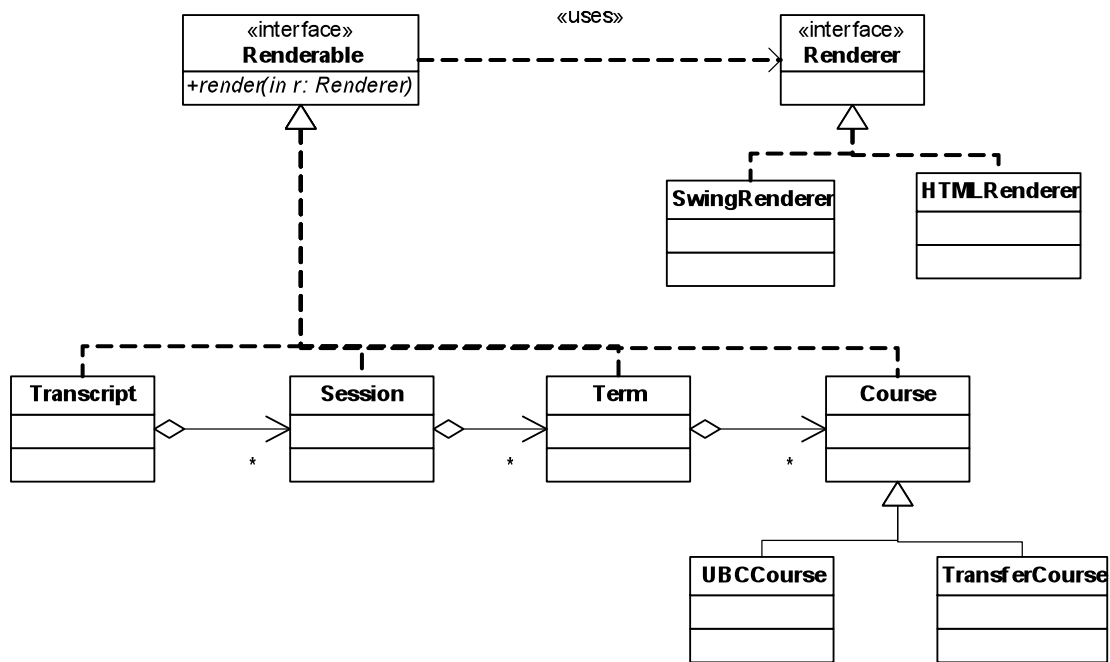


SOLUTIONS to MIDTERM PRACTICE EXERCISES

Software Design

1. Although there may be other ways to design this problem, the following design is probably the simplest one.



2.
a)

	precondition	postcondition
addItem	<i>stronger</i>	<i>same</i>
computeBill	<i>same</i>	<i>stronger</i>
checkout	<i>stronger</i>	<i>stronger</i>

b) DeliveredGroceryOrder is not a proper subtype of GroceryOrder according to the LSP because the preconditions on addItem and checkout are stronger. For the LSP to hold, preconditions in the subclass must be weaker (or the same) and postconditions must be stronger (or the same).

Exceptions

1. Output is:

```
catchIt caught: wow an exception
catchIt caught: wow amazing exceptional!
In throwIt a is: 10
```

Software Testing

1.

- a. amount > 0
amount <= 0

b. Some sample test cases (others were possible):

- 1) theRow=25, theSeat = 50; typical
- 2) theRow=1, theSeat = 1; boundary
- 3) theRow=50, theSeat = 10; boundary
- 4) theRow=49, theSeat = 1; boundary

Java Collections etc.

1.

```
public static <E> void deleteAll(List<E> list, E obj ){
    Iterator<E> itr = list.iterator();

    while ( itr.hasNext() ) {
        if ( obj.equals( itr.next() ) )
            itr.remove();
    }
}
```

This operation take $O(n)$ time for LinkedList and $O(n^2)$ for an ArrayList. What if we don't use iterator and write:

```
public static <E> void deleteAll(List<E> list, E obj ){

    for ( E cur : list ) {
        if ( obj.equals( cur ) )
            list.remove(cur);
    }
}
```

Is this a correct code?

2.

```
public static <E> List<Integer> getIndices(List<E> lst, E obj) {
    List<Integer> result = new ArrayList<Integer>();

    ListIterator<E> itr = lst.listIterator();

    while ( itr.hasNext() ) {
        if ( obj.equals( itr.next() ) )
            result.add(itr.previousIndex());
    }
    return result;
}
```

This operation take $O(n)$ time.

3.

```
public static <E> List<E> subst(List<E> list,E old, E new) {

    List<E> newList = new ArrayList<E>();

    Iterator<E> itr = list.iterator();
    while ( itr.hasNext() ) {
        E nextElt = itr.next();
        if (nextElt.equals(old))
            newList.add(new);
        else
            newList.add( nextElt );
    }
    return newList;
}
```

4.

```
public void addItem(GroceryItem newItem) {

    Integer count = itemCount.get(newItem);
    if(count == null) {
        itemCount.put(newItem,1);
    }
    else
        itemCount.put(newItem, count++);

    totalItems++;

}
```

```
5. public boolean equals( Object o ) {  
    if ( o == null )  
        return false;  
    if ( getClass() != o.getClass() )  
        return false;  
  
    Dog d = (Dog) o ;  
    return breed.equals(d.breed) && name.equals(d.name)  
        && gender.equals(d.gender);  
}
```