Lec04 作业

1、Java 的对象如何算相同

➢ 举出一个场景，你必须改写现有类库的 equals 方法

解：运算符'=='和对象中的 equals()方法都可以用来判断 Java 的对象是否相同。

1) 运算符'=='：比较的两个对象的地址是否相同
2) equals()方法：默认的 equals()方法和运算符'=='一样，对地址进行比较，通过改写该方法可以做到判断两个对象内容是否相同。

```
1.  class Student{
2.      String name;
3.      int age;
4.      String major;
5.      public Student() {
6.      }
7.      public Student(String name, int age, String major) {
8.          this.name = name;
9.          this.age = age;
10.         this.major = major;
11.     }
12.     @Override
13.     public boolean equals(Object obj) {
14.         if (obj == this) return true;
15.         if (!(obj instanceof Student)) return false;
16.         Student student = (Student) obj;
17.         return name.equals(student.name) && age == student.age &&
    major.equals(student.major);
18.     }
19.     public String getName() {
20.         return this.name;
21.     }
22.     public void setName(String name) {
23.         this.name = name;
24.     }
25.     public int getAge() {
26.         return this.age;
27.     }
28.     public void setAge(int age) {
29.         this.age = age;
30.     }
31.     public String getMajor() {
32.         return this.major;
33.     }
```

```
    public void setMajor(String major) {
        this.major = major;
    }
    @Override
    public String toString() {
        return "{" +
            " name='" + getName() + "'" +
            ", age='" + getAge() + "'" +
            ", major='" + getMajor() + "'" +
            "}";
    }
    public static void main(String[] args) {
        Student JiYi1 = new Student("JiYi", 23, "CS");
        Student JiYi2 = new Student("JiYi", 23, "CS");
        System.out.println("JiYi1 == JiYi2: "+(JiYi1 == JiYi2));
        System.out.println("JiYi1.equals(JiYi2): "+JiYi1.equals(JiYi2));
    }
}
```

```
输出结果为：
JiYi1 == JiYi2: false
JiYi1.equals(JiYi2): true
```

运算符'=='依据地址判断两个对象不同，改写后 equals()方法依据内容判断两者相同。

2、总结 JavaScript 语言的面向对象特征，你认为 JavaScript(是/否)归属于面向对象语言的理由是什么？

解：JavaScript 的面向对象特性是基于原型链的。使用原型对对象进行封装，使用原型链、基于伪装或基于组合方式实现继承。通过父类引用指向子类实例的方式实现了多态。

因此 JavaScript 是面相对象的语言。

3、class TalkingClock 是一个类，class TimePrinter 是一个类，为什么 TimePrinter 可以使用 TalkingClock 的私有变量，请分析这么使用的潜在安全风险。

解：TimePrinter 类是定义在 TalkingClock 中的内部类，内部类可以直接访问外部类中包括私有域在内的元素。

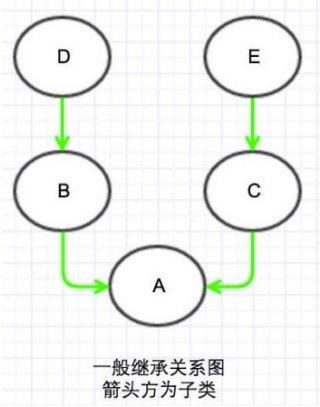
安全风险：如果内部类访问了私有数据域，就有可能通过附加在外围类所在的包中的其他类访问它们。

4、多态作业

解：见“多态.pdf”

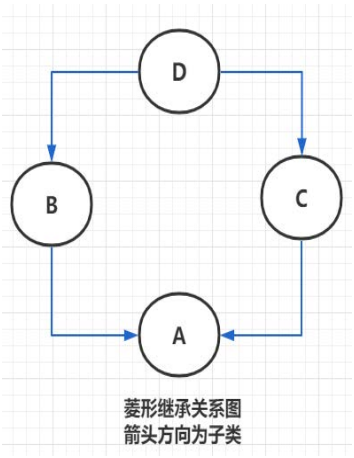
5、查阅 Python 中 MRO 生成算法(DFS、BFS 和 C3 算法)，并根据 C3 算法写出如下两幅图的 MRO 列表

解：



一般继承关系图
箭头方为子类

mro(D) = [D, O]
mro(B) = [B] + merge(mro(D), [D])
= [B] + merge([D, O], [D])
= [B, D] + merge([O])
= [B, D, O] mro(E)
= [E, O] mro(C)
= [C] + merge(mro(E), [E])
= [C] + merge([E, O], [E])
= [C, E] + merge([O])
= [C, E, O]
mro(A) = [A] + merge(mro(B), mro(C), [B, C])
= [A] + merge([B, D, O], [C, E, O], [B, C])
= [A, B] + merge([D, O], [C, E, O], [C])
= [A, B, D] + merge([O], [C, E, O], [C])
= [A, B, D, C] + merge([O], [E, O])
= [A, B, D, C, E] + merge([O], [O])
= [A, B, D, C, E, O]



菱形继承关系图
箭头方向为子类

mro(D) = [D, O]
mro(B) = [B] + merge(mro(D), [D])
= [B] + merge([D, O], [D])
= [B, D] + merge([O])
= [B, D, O]
mro(C) = [C] + merge(mro(D), [D])
= [C] + merge([D, O], [D])
= [C, D] + merge([O])
= [C, D, O]
mro(A) = [A] + merge(mro(B), mro(C), [B, C])
= [A] + merge([B, D, O], [C, D, O], [B, C])
= [A, B] + merge([D, O], [C, D, O], [C])
= [A, B, C] + merge([D, O], [D, O])
= [A, B, C, D, O]