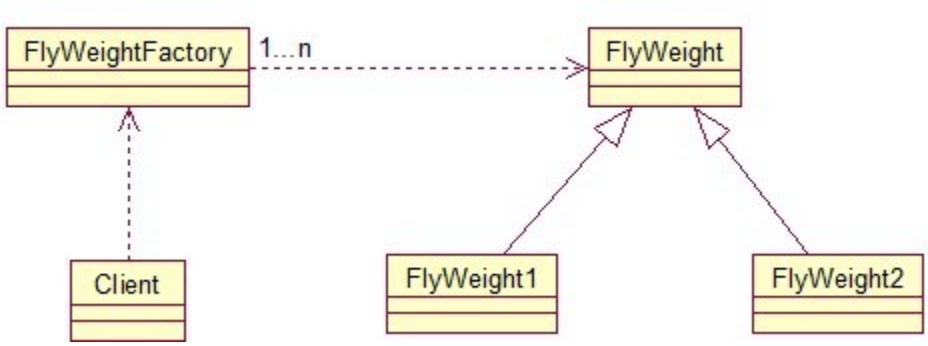
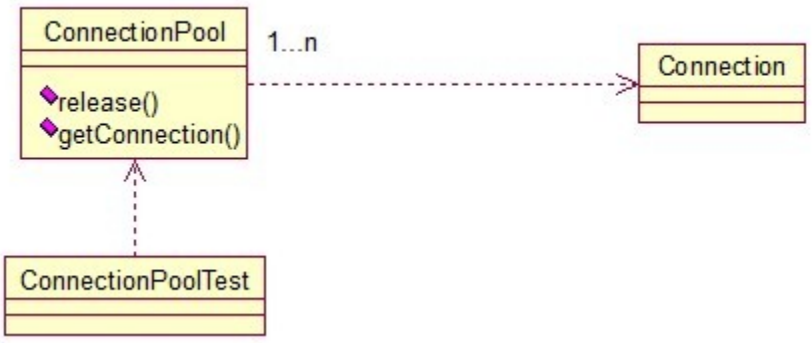
享元模式的主要目的是实现对象的共享，即共享池，当系统中对象多的时候可以减少内存的开销，通常与工厂模式一起使用。



FlyWeightFactory负责创建和管理享元单元，当一个客户端请求时，工厂需要检查当前对象池中是否有符合条件的对象，如果有，就返回已经存在的对象，如果没有，则创建一个新对象，FlyWeight是超类。一提到共享池，我们很容易联想到Java里面的JDBC连接池，想想每个连接的特点，我们不难总结出：适用于作共享的一些个对象，他们有一些共有的属性，就拿数据库连接池来说，url、driverClassName、username、password及dbname，这些属性对于每个连接来说都是一样的，所以就适合用享元模式来处理，建一个工厂类，将上述类似属性作为内部数据，其它的作为外部数据，在方法调用时，当做参数传进来，这样就节省了空间，减少了实例的数量。

看个例子：



看下数据库连接池的代码：

[java] view plaincopy

```
public class ConnectionPool {

    private Vector<Connection> pool;
```

```
    /*公有属性*/
    private String url = "jdbc:mysql://localhost:3306/test";
    private String username = "root";
    private String password = "root";
    private String driverClassName = "com.mysql.jdbc.Driver";

    private int poolSize = 100;
    private static ConnectionPool instance = null;
    Connection conn = null;

    /*构造方法，做一些初始化工作*/
    private ConnectionPool() {
        pool = new Vector<Connection>(poolSize);

        for (int i = 0; i < poolSize; i++) {
            try {
                Class.forName(driverClassName);
                conn = DriverManager.getConnection(url, username, passwor
d);
                pool.add(conn);
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    /* 返回连接到连接池 */
    public synchronized void release() {
        pool.add(conn);
    }
```

```
        /* 返回连接池中的一个数据库连接 */
        public synchronized Connection getConnection() {
            if (pool.size() > 0) {
                Connection conn = pool.get(0);
                pool.remove(conn);
                return conn;
            } else {
                return null;
            }
        }
    }
```

通过连接池的管理，实现了数据库连接的共享，不需要每一次都重新创建连接，节省了数据库重新创建的开销，提升了系统的性能！本章讲解了7种结构型模式，因为篇幅的问题，剩下的11种行为型模式，

本章是关于设计模式的最后一讲，会讲到第三种设计模式——
行为型模式，共11种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。这段时间一直在写关于设计模式的东西，终于写到一半了，写博文是个很费时间的东西，因为我得为读者负责，不论是图还是代码还是表述，都希望能尽量写清楚，以便读者理解，我想不论是我还是读者，都希望看到高质量的博文出来，从我本人出发，我会一直坚持下去，不断更新，源源动力来自于读者朋友们的不断支持，我会尽自己的努力，写好每一篇文章！希望大家能不断给出意见和建议，共同打造完美的博文！