深圳大学
Shenzhen University

第16讲

状态模式

软件体系结构与设计模式

Software Architecture & Design Pattern

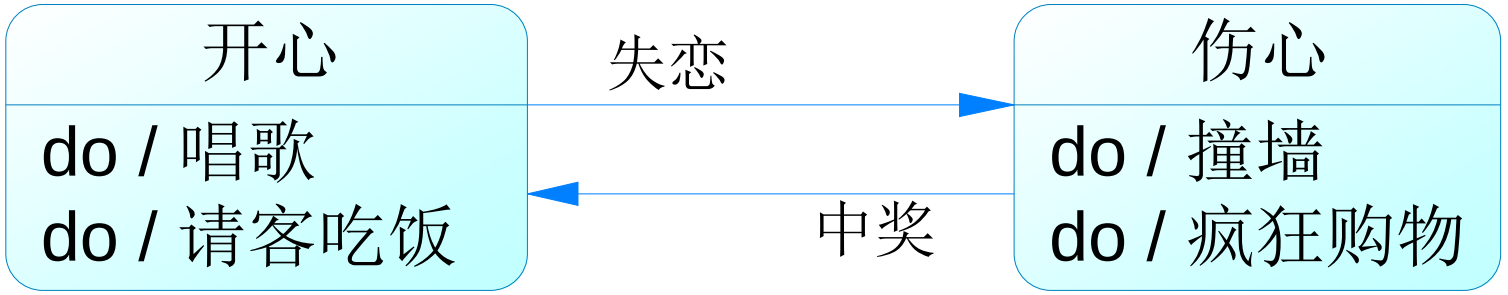
深圳大学计算机与软件学院

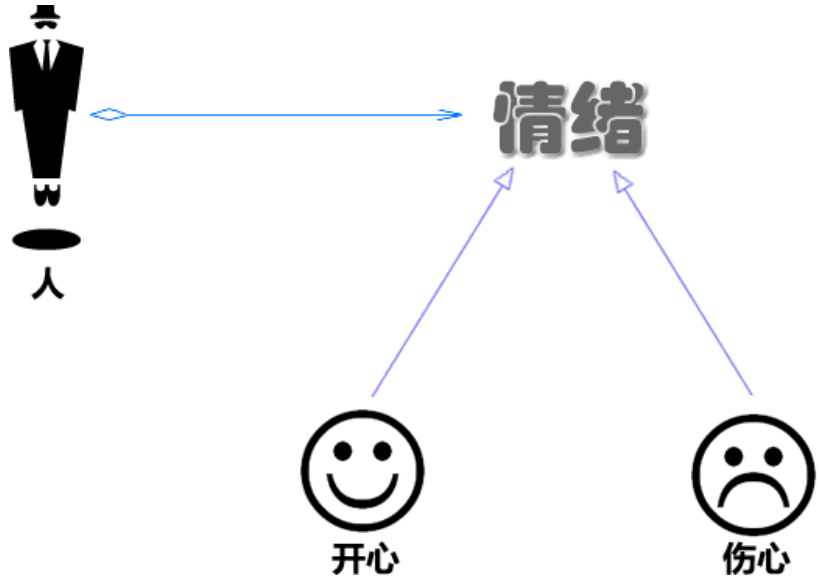
主要内容

- 状态模式动机与定义
- 状态模式结构与分析
- 状态模式实例与解析
- 状态模式效果与应用

状态模式动机

- **人有悲欢离合**





状态模式动机

- 在软件系统中：
  - 有些对象具有多种状态
  - 这些状态在某些情况下能够相互转换
  - 对象在不同的状态下将具有不同的行为
- 传统做法：使用复杂的条件判断语句来进行状态的判断和转换操作 → 导致代码的可维护性和灵活性下降 → 出现新的状态时，代码的扩展性很差，客户端代码需要进行相应的修改，违背了开闭原则

状态模式定义

- 对象行为型模式

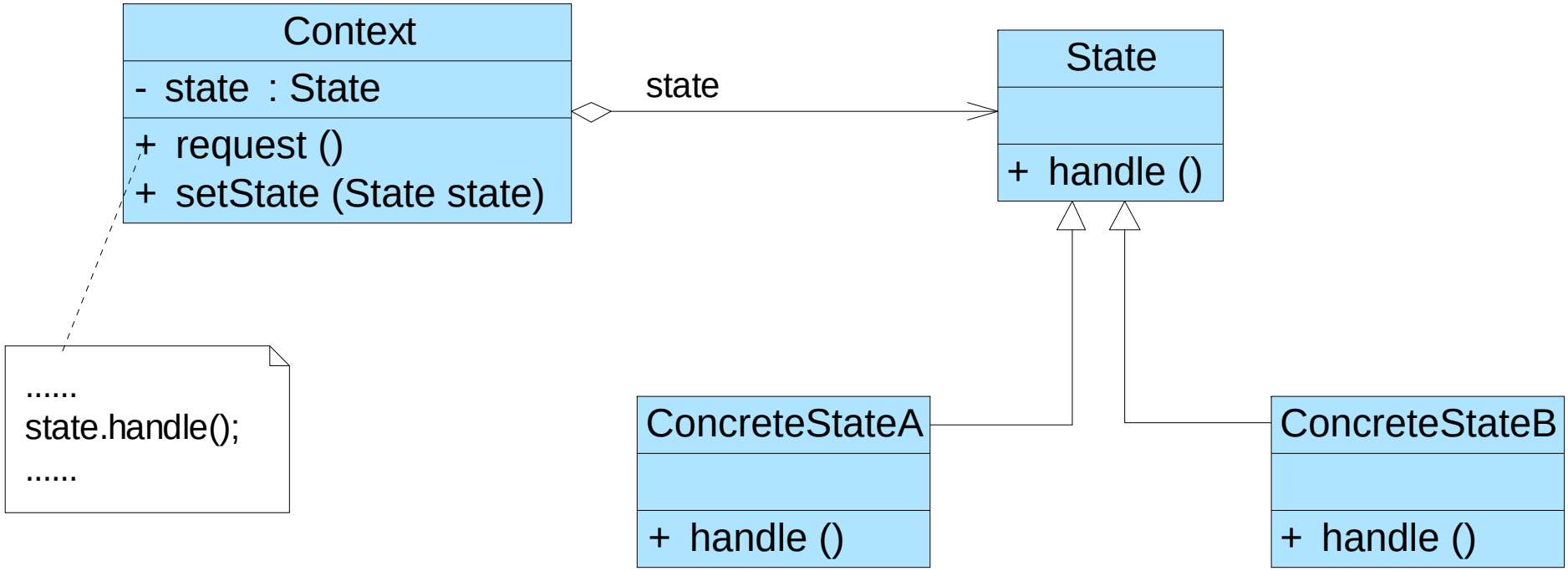
状态模式：允许一个对象在其内部状态改变时改变它的行为。对象看起来似乎修改了它的类。

State Pattern: Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.

- 又名状态对象(Objects for States)
- 用于解决系统中复杂对象的状态转换以及不同状态下行为的封装问题
- 将一个对象的状态从该对象中分离出来，封装到专门的状态类中，使得对象状态可以灵活变化
- 对于客户端而言，无须关心对象状态的转换以及对象所处的当前状态，无论对于何种状态的对象，客户端都可以一致处理

状态模式结构

- **状态模式**
- **包含如下角色：**
  - **Context: 环境类**
  - **State: 抽象状态类**
  - **ConcreteState: 具体状态类**



状态模式分析

- 状态转换的实现：
  - (1) 统一由环境类来负责状态之间的转换，环境类充当状态管理器(State Manager)角色

```
……
public void changeState()
{
    //判断属性值，根据属性值进行状态转换
    if (value == 0)
    {
        this.setState(new ConcreteStateA());
    }
    else if (value == 1)
    {
        this.setState(new ConcreteStateB());
    }
    ……
}
……
```

状态模式分析

- 状态转换的实现：
  - (1)由具体状态类来负责状态之间的转换，可以在具体状态类的业务方法中判断环境类的某些属性值，再根据情况为环境类设置新的状态对象，实现状态转换

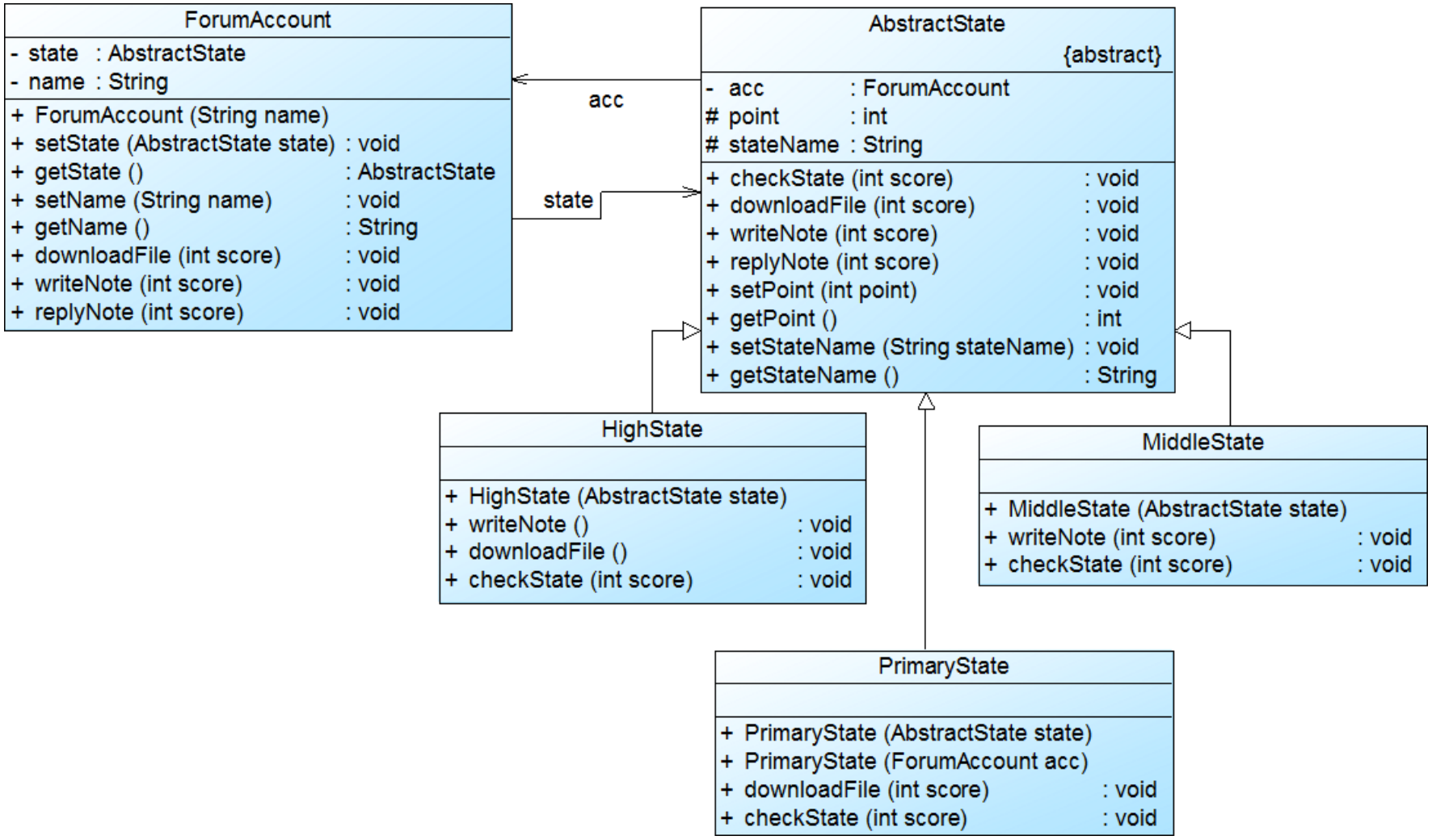
```
……
  public void changeState(Context ctx) {
     //根据环境对象中的属性值进行状态转换
     if (ctx.getValue() == 1) {
        ctx.setState(new ConcreteStateB());
     }
     else if (ctx.getValue() == 2) {
        ctx.setState(new ConcreteStateC());
     }
     ......
  }
  ……
```

论坛用户等级：实例说明

- 在某论坛系统中，用户可以发表留言，发表留言将增加积分；用户也可以回复留言，回复留言也将增加积分；用户还可以下载文件，下载文件将扣除积分。该系统用户分为三个等级，分别是新手、高手和专家，这三个等级对应三种不同的状态，这三种状态分别定义如下：
- (1) 如果积分小于100分，则为新手状态，用户可以发表留言、回复留言，但是不能下载文件。如果积分大于等于1000分，则转换为专家状态；如果积分大于等于100分，则转换为高手状态。
- (2) 如果积分大于等于100分但小于1000分，则为高手状态，用户可以发表留言、回复留言，还可以下载文件，而且用户在发表留言时可以获取双倍积分。如果积分小于100分，则转换为新手状态；如果积分大于等于1000分，则转换为专家状态；如果下载文件后积分小于0，则不能下载该文件。
- (3) 如果积分大于等于1000分，则为专家状态，用户可以发表留言、回复留言和下载文件，用户除了在发表留言时可以获取双倍积分外，下载文件只扣除所需积分的一半。如果积分小于100分，则转换为新手状态；如果积分小于1000分，但大于等于100，则转换为高手状态；如果下载文件后积分小于0，则不能下载该文件。

状态模式实例与解析

- **论坛用户等级：参考类图**



状态模式实例

- **论坛用户等级：参考代码**
- DesignPatterns之state包

Client.java

```
3  public class Client
4  {
5      public static void main(String args[])
6      {
7          ForumAccount account=new ForumAccount("张三");
8          account.writeNote(20);
9          System.out.println("----------------------------------------");
10         account.downloadFile(20);
11         System.out.println("----------------------------------------");
12         account.replyNote(100);
13         System.out.println("----------------------------------------");
14         account.writeNote(40);
15         System.out.println("----------------------------------------");
16         account.downloadFile(80);
17         System.out.println("----------------------------------------");
18         account.downloadFile(150);
19         System.out.println("----------------------------------------");
20         account.writeNote(1000);
21         System.out.println("----------------------------------------");
22         account.downloadFile(80);
23         System.out.println("----------------------------------------");
24     }
25 }
```

AbstractState.java

```java
public abstract class AbstractState
{
    protected ForumAccount acc;
    protected int point;
    protected String stateName;
    public abstract void checkState(int score);

    public void downloadFile(int score)
    {
        System.out.println(acc.getName() + "下载文件，扣除" + score + "积分。");
        this.point-=score;
        checkState(score);
        System.out.println("剩余积分为: " + this.point + "，当前级别为: " + acc.getState().stateName + "。");
    }

    public void writeNote(int score)
    {
        System.out.println(acc.getName() + "发布留言" + "，增加" + score + "积分。");
        this.point+=score;
        checkState(score);
        System.out.println("剩余积分为: " + this.point + "，当前级别为: " + acc.getState().stateName + "。");
    }

```

AbstractState.java

```
26      public void replyNote(int score)
27      {
28          System.out.println(acc.getName() + "回复留言，增加" + score + "积分。");
29          this.point+=score;
30          checkState(score);
31          System.out.println("剩余积分为: " + this.point + "，当前级别为: "
32          + acc.getState().stateName + "。");
33      }
34
35      public void setPoint(int point) {
36          this.point = point;
37      }
38
39      public int getPoint() {
40          return (this.point);
41      }
42
43      public void setStateName(String stateName) {
44          this.stateName = stateName;
45      }
46
47      public String getStateName() {
48          return (this.stateName);
49      }
50  }
```

ForumAccount.java

```
3  public class ForumAccount
4  {
5      private AbstractState state;
6      private String name;
7      public ForumAccount(String name)
8      {
9          this.name=name;
10         this.state=new PrimaryState(this);
11         System.out.println(this.name + "注册成功! ");
12         System.out.println("---------------------------------------");
13     }
14
15     public void setState(AbstractState state)
16     {
17         this.state=state;
18     }
19
20     public AbstractState getState()
21     {
22         return this.state;
23     }
24
25     public void setName(String name)
26     {
27         this.name=name;
28     }
```

ForumAccount.java

```
29
30      public String getName()
31      {
32          return this.name;
33      }
34
35      public void downloadFile(int score)
36      {
37          state.downloadFile(score);
38      }
39
40      public void writeNote(int score)
41      {
42          state.writeNote(score);
43      }
44
45      public void replyNote(int score)
46      {
47          state.replyNote(score);
48      }
49  }
```

HighState.java

```java
public class HighState extends AbstractState
{
    public HighState(AbstractState state)
    {
        this.acc=state.acc;
        this.point=state.getPoint();
        this.stateName="专家";
    }

    public void writeNote(int score)
    {
        System.out.println(acc.getName() + "发布留言" + "，增加" + score + "*2个积分。");
        this.point+=score*2;
        checkState(score);
        System.out.println("剩余积分为: " + this.point + "，当前级别为: "
        + acc.getState().stateName + "。");
    }

    public void downloadFile(int score)
    {
        System.out.println(acc.getName() + "下载文件，扣除" + score + "/2积分。");
        this.point-=score/2;
        checkState(score);
        System.out.println("剩余积分为: " + this.point + "，当前级别为: "
        + acc.getState().stateName + "。");    }

```

HighState.java

```
29      public void checkState(int score)
30      {
31          if(point<0)
32          {
33              System.out.println("余额不足，文件下载失败！");
34              this.point+=score;
35          }
36          else if(point<=100)
37          {
38              acc.setState(new PrimaryState(this));
39          }
40          else if(point<=1000)
41          {
42              acc.setState(new MiddleState(this));
43          }
44      }
45  }
```

状态模式实例与解析

MiddleState.java

```
3  public class MiddleState extends AbstractState
4  {
5      public MiddleState(AbstractState state)
6      {
7          this.acc=state.acc;
8          this.point=state.getPoint();
9          this.stateName="高手";
10     }
11
12     public void writeNote(int score)
13     {
14         System.out.println(acc.getName() + "发布留言" + "，增加" + score + "*2个积分。");
15         this.point+=score*2;
16         checkState(score);
17         System.out.println("剩余积分为: " + this.point + "，当前级别为: "
18         + acc.getState().stateName + "。");
19     }
```

状态模式实例与解析

MiddleState.java

```
20
21      public void checkState(int score)
22      {
23          if(point>=1000)
24          {
25              acc.setState(new HighState(this));
26          }
27          else if(point<0)
28          {
29              System.out.println("余额不足，文件下载失败！");
30              this.point+=score;
31          }
32          else if(point<=100)
33          {
34              acc.setState(new PrimaryState(this));
35          }
36      }
37  }
```

PrimaryState.java

```
3  public class PrimaryState extends AbstractState
4  {
5      public PrimaryState(AbstractState state)
6      {
7          this.acc=state.acc;
8          this.point=state.getPoint();
9          this.stateName="新手";
10     }
11
12     public PrimaryState(ForumAccount acc)
13     {
14         this.point=0;
15         this.acc=acc;
16         this.stateName="新手";
17     }
18
19     public void downloadFile(int score)
20     {
21         System.out.println("对不起，"
22         + acc.getName() + "，您没有下载文件的权限！");
23     }
24
```

状态模式实例与解析

PrimaryState.java

```
25      public void checkState(int score)
26      {
27          if(point>=1000)
28          {
29              acc.setState(new HighState(this));
30          }
31          else if(point>=100)
32          {
33              acc.setState(new MiddleState(this));
34          }
35      }
36  }
```

状态模式效果与应用

- **状态模式优点：**
  - 封装了状态的转换规则，可以对状态转换代码进行集中管理，而不是分散在一个个业务方法中
  - 将所有与某个状态有关的行为放到一个类中，只需要注入一个不同的状态对象即可使环境对象拥有不同的行为
  - 允许状态转换逻辑与状态对象合成一体，而不是提供一个巨大的条件语句块，可以避免使用庞大的条件语句来将业务方法和状态转换代码交织在一起
  - 可以让多个环境对象共享一个状态对象，从而减少系统中对象的个数

状态模式效果与应用

- **状态模式缺点：**
  - 会增加系统中类和对象的个数，导致系统运行开销增大
  - 结构与实现都较为复杂，如果使用不当将导致程序结构和代码混乱，增加系统设计的难度
  - 对开闭原则的支持并不太好，增加新的状态类需要修改负责状态转换的源代码，否则无法转换到新增状态；而且修改某个状态类的行为也需要修改对应类的源代码

状态模式效果与应用

- **在以下情况下可以使用状态模式：**
  - 对象的行为依赖于它的状态（例如某些属性值），状态的改变将导致行为的变化
  - 在代码中包含大量与对象状态有关的条件语句，这些条件语句的出现会导致代码的可维护性和灵活性变差，不能方便地增加和删除状态，并且导致客户类与类库之间的耦合增强

谢谢