IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS [illegible]

[illegible]

SOFTWARE ENGINEERING 2

[illegible]

[illegible]

Answer [illegible]

Corrected Copy

[illegible]

[illegible]

1a Briefly explain (in no more that 15 lines) the concepts of *abstract class, abstract operation (method)* and *association (with) class.*

b *SafeBooking* is a travel booking system that caters for complex journeys. In SafeBooking a complex journey may be composed of any number of segments which can be either a simple journey or a complex journey. Simple journeys can be either transportations, or stopovers. Each simple journey has a start date and an end date. Transportations also have a departure time and arrival time, as well as an origin and a destination. Each stopover has a location. Each simple journey is associated with at most one booking made with a supplier, and each booking covers a single simple journey. Bookings can be either accommodation bookings or transportation bookings. All bookings have a start date and an end date. Transportation bookings also have a departure time and an arrival time, as well as a departure and a destination location. Accommodation bookings have a location.

Draw a UML class diagram showing the system described above. Show classes, their relationships and their attributes. Indicate the cardinality (multiplicity) of associations and any classes that are abstract.

c Assume that the Java classes implementing bookings exist. Write a Java implementation for the journey classes which permits to check that booking details correspond to the journey details. For a complex journey this requires checking all its segments. For simple journeys this requires checking start and end dates, departure and arrival times and locations. Furthermore, you should check that transportations are only associated with transportation bookings and that stopovers are associated only with accommodation bookings.

Indicate the classes, their attributes and any checking functions, but constructors are **not** required. Indicate classes or methods that are abstract.

You may use the following Java operators:

```
Object instanceof Class -> Boolean
// returns true if object o is an instance of
// class c and false otherwise

Super.method() // calls the with name "method" in the superclass.
```

For questions 1b and 1c assume that dates are encoded using a class `Date`, locations are encoded as strings and time is encoded as the number of minutes since the beginning of the day. Assume all attributes are *public*.

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

2a A bank account system is used to manage bank accounts which can be accessed by several customers. Accounts can be either: *(i) Interest Accounts* where the bank pays interest on the current balance which must remain positive or *(ii) Mortgage Accounts* where interest is added to the amount payable to the Bank and which are associated with a singe collateral which has a value. The system is flexible so that it can be used with multiple implementations of the various accounts.

For mortgage accounts a Bank can set the collateral associated with a mortgage account and link a mortgage account to any number of interest accounts. A collateral is denoted by an interface containing a `getValue()` operation which returns its value.

A customer can perform only the following operations on an account:

```
void  payIn(float sum)       // pay in a sum of money
float withdraw(sum float)    // withdraw a sum of money, returns the sum
float getBalance()           // returns current balance
float get Rate()             // returns the current interest rate
```

A bank can perform only the following operations on an account:

```
float updateBalance()  /* performed daily, updates the balance
                          with the daily interest */
float calculateRate(float profit) /* calculates(and returns)an
                        interest rate given a profit margin */
float getBalance()     // returns current balance
```

Draw a UML class diagram showing the bank and customer classes and all the interfaces needed. Indicate the functions of each interface and show all relationships between the entities including dependencies.

b All banks have a common yearly *base-rate* which is shared by all banks and each bank has its own profit-margin target. The interest rate on an interest account is equal to the base-rate minus the profit-margin. The interest rate on a mortgage is the sum (S) of the base-rate and the profit-margin increased by a risk premium (R). R is equal to S multiplied by the ratio between the account balance and the value of the collateral. In the case of a mortgage, the interest rate is applied to the balance of the account reduced by the sum of all the balances in the linked interest accounts. Daily interest is equal to the yearly interest rate divided by 365.

Design the classes which satisfy the interfaces written in question 2*a* and implement the functionality described above. You should:

i) Draw a UML class diagram specification of the classes together with the interfaces they implement. Indicate classes, interfaces and their relationships but do not indicate operations.

ii) Write the Java implementation for these classes. Show attributes and functions of classes but **not** the constructors.

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

3a Briefly explain (in no more than 15 lines):

i) the role of *statechart diagrams, class diagrams* and *interaction diagrams* in UML stating what they can specify and why they are complementary,

ii) The differences between an *activity* and an *action* in UML statecharts.

b Consider the following aspects of a mobile-phone:

When switched on, a mobile phone enters a *start-up* state where it starts by attempting to detect a network on which it can log-on.

When logged onto a network, it enters its usual state of operation (also called *Ready* state) in which it continuously polls the network. In this state, any numerical key-press starts dialling a number. While dialling a number each numerical key-press adds the key to the dialling buffer and pressing *cancel* removes the last key from the buffer. Pressing *cancel* when the buffer is empty, will return the phone to the Ready state.

Pressing *call* in the dialling stage will attempt the connection. If successful, the phone enters a *talking* state and if unsuccessful, the phone returns to the *Ready* state. While *talking*, pressing the *callEnd* button will return the phone to the *Ready* state.

While the phone is either *Ready*, dialling a number, or connecting, an incoming call will display the caller's number and the phone will continuously ring. Pressing *call* will accept the incoming call while pressing *callEnd* will return the phone into whatever state it was in when the incoming call arrived. In this latter case, the caller's number is added to the *missedCalls* list.

Loss of signal while dialling a number, connecting or while in *Ready* state will cause a re-start of the phone and network detection.

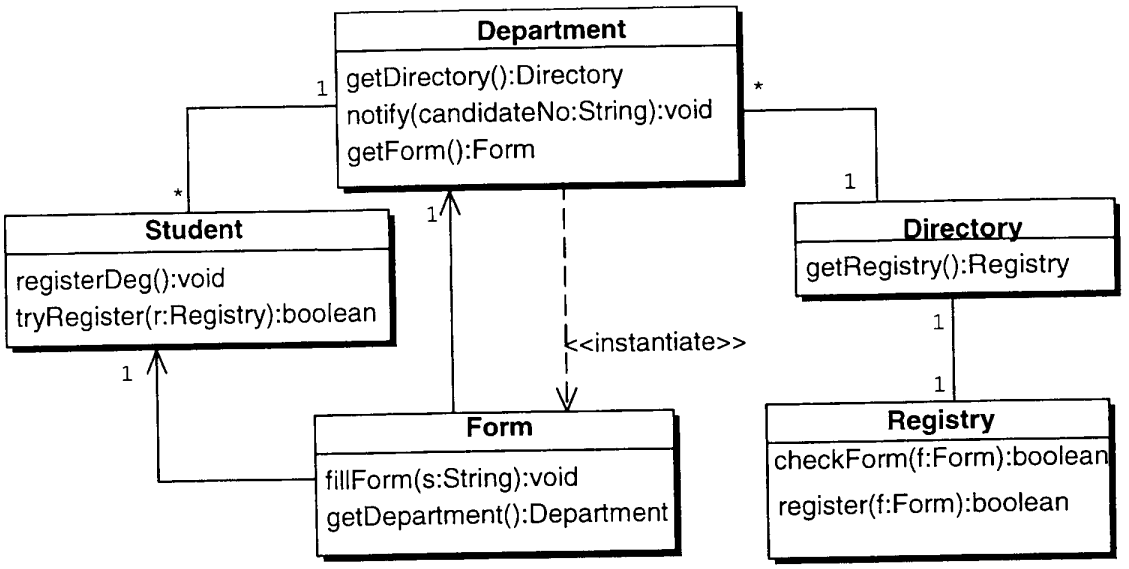
Draw a UML state chart diagram for the system above.

c The phone is powered by a battery and will switch off if the battery is empty or if the on/off button is pressed. If the phone detects a continuous input current (placed on the charger) it will start charging. The phone will revert to being battery powered when removed from the charger.

Considering your answer to question 3b as a composite state, draw a UML statechart diagram of the phone together with its battery. Also show the changes triggered by its on/off switch.

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

4a A university department has a number of students who must register at the beginning of every year with the central college registry. The UML class diagram below represents the relationships between classes in the system and the operations of each class.



To register with the central registry (*registerDeg* operation) a student must first obtain a reference to the central directory from the department it is associated with, and then obtain a reference to the registry from the directory (*getRegistry*). The student then tries to register by repeatedly invoking its own *tryRegister* operation until this operation returns `true`. In the *tryRegister* operation, the student first obtains a new form from the department (*getForm*) and fills in the form (*fillForm*). The student then tries to submit this form to registry by calling the *register* operation on the registry object.

The registry first checks the form (*checkForm*). If this operation returns true it will first query the submitted form to obtain the department concerned (*getDepartment*) and will then notify the department with a new candidate number (*notify*), otherwise it will return `false` as the result of the operation.

Note that when *getForm* is called, the department creates a new instance of a form.

Draw a UML sequence diagram for the *registerDeg* operation of the student class. Indicate message sequence numbers and parameters where known.

b Draw a UML collaboration diagram for the *registerDeg* method. Indicate which objects and links are transient. Message sequence numbers, parameters and return values are not required.

c Briefly explain (in no more than 15 lines) the differences and similarities between sequence and collaboration diagrams and why they are both useful. Use your answer to questions 4a and 4b to support your argument.

The three parts carry, respectively, 50 %, 30%, 20% of the marks.