Objectives:
To discuss the solution of the midterm and answer any doubts about grading issues. Also, during the discussion, refresh the topics from the first part of the course.

Reading Assignment:
Textbook: pages thru 306, 320-328 (for next week).

Contents:
<table>
<thead>
<tr>
<th>Activity</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discuss the answers of Midterm</td>
<td>2 hr</td>
</tr>
<tr>
<td>Return the graded papers and answer questions</td>
<td>1 hr</td>
</tr>
</tbody>
</table>
Discuss solution to Midterm 100 min / 100 min
Return Graded Midterms and discuss individual problems 50 min / 150 min
CS445 Midterm Exam Solution

**Question 1 (10 points)**

**Differentiate between weak and strong composition**

**Give examples**

Looking for an answer in which students should give the definition of composition and explain the issue of ownership or containment. With respect to that, then give the difference between lifetime based on the ownership/containment issue. Thus defining and differentiating weak and strong composition.

Not looking for code examples here, though code examples are also ok if they are explained in a proper way, i.e. plain English - stating that if the container is destroyed so will the contained in the case of strong and not in the case of weak. The answer would really be a good one if students mention the relationship in which containee gets destroyed and container can or cannot be destroyed (missile of a fighter example) that was beaten to death in class.

**Question 2 (10 points)**

**Discuss difference between an Abstract class and an Interface**

Pretty much the true java question of the exam. Looking for an answer in which there is both definitions of Abstract class and Interface. Students should have noted that none of the classes can be instantiated. However variables of abstract classes can be declared and points would not be docked for that. This question is aiming at java but students explaining on basis of c++ (i.e. interface as a header file…etc.) shall be considered for partial credit based on the answer

**Question 3 (5 points)**

**What is dynamic binding?**

If you answer this question and understand what you are saying then the next two questions are automatic. The reason is based on the understanding of this answer, your next two would depend upon. Though you can get one or the other correct, but its really the understanding that at run-time the type of the method invocation can be determined. You need to have a good understanding of polymorphism and how its implemented in memory.
Question 4 (15 points)

There are 3 classes: Circle, Square and Triangle. Each class has the data element that contains the name of the shape (Circle, Square, Triangle). The only condition that we have for the name is that it should not be null. Based on this information, write a skeleton structure of the classes (at least 1 constructor and accessor methods needed. Define the accessor methods based on the information given).

Really looking for the use of inheritance here. If the student has answered using delegation or aggregation, points could be severely deducted as this is a straightforward question of usage of commonality or generalization. However, partial credit might be given based on the performance.

Though good code is expected, points shall not be cut for minor syntax errors.

```java
public class Shape {
    private String name;

    public Shape ( String nameIn )
    {
        setName (nameIn);
    }

    public String getName ()
    {
        return (name);
    }

    public void setName (String nameIn)
    {
        // some sort of error checking to see if its not null is expected
        // points shall be deducted for not having that
        name = nameIn;
    }
}
```

```java
public class Circle extends Shape {
    public Circle (String nameIn)
    {
        super (nameIn);
    }
}
```

```java
public class Square extends Shape {
    public Square (String nameIn)
    {
        super (nameIn);
    }
}
```

```java
public class Triangle extends Shape {
    public Triangle (String nameIn)
    {
        super (nameIn);
    }
}
```

Mentioning the accessors in the child classes is not necessary. If overdone, points shall not be deducted unless major mistakes occur. Another point about not overdoing.
Question 5 (20 points)

Consider the following code

```java
public interface Flyable {
    ...
    public void takeoff();
    ...
} // end of interface Flyable

class F16 extends Fighter implements Flyable {
    ...
    public void takeoff()
    {
        ...
    }
} // end of class F16

class F22 extends Fighter implements Flyable {
    ...
    public void takeoff()
    {
        ...
    }
} // end of class F22
```

The implementation of the code is in Java. Consider the scenario in which F16 and F22 are fighters on a warship. Now the ship has multiple number of both the Fighters. The number of that aircraft can vary depending upon the other conditions based upon the extra helicopters that its carrying for the mission.

Now the Ship is under enemy fire and has to get all its fighters off the board. It can do so by invoking the takeoff method of the Fighters.

```java
if (underFire)
{
    ...
    // your code
}
```

Pretty much the easiest of all the questions if you know what you are doing.

The only thing required here is going through a vector of ‘Fighters’ and then just using the object returned to call the ‘takeoff’ method on it. Again coming back to dynamic binding. Points would be deducted here for overdoing work and trying to find the type of the aircraft since its not necessary and it would add additional overhead.