Homework 4
Kevin Bumgarner
Comp 3700 - 001

Question 1:

Visibility is the design concept described as the ability of one class to see or reference an object of another class.

Visibility types and Examples:

1. **Attribute Visibility**: A = Animal, B = Dog
```
Class Animal {
    private Dog d = new Dog();
}
```
2. **Local Visisibility**: A = paymentSystem, B = Item
```
Class paymentSystem {
    Item i = new Item();
}
```
3. **Global Visibility**: Dog d = new Dog(); **This variable is global
4. **Parameter Visibility**: A = paymentSystem, B = Item
```
paymentSystem.buyItem(new Item, amount);
```

Question 2:

a. A peer-to-peer system is a decentralized system containing many machines on the same network. This allows computations to be carried out by any machine on the network.
b. An example of a peer-to-peer system includes decentralized networks like the Ethereum network or more popular bitcoin.

Question 3:

My proposed Architecture for the Holiday Resort Software System is **Client-Server Architecture.** I would use this because Clients would be accessing the software features through servers across a wide area such as different states in the US. The servers being accessed would offer different services to the clients and distribute the workload across multiple servers.

Question 4:

System Interface = public class PointOfSaleSystem {
```
        public makeNewSale();
        public enterItem(itemID, quantity);
        public endSale();
        makePayment();
}
```

Question 5:

Knowing Responsibilities:

1. **knowing some data (attributes)**: The Dog class knows about private attributes like dog height
```
Class Dog {
   private height;
}
```
2. **knowing about related objects (using links)**: The Dog Class Extends Animal Class
```
Class Dog Exends <Animal> (){}
```
3. **knowing about things it can derive (derived attributes)**: The Dog Class can calculate the height of a Dog
```
Dog.getHeight(maxTheDog);
```

Doing Responsibilities:

1. **doing something itself**: The Dog class can create dog objects
```
Dog maxTheDog = new Dog();
```
2. **initiating action in other objects**: The Animal class can create new dogs
```
Animal.createDogs(maxTheDog);
```
3. **controlling and coordinating activities in other objects**: The Animal class can give a dog a home
```
Animal.giveAnimalHome(maxTheDog);
```

Question 6:

'Cup' Should follow an **Information Expert** GRASP design principle as long as it has information about the two die and the functionality to randomly generate a sum of two die. The Cup class will have the necessary information to fulfill the responsibility of returning the sum of two die and that is its primary purpose, nothing more.