E

№ 0570809

國立雲林科技大學考試答案卷

1.+2

第 1 頁

學年		學期		日期		考別	□平時考 □期中考 □學期考
科目						評分	70
系所	資管系	年級	Ⅲ	學號	B10323024	姓名	賴宥穎 Jim

一.

+7

1. Decorator
2. Strategy
3. Mediator
4. Iterator
5. Bridge
6. Visitor
7. Builder
8. Flyweight
9. Mementor
10. Observer

+6

二. Similarities +2

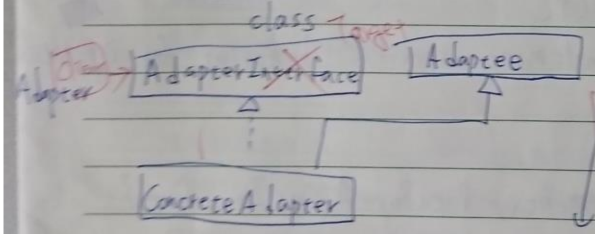
Adapter use Adaptee, and Subject Proxy use Proxy. The use way is same. Use them to do other thing

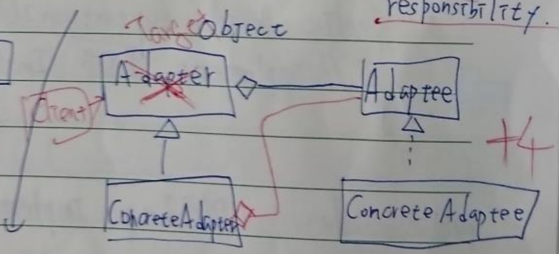
provide one interface for client

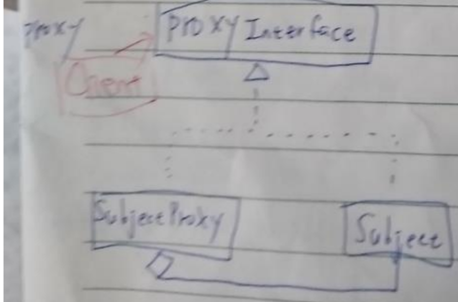
Different

Adapter used after implement. We can use Adaptee to change method detail in Adapter

Proxy help Subject to protect. remote. smart. etc..., Let subject to focus its responsibility.







Protect : we can limit user

remote : we can remote the subject

smart : we can detect subject resources.

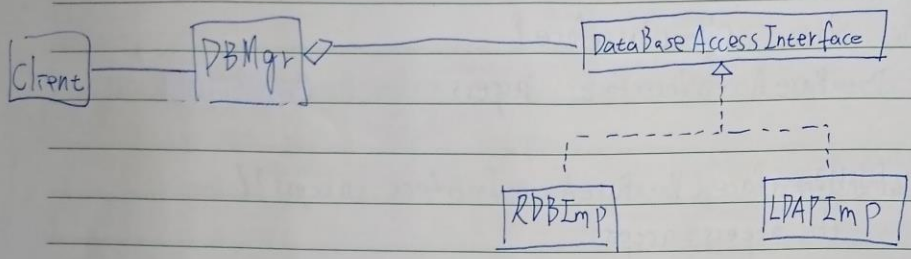
1

+18

三、

```
public interface DataBaseAccessInterface{
        abstract public void operation1();
        abstract public void operation2();        +13
}
public class RDBImp implements DataBaseAccessInterface{
                                ;
       RDBImp(        ){

       }

      public void operation1(){
          // implement RDB Implement specific behavior for operation1
      }
     public void operation2(){
        // implemene RDB Implement specific behavior for Operation2.
     }
}
public class LDAPImp implements DataBaseAccessInterface{

     LDAPImp(){

     }
     public void operation1(){
        //implement LDAP Implement specific behavior for operation1.
     }
    public void operation2(){
       // implement LDAP Implement specific behavior for operation2.
    }
}
```

```
public class DBMgr {
    DataBaseAccessInterface Imp.

    DBMgr (DataBaseAccessInterface Imp ){
        this.Imp = Imp.;

    }
    public void operation1() {
        Imp.operation1();
    }
    public void operation2() {
        Imp.operation2();
    }
    /* other operation including operations to change the Imp to
       refer to a different concrete implementation */
    public void setAccess (DataBaseAccessInterface Imp){
        this.Imp = Imp;
    }
}
public class Client{
    public void use (){
        RDBImp rdbImp=new RDBImp ();
        DBMgr mgr = new DBMgr (rdbImp);
        mgr.operation1();    // RDB operation 1
        mgr.operation2();    // RDB operation 2
        LDAPImp ldapImp= new LDAPImp();
        mgr.setAccess(ldapImp);
        mgr.operation1();    //LDAP operation1;
        mgr.operation2();    //LDAP operation2;
    }
}
```



Client can selet a different concrete Implementation
Client only know : use DBMgr, and it doesn't know how to work detail. The concrete Implementation are RDBImp and LDAPImp.

+5

We can assign RDBImp or LDAPImp to DBMgr constructor when DBMgr is created.
Even we add setAccess in DBMgr.
We can use this method to indicate RDBImp or LDAPImp, so we can use different database access.

№ 0570808

國立雲林科技大學考試答案卷　　　　第 1 頁

學　年		學　期		日　期		考　別	□平時考　□期中考　□學期考
科　目						評　分	
系　所	資管系	年　級	四	學　號	B10323024	姓　名	賴宥穎 Jim

四　　+7

+19 1. connectDB() and disconnectDB is concrete operation
Becuase GetDiagram and SaveDiagram is same way to connect and disconnect, so we use concret method in parent, they can shareable use.

queryDB is primitive method.　　+12
The reason is, it's a abstract method,
GetDiagram and SaveDiagram need to overriding queryDB, the implemention is different.
And queryDB has to be used in template method.
Because we use queryDB to read, update etc....
So its primitive.

processResult is factory method and hook method.
factory method reason: Because we use processResult to create state diagram. And we use high level class to call lower level class, it follow "Don't call us, we call you"
Lower level class can different implement, let parent class use.

hook method reason: we can selet this action need to do. GetDiagram use processResult to create a stat diagram, but SaveDiagram use processResult to do nothing so, it's a hook method

5

```
public class DBMgr{
    DBImplInterface imp;
    DBMgr(DBImplInterface imp){
        this.imp=imp;
    }
    StateDiagram getDiagram(String name){
        return imp.getDiagram(name);
    }
    void saveDiagram (StateDiagram d){
            imp.saveDiagram(d);
    }
}
public interface DBImplInterface{
        abstract StateDiagram getDiagram(String name);
        abstract void saveDiagram(StateDiagram d);
}
public class RDBImp implements DBImplInterface{

        RDBImp() {

        }


    StateDiagram getDiagram(String name){

            RDBImplCmd sd = new GetDiagram("RDB", name);
            sd.execute();
            return sd.getResult();
    }
    void saveDiagram(StateDiagram d){
        RDBImpCmd sd = new SaveDiagram("RDB", d);
        sd.execute();
    }
}
public class LPAPImp implements DBImplInterface{

    LDAPImp(){

    }

    StateDiagram getDiagram(String name){
            RDBImpCmd sd = new GetDiagram("LDAP", name);
            sd.execute();
           return sd.getResult();
    }
    void saveDiagram(StateDiagram d){
          RDBImCmd sd = new SaveDiagram("LDAP", d);
          sd.execute();
    }

}
```

```
public abstract class RDBImpCmd {
        String type; // [redacted] RDB or LDAB
        Object result;
        RDBImpCmd(String type){
                this.type = type;

        }
        final void execute(){
                try {
                        connectDB();
                        queryDB();
                        disconnectDB();
                        processResult();
                } catch (SQLException e){
                        disconnectDB();
                }
        }


        void connectDB(){
           //According type to connect db.
        }
        void disconnect(){
           // Disconnect db
        }
        Object getResult(){
              return result;
        }
        abstract void queryDB();
8 }     ~~abstract void processResult();~~
```

№ 0570885

第 1 頁

國立雲林科技大學考試答案卷

學 年		學 期		日 期		考 別	□平時考 □期中考 □學期考
科 目						評 分	
系 所	資管系	年 級	四	學 號	B10323024	姓 名	賴宥翔 Jim

cont 四

```
public class GetDiagram extends RDBImpCmd {
        String name;
        GetDiagram(String type, String name){
            super(type);
            this.name=name;
        }
        String data; // save queryDB result
        void queryDB(){
          // According type to query the different database to
          // retrieve the diagram data
          // save into data variable.
       }

       void processResult(){
          // According data to create a state diagram and
          // populate it with query result.
          // save into result variable.
       }
}
```

```
public class SaveDiagram extends RDBImplCmd{
    StateDiagrom d;

    SaveDiagram (String type, StateDiagram d){
            super(type);
            this.d = d;
    }

    void queryDB(){
     //According type to save diagram to different database.
    }
    void processResult(){
         // do nothing
    }
}

public class EditController{
    public void use(){
        RDBImp rdbImp = new RDBImp();
        DBMgr mgr= new DBMgr(rdbImp);
        StateDiagram d= mgr.getDiagram("stateDiagram");
        mgr.saveDiagram(d);
          // ↑ use RDB database to work
          // ↓ use LDAP database to work
        LDAPImp ldapImp = new LDAPImp();
        mgr= new DBMgr(ldapImp);
        StateDiagram d2= mgr.getDiagram("stateDiagram");
        mgr.saveDiagram(d2);
    }
}
```

國立雲林科技大學考試答案卷 № 0648657 第 1 頁

學 年		學 期		日 期		考 別	□平時考 □期中考 □學期考
科 目						評 分	
系 所	貿管系	年 級	四	學 號	B10323024	姓 名	賴宥穎

+20

五

```
public abstract class IteratorFactory{
    public Iterator algorithm( Component c ){
      return factoryMethod (list);
    }
    abstract Iterator factoryMethod( Component c' );
}
```

+15

```
public class LevelOrderIteratorFactory extends IteratorFactory{
    Iterator factoryMethod( Component c ){
        return new LevelOrderIterator( c );
    }
}
public class InOrderIteratorFactory extends IteratorFactory{
    Iterator factoryMethod( Component c ){
        return new InOrderIterator( c );
    }
}
public class PostOrderIteratorFactory extends IteratorFactory{
    Iterator factoryMethod( Component c ){
        return new PostOrderIterator( c);
    }
}
```

11

```
public class PreOrderIteratorFactory extends IteratorFactory {
    Iterator factoryMethod ( Component c ){
        return new PreOrderIterator( c );
    }
}
```

+5

```
public class Iterator {
    ArrayList<Object> result ;// save to sort result
    int pos=0;
    Object next (){
        if ( result.size >pos) return result.get (pos++);
        return null
    }
    boolean hasNext (){
        if (result.size >pos) return true;
        return false;
    }
    void remove (){
        int i =0;
        for (   i= pos-1 ; i < list.size -2 ; i++ )
            list.get (i) = list.get ( i+1);
        list.get (i ) = null;
    }

    Iterator (){}
}
```

```
public class InOrderIterator extend Iterator{
        InOrderIterator(Componene c){
                // Use C to in order
                // And save into result
        }
}
public class LevelOrderIterator extends Iterator{
        LevelOrderIterator(Component c){
                // Use c to level order
                // And save into resule
        }
}

public class PostOrderIterator extends Iterator{
        PostOrderIterator(Componene c){
                // Use c to post order
                // And save into resule
        }
}
public class PreOrderIterator extends Iterator{
        PreOrder Iterator(Component c){
                // Use c to pre order
                // And save into resulte
        }
}
```