Software Engineering III

Module Review / Revision

Module Content

I

- Class Diagrams
- Associations
- Sequence Diagrams
- Principle of least Knowledge
- Interfaces / Boundary Classes

II

- Design Patterns

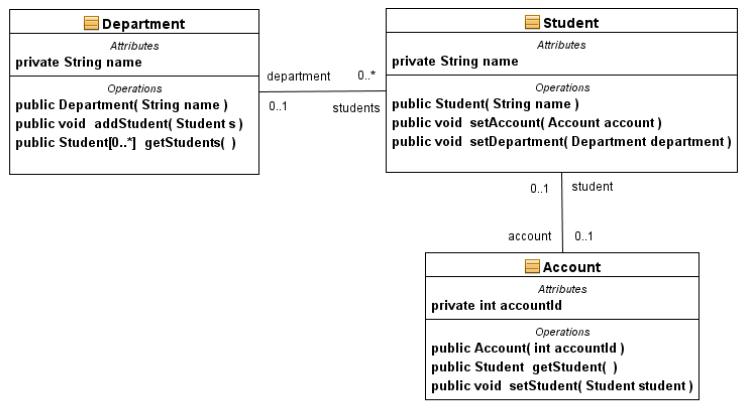
III

- OO Testing
- ORM

Class Diagrams
Associations
Sequence Diagrams
Principle of least Knowledge
Interfaces / Boundary Classes
State Machine Diagrams

Class Diagrams

Associations / Referential Integrity



Sequence Diagrams

```
public class SomeClient {

    SomeClient (Circle circleA,
                Circle circleB)
    {
        circleA.setRadius(10);
        circleB.setRadius(12);
    }
}
```

```
public class Circle {

  private int radius;

  public int getRadius() {
        return radius;
  }

  public void setRadius(int new_radius)
  {
        radius = new_radius;
  }

}
```

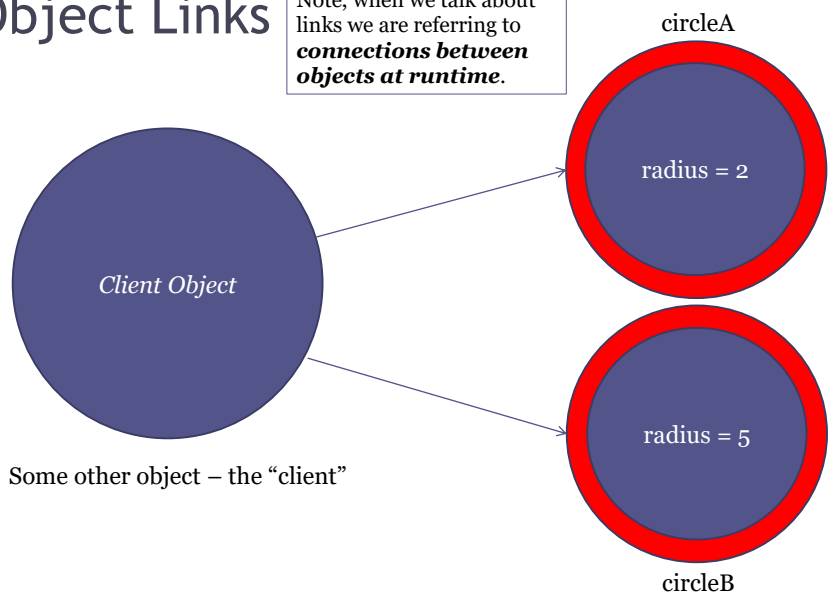
Consider an instance of the above class - it will contain two *references* to two Circle objects. This means that at runtime it will be *linked* to those objects

Vice versa, the two instances of this class (circleA and circleB) are *linked* to the client object

The actual *link* is provided by the references (*circleA* and *circleB*) within the client object
– interestingly, this means that only the client object "knows" about the link. In this sense, the link is *uni-directional*.

Object Links

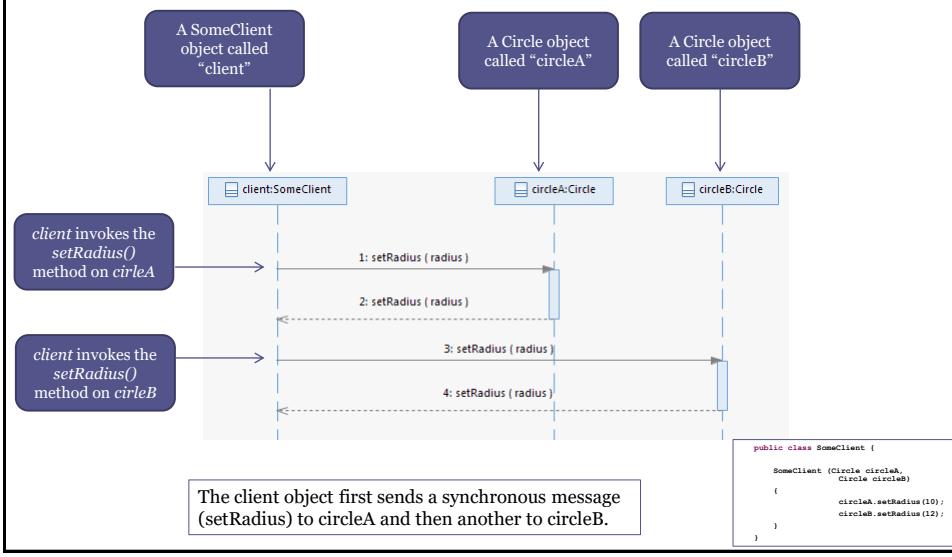
Note, when we talk about links we are referring to ***connections between objects at runtime***.



Some other object – the "client"

Sequence Diagram

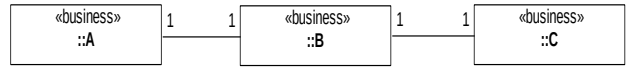
Depicting how the code executes



The client object first sends a synchronous message (setRadius) to circleA and then another to circleB.

Dependencies & Associations & The Principle of Least Knowledge

- It is useful to think of the associations between classes in a class diagram as ***paths of communication*** between objects of the classes.
- In the diagram below, objects of class A may talk to (i.e. send messages to) objects of class B and objects of class B may talk to objects of class A.
- Likewise, objects of classes B and C may also communicate. As there is no direct line of communication between A and C, objects of these classes ***may not communicate directly***. If they do communicate, then there is a bug in the model.



Law of Demeter / Principle of Least Knowledge

- In response to a message M, an object O should send messages only to the following objects:

1. O itself
2. Objects which are sent as arguments to the message M
3. Objects which O creates as part of its reaction to M
4. Objects which are directly accessible from O, that is, using values of attributes of O.

In principle:
- *Each unit should have only limited knowledge about other units: only units "closely" related to the current unit.*
- *Each unit should only talk to its friends; don't talk to strangers.*
- *Only talk to your immediate friends.*

II

Design Patterns

15

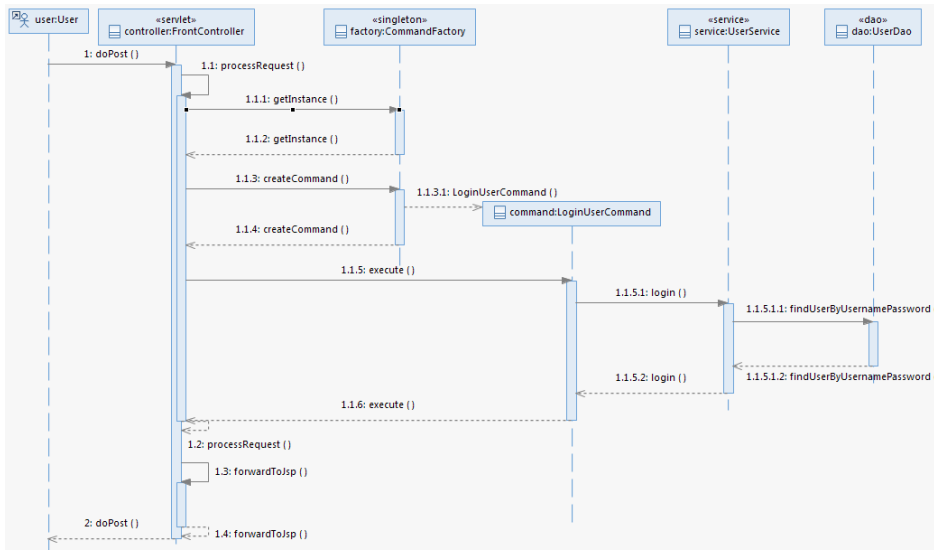
Kinds of Patterns

- ***Creational patterns*** focus on the "creation, composition, and representation of objects, e.g.,
  - **Factory method pattern:** centralize creation of an object of a specific type choosing one of several implementations
- ***Structural patterns*** focus on problems and solutions associated with how classes and objects are organized and integrated to build a larger structure, e.g.,
  - **Adapter pattern:** 'adapts' one interface for a class into one that a client expects
- ***Behavioral patterns*** address problems associated with the assignment of responsibility between objects and the manner in which communication is effected between objects, e.g.,
  - **Command pattern:** Command objects encapsulate an action and its parameters

Patterns

- Singleton
- Command
- Factory
- FrontController
- DAO
- MVC
- Context Object
- REST Architectural Style
- Adapter
- Decorator
- Proxy
- Bridge

Application Example



III

OO Testing
ORM

Exam

Exam

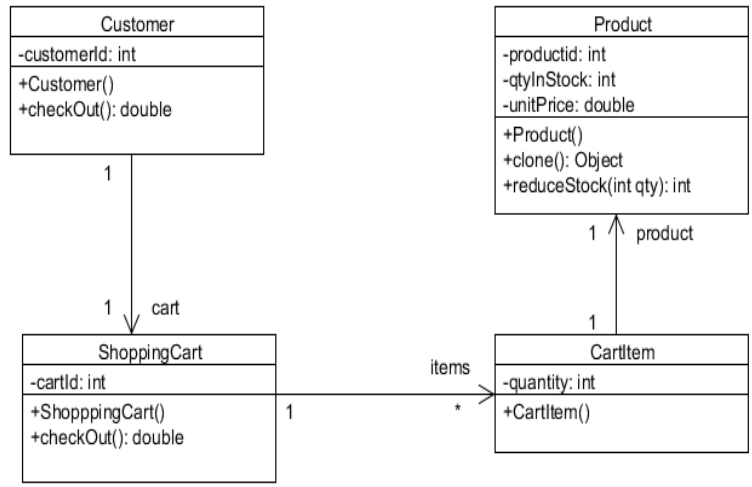
- Answer **3** out of 4 questions
- Each question carries equal marks
- 2 hours for exam
  - 10 minutes to read paper / choose questions
  - 33 minutes per question
  - 10 minutes to re-read your answers

Exam

- Questions are similar in nature to past papers
- There will be 2 questions on *part II* as described earlier
- Study the slides and read the relevant referenced material – chapters etc.
- Go through lab material in conjunction with class slides to help your understanding

Past Paper Questions

Question [17 Marks]



Question - continued

The class specification diagram above shows the associations between four classes. The diagram does not show all methods, but you may assume that all setter / getter methods are implemented for the various attributes in each class. A student's attempt at **implementing the method, *checkout()*, in the *Customer* class and a method, *checkout()*, in the *ShoppingCart* class is shown below** *(next slide)*.

Question - continued

```
public class Customer {

  private int customerId;
  ShoppingCart cart = // etc.

  public double checkOut() {
    double amount = cart.checkOut();
    return amount;
  }
  public static void main(String[] args) {
    Customer customer = new Customer();
    // etc.
    double amount = customer.checkOut();
  }
}
```

```
public class ShoppingCart {

  List<CartItem> items = new // etc.

  public double checkOut() {
    double amount = 0.0;
    for (CartItem item : items) {
      Product p = item.getProduct();
      double price = p.getUnitPrice();
      double qtyInStock = p.getQtyInStock();
      if (item.getQuantity() <= qtyInStock) {
        p.reduceStock(item.getQuantity());
        amount += price * item.getQuantity();
      }
    }
    return amount;
}}
```

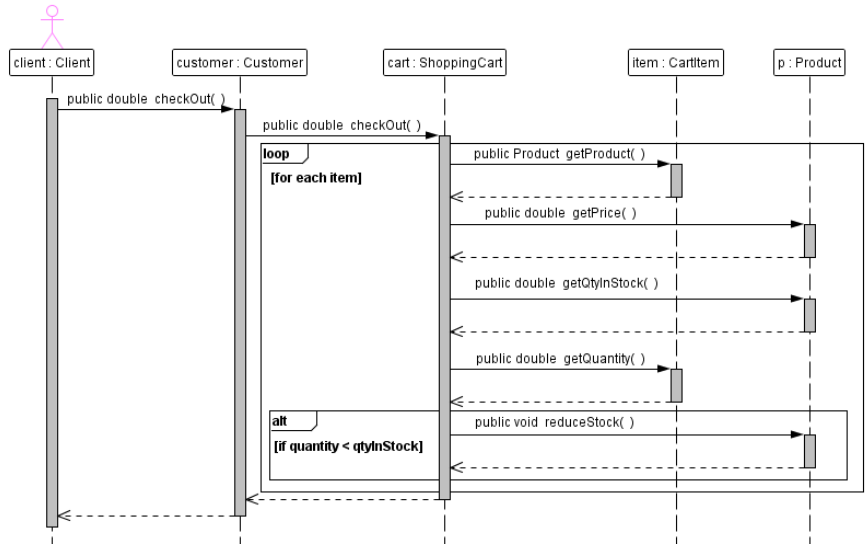
Question - continued

- Draw a sequence diagram to illustrate the sequence of messages starting at

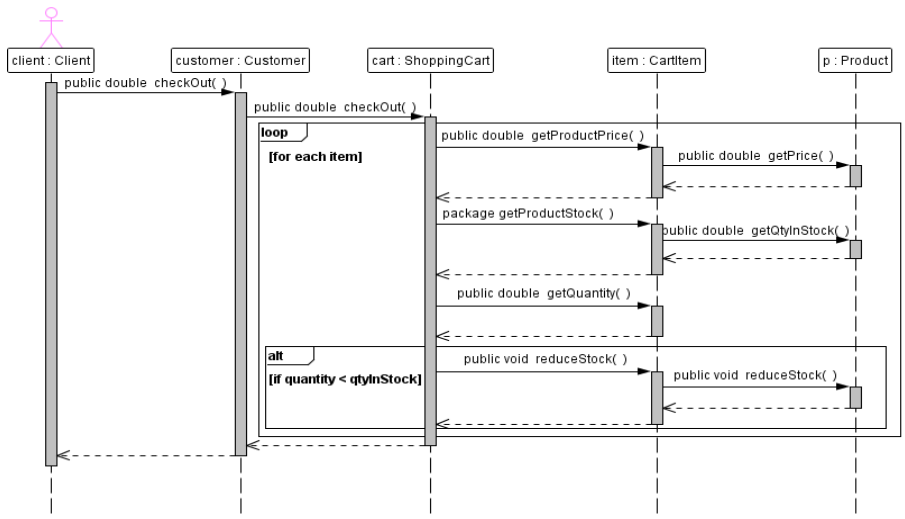
  double amount = customer.checkout() ;

- Without changing the relationships between the classes, draw a new sequence diagram to provide a better implementation. Explain the reasons for your changes.

According to the code



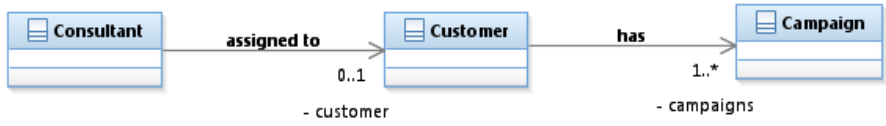
Improved Sequence Diagram



Question (part – 15 marks)

- An advertising agency assigns one of its consultants to support **one** of its customers at any given time. A consultant may occasionally be not assigned to any customer. A customer will have one or more active campaigns and the consultant will assist in the progress of each of these campaigns. Assume that all queries and processing will be met by navigating from consultant to customer and from customer to campaign.
- Draw a class diagram to model these requirements showing clearly the attributes and additional class(es) required to model the associations. Include the navigational and multiplicity notation for each association in the diagram.

Solution



Question

- The *Data Access Object* and *Transfer Object* are two patterns that can be used together in the design of a *Persistence Layer*.
  - Outline three benefits of using the above patterns in the design and implementation of a software system.

    [6 marks]
  - Using a language of your choice, describe how you could implement the patterns in code.

    [10 marks]

Outline Solution

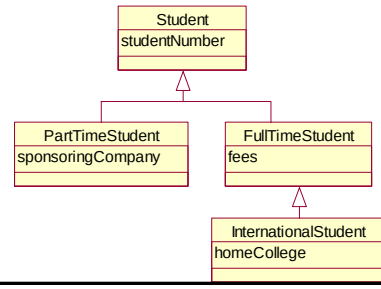
- **2 x 3 marks each for brief description– e.g.**
  - Encapsulation
  - Separation of concerns
  - Refactoring
  - Coding to interfaces (Abstraction / Indirection)
- Examples of:
  - DAO Interface with operation specifications for various CRUD functionality for example.
  - DAO Class Implementing the interface. Possible use of JDBC or ORM code.
  - A DTO Class encapsulating data attributes of the entity with getter/setter operations

Question

- When mapping a class diagram which contains inheritance from hierarchical levels of classes, a simple mapping approach is to map the entire class diagram tree onto one database table.
  - Using the class diagram below, derive appropriate columns for the table.

    [6 marks]
  - State the advantages and disadvantages of this approach and when it is suitable to adopt as a mapping strategy

    [8 marks]



Outline Solution

StudentNumber <<PK>>
StudentType
SponsoringCompany
Fees
HomeCollege

Advantages:

- Simple approach.
- Easy to add new classes, you just need to add new columns for the additional data.
- Supports polymorphism by simply changing the type of the row.
- Data access is fast because the data is in one table.
- Ad-hoc reporting is very easy because all of the data is found in one table.

Disadvantages:

- Coupling within the class hierarchy is increased because all classes are directly coupled to the same table. A change in one class can affect the table which can then affect the other classes in the hierarchy.
- Space potentially wasted in the database.
- Indicating the type becomes complex when significant overlap between types exists.
- Table can grow quickly for large hierarchies.

When suitable for adoption:

- This is a good strategy for simple and/or shallow class hierarchies where there is little or no overlap between the types within the hierarchy.