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。
3. **class** Student{
4. String name;
5. **int** age;
6. String major;
7. **public** Student() {
8. }
9. **public** Student(String name, **int** age, String major) {
10. **this**.name = name;
11. **this**.age = age;
12. **this**.major = major;
13. }
14. @Override
15. **public** **boolean** equals(Object obj) {
16. **if** (obj == **this**) **return** **true**;
17. **if** (!(obj **instanceof** Student)) **return** **false**;
18. Student student = (Student) obj;
19. **return** name.equals(student.name) && age == student.age && major.equals(student.major);
20. }
21. **public** String getName() {
22. **return** **this**.name;
23. }
24. **public** **void** setName(String name) {
25. **this**.name = name;
26. }
27. **public** **int** getAge() {
28. **return** **this**.age;
29. }
30. **public** **void** setAge(**int** age) {
31. **this**.age = age;
32. }
33. **public** String getMajor() {
34. **return** **this**.major;
35. }
36. **public** **void** setMajor(String major) {
37. **this**.major = major;
38. }
39. @Override
40. **public** String toString() {
41. **return** "{" +
42. " name='" + getName() + "'" +
43. ", age='" + getAge() + "'" +
44. ", major='" + getMajor() + "'" +
45. "}";
46. }
47. **public** **static** **void** main(String[] args) {
48. Student JiYi1 = **new** Student("JiYi", 23, "CS");
49. Student JiYi2 = **new** Student("JiYi", 23, "CS");
50. System.out.println("JiYi1 == JiYi2: "+(JiYi1 == JiYi2));
51. System.out.println("JiYi1.equals(JiYi2): "+JiYi1.equals(JiYi2));
52. }
53. }
54. 输出结果为：
55. JiYi1 == JiYi2: false
56. 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。

1. 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。

1. 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。

1. 多态作业

解：见“多态.pdf”

1. 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

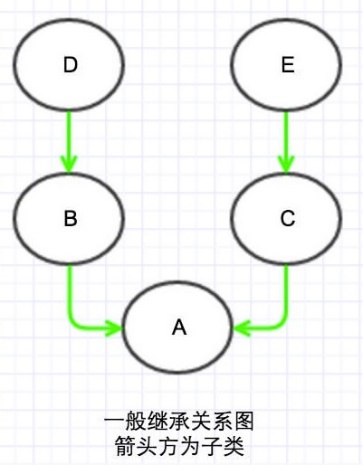
解：

mro(D) = [D, O]

mro(E) = [E, 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(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])

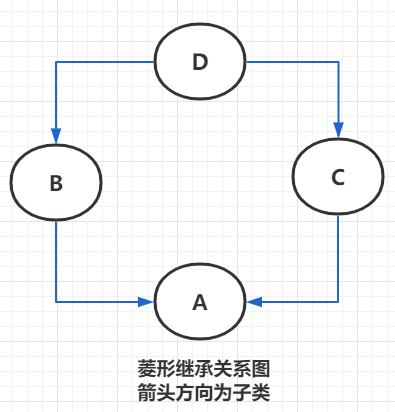
= [A, B, D] + merge([O], [C, E, O], [C])

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

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

= [A, B, D, C, E, O]

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]