Abstraction

Abstraction is to hide the detailed implementation from the user. In other words, it is a concept of only providing essential data/functions to the user, and users would not be able to see or to use the functions that they don't need.

Take "interface" in Java as an example. Interface class is a way to achieve Abstraction in Java. A interface class looks like below, which has the names of a interface methods that does not have a body.

```
interface StudentOps {
   void insert(Student);
   void delete(Student);
   void update(Student);
}
```

Then it would be implemented by a class, which will cover the all the implementation of interface methods.

```
class JdbcStudentOpsImpl implements StudentOps {
   public void insert(Student s) {
     // code
         validate()
   }
   public void delete(Student s) {
     // code
         validate()
   }
   public void update(Student s) {
     // code
         validate()
   }

   public void validate(Student s) {
     // code
   }
}
```

When we create an instance of interface by executing the constructor of a concrete class that implements the interface, we would not be able to access the "validate" method, because it is hidden from the user.

```
StudentOps studentOps = new JdbcStudentOpsImpl

studentOps.insert(std)
studentOps.delete(std)
studentOps.update(std)
```

Abstraction prevents the programmer wrongly called the undesired function which should only be used in implemented class. And because the implementation method is written in implemented class, we can create and use another implementation class without changing the function we called if we want to use another method to achieve the features. For example, if we want to change the DB connection from jdbc to odbc connection, we could simply create another odbc implementation class and change the object we initiated without change the function we called.

```
StudentOps studentOps = new odbcStudentOpsImpl

studentOps.insert(std)
studentOps.delete(std)
studentOps.update(std)
```

https://stackoverflow.com/questions/62112528/how-does-interface-helps-in-hiding-details -The example java code

Encapsulation
“Encapsulation is defined as the wrapping up of data under a single unit.” The data/function should be only used or seen within a class if it is unnecessary for other part of code.

In Java, it could be achieved by “private” declaration. By declaring private before a variable or a function, the variable and function can only be access within the class or function and cannot be called outside of the class or function.

```
Class Birthday{
        Private String date;

        Public void setDate(int month, int day){
                date=Integer.toString(month)+”*”+Integer.toString(day)
        }
}
```

In the case above, the date can only be changed by called by the setDate function and cannot be directly changed through called the variable from the object. That’s say the date variable in Birthday object should strictly follow the format “month*day“, by using private variable, it prevent the user change the variable and not followed the format.