1)Classes that are used in the class hierarchy to define base class and which give the abstraction ability. We can't create objects from this class. We can't write static methods in the abstract classes. We usually write abstract methods. But it can contain non-static and non-abstract method. I defined the Beverage class as abstract. I think tea beverage and caffein beverage are can be abstract classes in different implementation.

```
public abstract class Beverage
```

2)They are methods with no body. We can only write in the abstract classes. They must be overridden by inheritance classes. They cannot be defined as private. I defined cost and withMilkOrWithLemon method as abstract.

```
public abstract int cost();
public abstract String withMilkorWithLemon();
```

3)Methods must have body in the class. Interfaces contain just the sign of the methods.
We use extends keyword to inherit from a class. We use implements keyword to inherit from an interface. We can not multiple inheritance from classes. We can multiple inheritances from interfaces. We can create an object from classes. We can not create an object an interfaces. Interfaces don't have constructers. If I did not write the sizeName method in the beverage class the beverage class would be an interface. Because interface does not contain methods body.
4)Static variables are independent of objects created from the class. We can use those variables without creating an object. Normally, when producing objects from a class, we create a new copy for each object we produce. But copies of a variable with the keyword static are not produced. All examples in the class share this static variable in a common way. Non-static variables are object-specific. Small Medium Large are my static variables.

```
public static final int SMALL =0;
public static final int MEDIUM =1;
public static final int LARGE =2;
```

5)All of my cost methods are an example to polymorphism. They are returning different value for each object. We also use polymorphism in the TestCafe class.

```
CaffeineBeverage cBeverage = new Latte( hasMilk: true, Beverage.SMALL);// We create a latte object from CaffeinBeverage Type.
order.add(new OrderItem(cBeverage,  amount: 2));//Order item constructor takes Beverage Type parameter but we pass CaffeinBeverage Type.
TeaBeverage tBeverage = new BlackTea( hasLemon: false, Beverage.SMALL);// We create a BlackTea object from TeaBeverage type.
order.add(new OrderItem(tBeverage,  amount: 3));//Order item constructor takes Beverage Type parameter but we pass TeaBeverage type.
```