Question 1 – Software Engineering in General (10 Marks):

1. Change cannot be easily accommodated in most software systems, unless the system was designed with change in mind.
   a. **True**
   b. False

2. Software Engineering is best described as:
   a. the practice of designing, building, and maintaining off-the-shelf software from prefabricated parts.
   b. the practice of designing, building and maintaining ad-hoc software without the use of formal methods.
   c. **the practice of designing, building and maintaining reliable and cost-effective software using standard techniques.**
   d. the practice of designing, building and maintaining fast and flexible software specifically for Engineering applications.
   e. the practice of designing, building and maintaining flashy, cheap and buggy software engineered to generate large initially sales and an on-going market for updates.

3. System maintenance is necessary because:
   a. Humans never get it right the first time.
   b. The deployment platform may change over time.
   c. The user's needs may change over time.
   d. **All of the above.**
   e. None of the above.

4. In requirements validation the requirements model is reviewed to ensure its technical feasibility.
   a. True
   b. **False**

5. A software process model is:
   a. A representation of the way in which software is used
   b. A representation of the way in which software may fail
   c. A representation of the way in which software processes data
   d. An attractive young person used in the process of selling software
   e. **A representation of the way in which software is developed**

6. In an analysis of some of the life cycle models, we can conclude that the ____________ model is the best.
   a. Waterfall
   b. Spiral
   c. XP
   d. Formal method
   e. Re-use
   f. **None of the above**

7. The five general phases in the Spiral model are:
   a. Analysis, Design, Engineering, Testing, and Payment
   b. Analysis, Design, Implementation, Testing, and Review
   c. Review, Decision, Engineering, Acceptance, and Planning

d. Review, Risk-analysis, Design, Implementation, and Planning
e. Review, Risk-analysis, Prototyping, Engineering, and Planning

8. Which of the following is not a disadvantage of the prototyping process model?
   a. Assumes a linear development approach
   b. Document is often sparse or completely absent
   c. Not really a complete development methodology
   d. Product is not a complete system but may be treated as such by management

9. Requirements specifications that are written in a natural language such as English tend to have problems that include
   a. Contradictions
   b. Ambiguities
   c. Omissions
   d. All of the above
   e. None of the above

10. A requirements specification is:
    a. A rough list of things that the proposed software ought to do.
    b. A formal list of things that the proposed software must do.
    c. A precise list of things that the proposed software ought to do.
    d. A mathematical specification of the exact behavior of the proposed software.
    e. An estimate of the resources (time, money, personnel, etc.) which will be required to construct the proposed software.

Please complete the following table; please provide <u>a single best answer from choices</u>.

MC Question	Your Best Answer (1 Mark each)
1	**a**
2	**c**
3	**d**
4	**b**
5	**e**
6	**f**
7	**e**
8	**a**
9	**d**
10	**b**

Question 2

The City Bank is developing a new software system, allowing their bank operators to

1. Register a new account for customers (function: ***registerAccount***)
2. Search for an existing account (function: ***searchAccount***)
3. Because it is a US bank, it needs to display total account balance in USD (function: ***calcullateBalance_USD***)

Please study the following code listings for class **Account** and class **AccountController.**

```java
import java.util.*;

public class AccountController {

  private final List<Account> account_list = new ArrayList<Account>();

  public void registerAccount(String name, String address, double balance) {
    Account account = new Account();
    account.setName(name);
    account.setAddress(address);
    account.setBalance(balance);
    account_list.add(account);
  }

  public Account searchAccount(String acc_name) {
    for (int i = 0; i < account_list.size(); i++) {
      if (account_list.get(i).getName() == acc_name)
        return account_list.get(i);
    }
    return null;
  }

  public double calculateBalance_USD() {
    double sum_HKD = 0;
    double sum_USD = 0;
    for (Account acc : account_list) {
      sum_HKD = sum_HKD + acc.getBalance();
    }
    sum_USD = sum_HKD / 7.8;
    return sum_USD;
  }
}

public class Account {
  private String _name;
  private String _address;
  private double _balance;

  public void setName(String name) { _name = name; }
  public void setAddress(String address) { _address = address; }
  public void setBalance(double balance) { _balance = balance; }

  public String getName() { return _name; }
  public String getAddress() { return _address; }
  public double getBalance() { return _balance; }

}
```

Question 2a - Roles of Variables (5 Marks)

Table 1. Roles for Variables in Software Programs

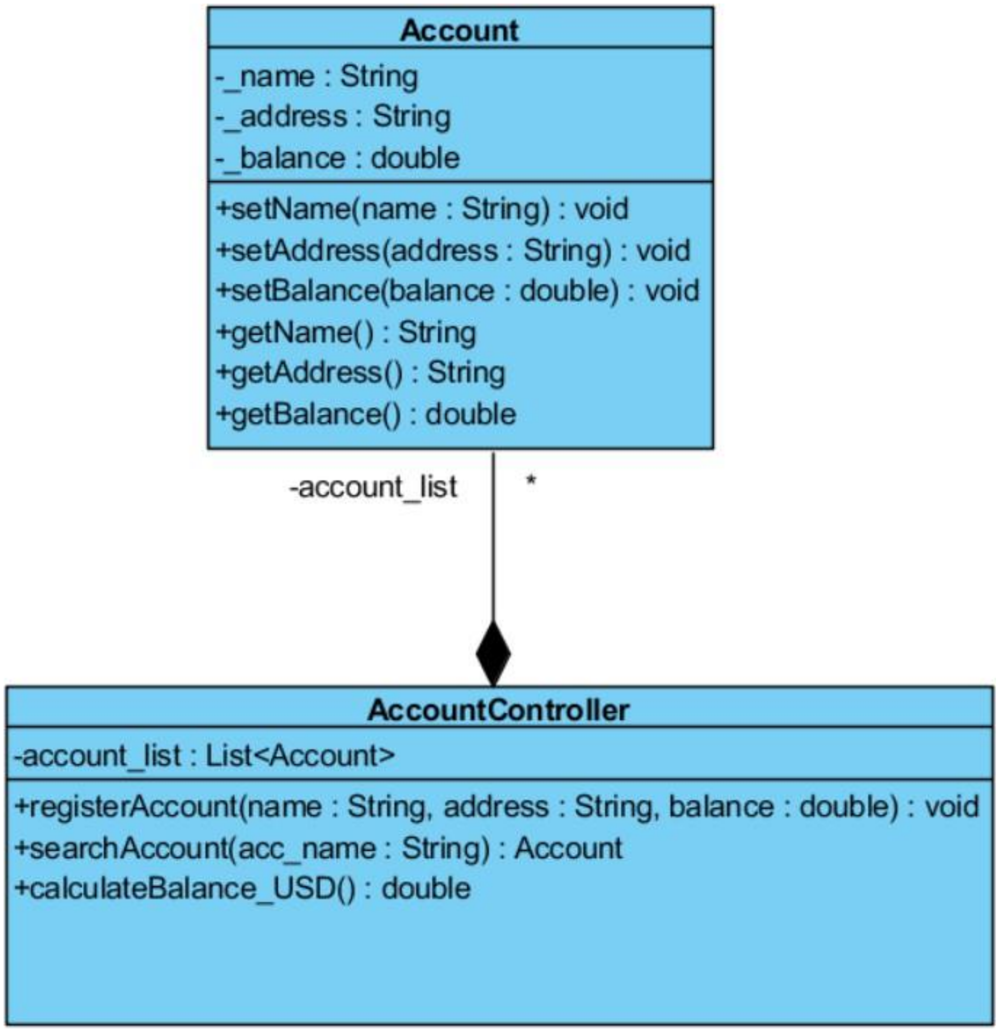
Role	**Description**
Constant/ Fixed value	A variable which is initialized without any calculation and whose value does not change thereafter.
Stepper	A variable stepping through values that can be predicted as soon as the succession starts.
Most-recent holder	A variable holding the latest value encountered in going through a succession of values.
Gatherer	A variable accumulating the effect of individual values in going through a succession of values.
Transformation	A variable that always gets its new value from the same calculation from value(s) of other variable(s).
One-way flag	A two-valued variable that cannot get its initial value once its value has been changed.
Temporary	A variable holding some value for a very short time only.
Organizer	A data structure, which is only used for rearranging its data and object elements after initialization.

Based on the code listings of class **AccountController**, correctly classify the role of each variable in the table below. You may use Table 1 for your reference. (5 Marks)

Variable	**Role of Variable (1 Mark Each)**
i	Stepper
sum_HKD	Gatherer
sum_USD	Transformation
account_list	Organizer
acc_name	Constant

Question 2b – Class Diagram (5 Marks)

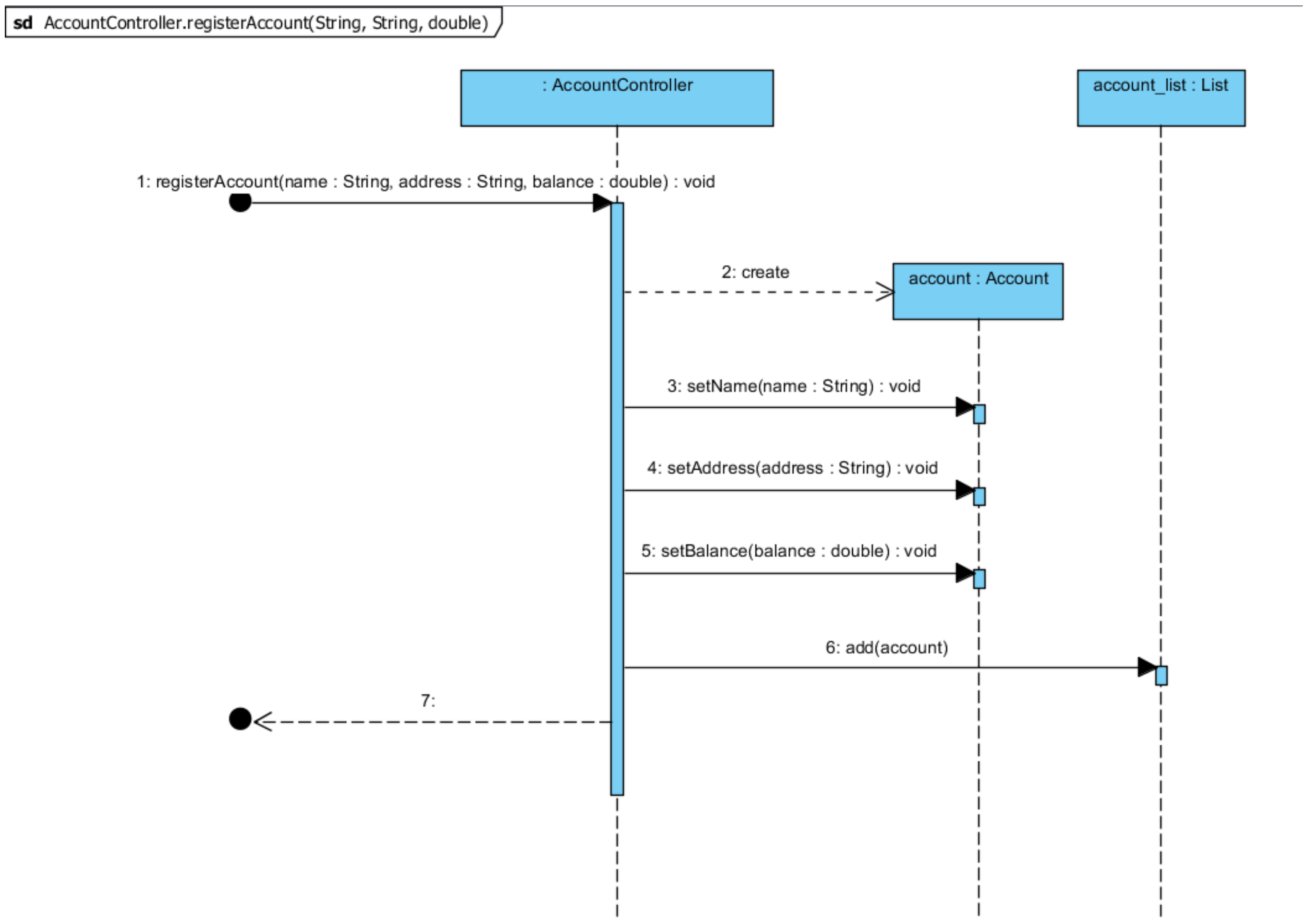
Based on the code listings of class **Account** and class **AccountController**, draw a complete class diagram. Correctly show all the attributes and operations, as well as correctly show their class association using correct UML notation (hint: association/aggregation/composition).



Account
-_name : String
-_address : String
-_balance : double
+setName(name : String) : void
+setAddress(address : String) : void
+setBalance(balance : double) : void
+getName() : String
+getAddress() : String
+getBalance() : double
-account_list
*
AccountController
-account_list : List<Account>
+registerAccount(name : String, address : String, balance : double) : void
+searchAccount(acc_name : String) : Account
+calculateBalance_USD() : double

Question 2c – Sequence Diagram (10 Marks)

Based on the code listings of function **AccountController.registerAccount**, complete the following Sequence Diagram with full details.


sd AccountController.registerAccount(String, String, double)
: AccountController
account_list : List
1: registerAccount(name : String, address : String, balance : double) : void
2: create
account : Account
3: setName(name : String) : void
4: setAddress(address : String) : void
5: setBalance(balance : double) : void
6: add(account)
7:

Question 2d – Design Principles and Patterns (10 Marks)

The current design of the **class Account** is restricted to a single type of Account. The City Bank is planning to differentiate their valued customers into 2 account types, so that customers receive different deposit interest rates according to their deposited balance.

They are:

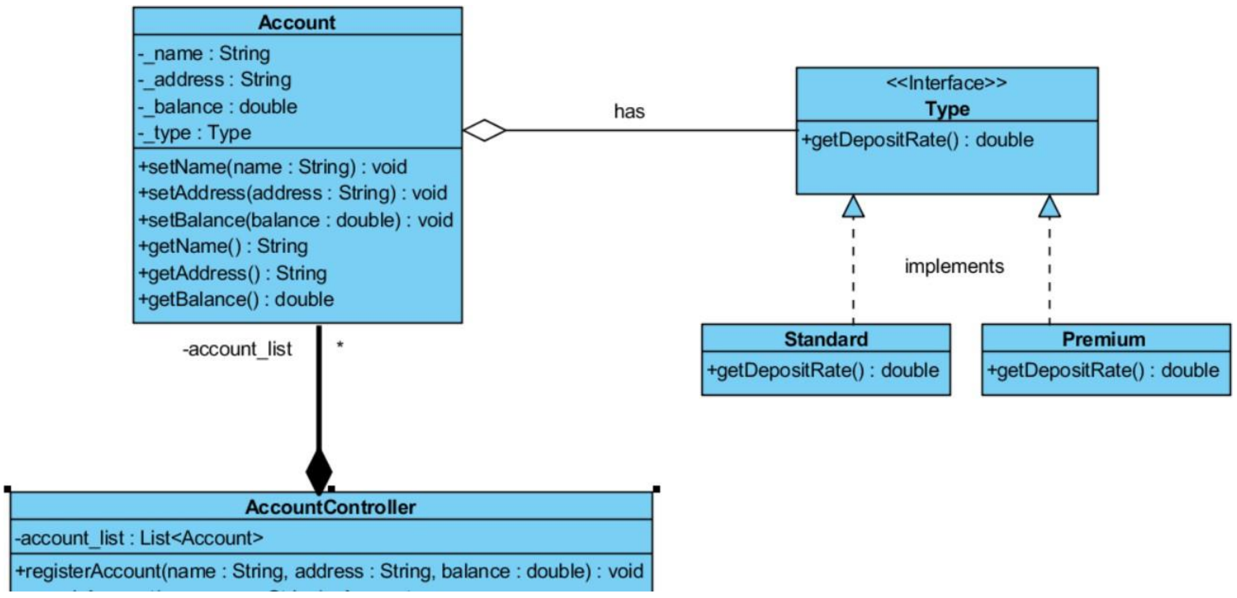
1. Account Type = Standard (If Balance > $0.00, Cash Deposit Rate is 1.0%)
2. Account Type = Premium (If Balance > $10,000, Cash Deposit Rate is 2.0%)

We will need the function **getDepositRate()** accordingly (for either Standard or Premium).

How do you modify the original design to provide different types of Account, so that the bank can easily upgrade a customer from Standard to Premium account and downgrade a customer from Premium to Standard, and to provide different deposit rates accordingly?

Which design pattern do you think it is most suitable? State Pattern (**2** Marks)

Like your solution in 2b, draw a modified/extended class diagram in here. (Showing only affected parts shall be sufficient to illustrate the concept) (**8** Marks)


Account
-_name : String
-_address : String
-_balance : double
-_type : Type
+setName(name : String) : void
+setAddress(address : String) : void
+setBalance(balance : double) : void
+getName() : String
+getAddress() : String
+getBalance() : double
has
<<Interface>>
Type
+getDepositRate() : double
implements
Standard
+getDepositRate() : double
Premium
+getDepositRate() : double
-account_list
*
AccountController
-account_list : List<Account>
+registerAccount(name : String, address : String, balance : double) : void

Question 3 – Software Requirements (20 Marks)

Case Study: ***Sports Facilities Booking System***

The sports center has a various of facilities and needs a Facilities Booking System to ease the booking procedure.

A user can make a booking via the system. First, he needs to check the available time slots of the facilities and select his preferred facility and time slot. Then the system will lead the user to the payment page. The user can pay the booking fee by the credit card. Also, he can choose to delay the payment and complete it at the reception of sports center. Once a new booking is created, the system will update the booking status timely.

The user can view his booking record for details. If he cannot go to the sports center on the booked time, he may cancel the booking or alter it to another available time.

The cleaners are responsible to clean the sports courts after each booked slot. They can view the occupation timetable of all the courts in the system.

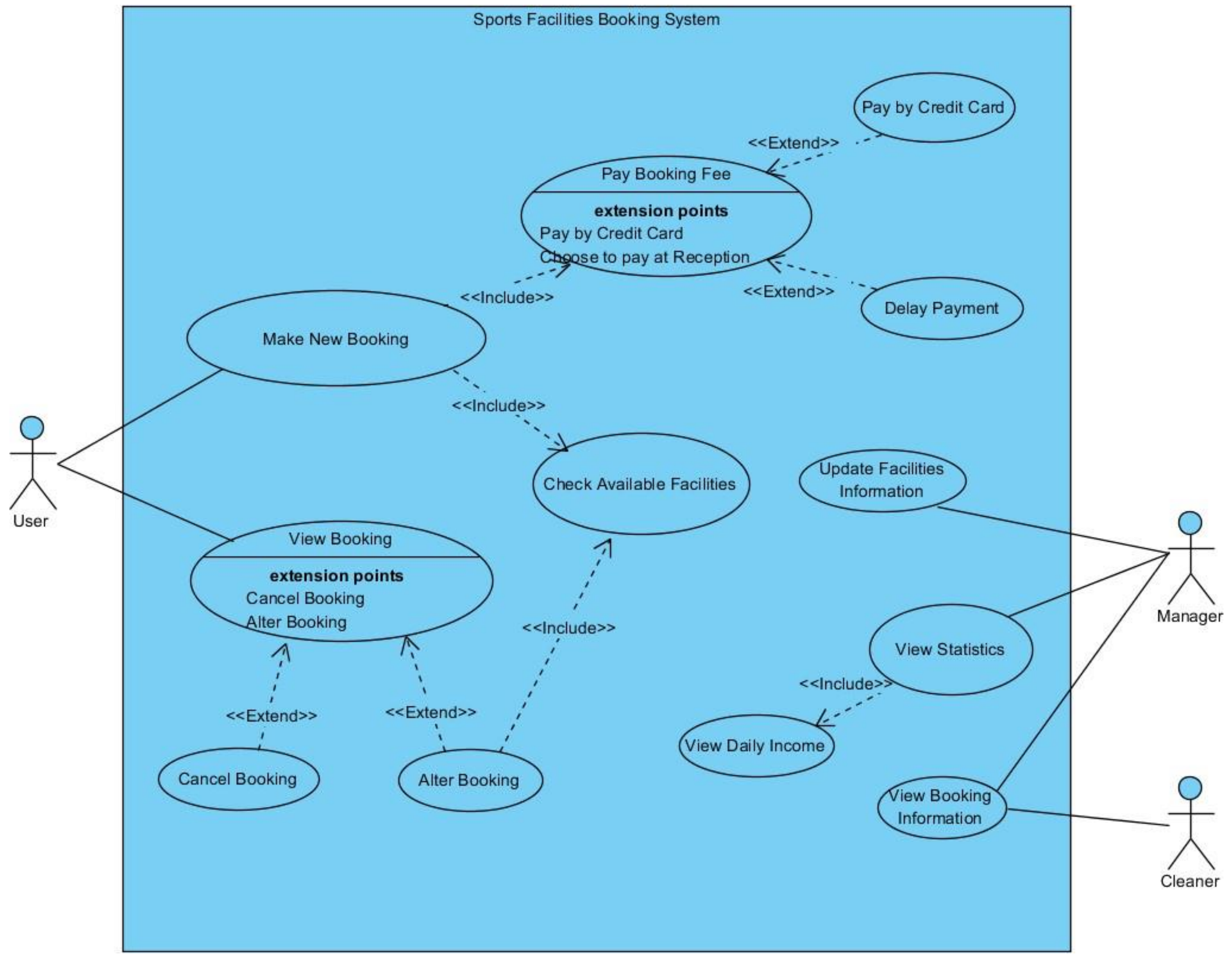
The manager can view the statistics info of all the facilities, including daily incomes. He can also view the occupation timetable of the courts. Besides, some facilities need periodic maintenance. The manager should update the facilities information before and after the maintenance.

Based on the description of the Sports Facilities Booking System above, the following shows the use cases and actors in the system.

Use Case	**Actor(s)**
Make New Booking	**User**
Check Available Facilities	**User (During make or alter booking)**
Pay Booking Fee	**User (During make new booking)**
Pay by Credit Card	**- User (If needed)**
Delay Payment	**- User (If needed)**
View Booking	**User**
Alter Booking	**- User (If needed)**
Cancel Booking	**- User (If needed)**
View Statistics	**Manager**
View Daily Income	**Manager (During view statistics)**
View Occupation Timetable	**Manager + Cleaner**
Update Facilities Information	**Manager**

Based on all the given information above:

Draw a complete **Use Case Diagram** of the Sports Facilities Booking System, you must utilize **<<include>>, <<extend>>** with correct arrows when possible, clearly indicating conditions for extension points if any. **(20 marks)**



-- END --