensive). The inputs to the operation are an employee and a project.

The two parts carry, respectively, 60%, 40% of the marks.

3 This question is about class-based schemas.

a i) What are class attributes in class schemas? How do they differ from one another?

ii) What is the difference between aggregation and inheritance in class-based schemas?

b Estate agents have a fixed commission rate ranging between 1% and 3%. (This is the fee of the estate agent computed with respect to the sale prices of properties it sells.)

Estate agents maintain two disjoint lists of properties. One list (called *For-Sale*) consists of the list of properties they have available for sale, each with its asking price. The other list (called *Sold*) consists of all the properties they have sold already, each with its actual sale price (the sale price of a property may be different from its asking price).

Estate agents also maintain a variable *Total-Fees* that records the sum of all the commission that they have charged on all the properties they have sold.

Estate agents have the following operations:
Add-Property: which adds a given property with a given asking price to the *For-Sale* list of the agent.
Delete-Property: which deletes a given property from the *For-Sale* list of the agent and outputs its asking price.
Sell: which sells a property, i.e. for a given property it deletes the property from the *For-Sale* list of the agent and adds it (with its given sale price) to the *Sold* list of the agent and updates the agent's *Total-Fees* accordingly.

i) Devise a class schema called *Estate-Agent* according to the requirements defined above. Do not give the axioms of the state schema and do not give the initial schema, but give all the other components of the class schema. You may assume a type "Prop" for property.

ii) Give a class schema for a Chain-of-Estate-Agents consisting of a collection of estate agents. You need to give only the state schema and the operations of the chain. The chain has the following operations. Define them all by the use of promotion of operations and object interactions. You should not give full schemas for any of the operations.

Add-to-some: which adds a given property with a given asking price to the *For-Sale* list of a given estate agent in the chain.
Transfer: which transfers a given property (with its asking price) from the *For-Sale* list of a given estate agent to another in the chain.
Add-to-two: which adds a given property with a given asking price to the *For-Sale* lists of two different given estate agents in the chain.

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

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

4a i) Briefly describe the difference between the *local control* and *central control* approaches for specifying associations between objects in Object-Z.

ii) Explain briefly the circumstances under which a **for** loop would be preferred over a **while** loop in implementing a specification.

b The supersecret government intelligence agency MI7½ is declassifying some historical documents for release to the public. The *Archive* module of MI7½'s computer system is used to manage secret documents. It contains two constants, a *secrecyThreshold* and a *storageInterval*, and an operation *CheckRelease*. The operation takes as input the *secrecyLevel* of a document and its *age* in years, and returns a true/false value *okToRelease* of type boolean that indicates whether or not the document is safe to release. A document must not be released if its *age* is less than its *secrecyLevel* times the *storageInterval*. In addition, if a document's *secrecyLevel* is greater than or equal to the *secrecyThreshold*, it must not be released whatever its age. Otherwise, the document is safe to release.

The following Object-Z schema formalises the preceding description:

Archive

↓(CheckRelease)

secrecyThreshold: **N**
storageInterval: **N**

---

secrecyThreshold = 5
storageInterval = 10

CheckRelease

secrecyLevel? : **N**
age? : **N**
okToRelease! : **B**

(age < secrecyLevel? * storageInterval) ∨
(secrecyLevel? >= secrecyThreshold) ⇔ okToRelease! = false

i) Write a skeleton for a Java class *Archive* that corresponds to this schema, giving the declaration of the class attributes, the header for the *CheckRelease* method, and its pre- and post-conditions.

ii) Implement the *CheckRelease* method of part i), giving suitable mid-conditions, and prove that your code satisfies the method's post-condition.

c Consider the following Java code:

```
public abstract class Bureaucracy {
    public static int x;

    // modifies: x
    // pre: none
    // post: x = x0 + 1
    abstract void oneStepForward();

    // modifies: x
    // pre: none
    // post: x = x0 – 2
    abstract void twoStepsBack();

    // pre: none
    // post: x = x0
    public void doNothing() {
        oneStepForward();
        oneStepForward();
        twoStepsBack();
    }
}
```

Prove that the given implementation of *doNothing* satisfies the method's post-condition. You may assume that the methods *oneStepForward* and *twoStepsBack* are already implemented correctly.

The three parts carry, respectively, 25%, 50%, and 25% of the marks.