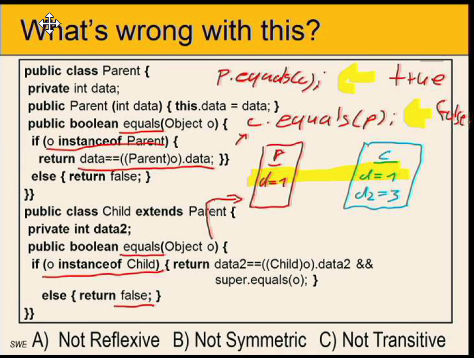


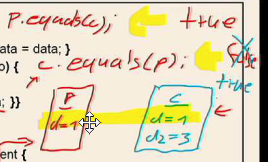
B)

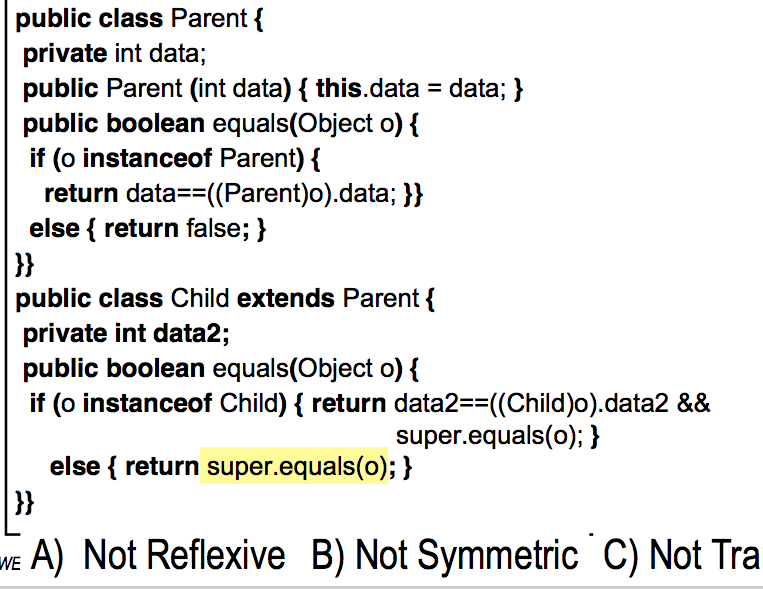
InsensitiveStr. equals(str) 🡪 true

str.equals(InsensitiveStr) 🡪 false

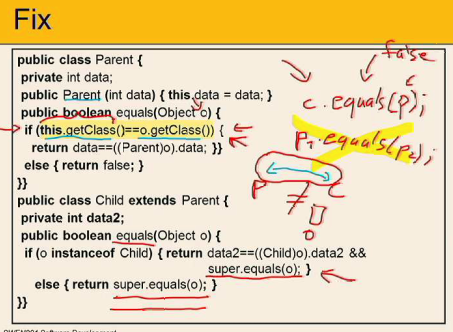


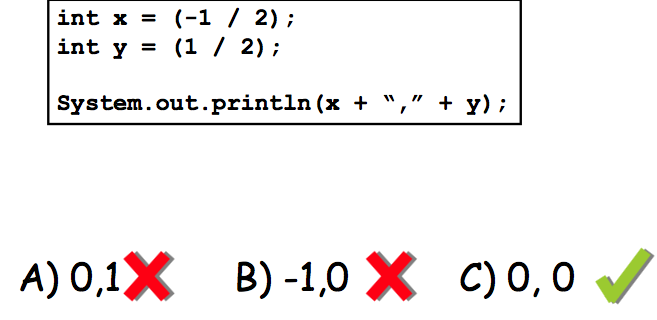
B)

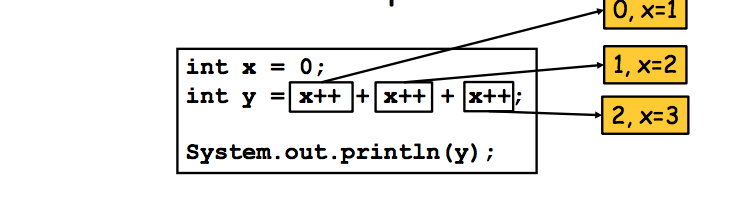


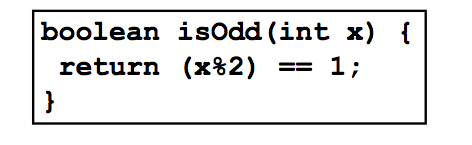


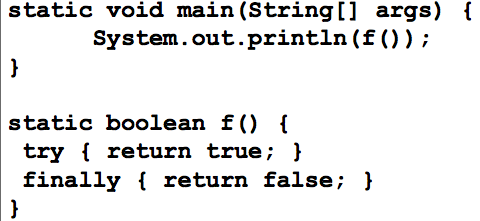


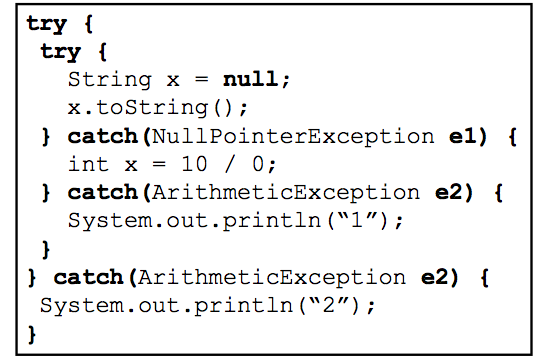


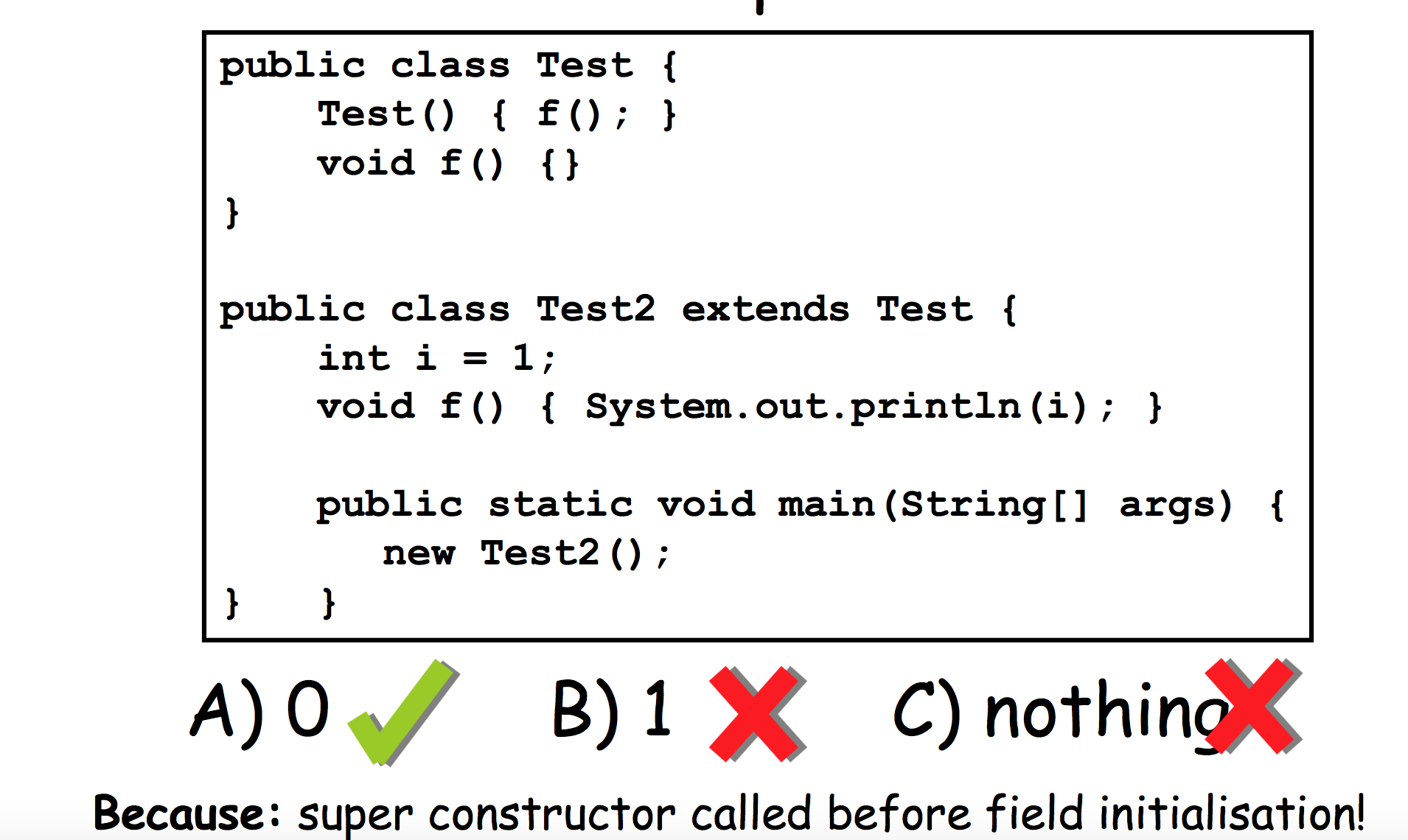


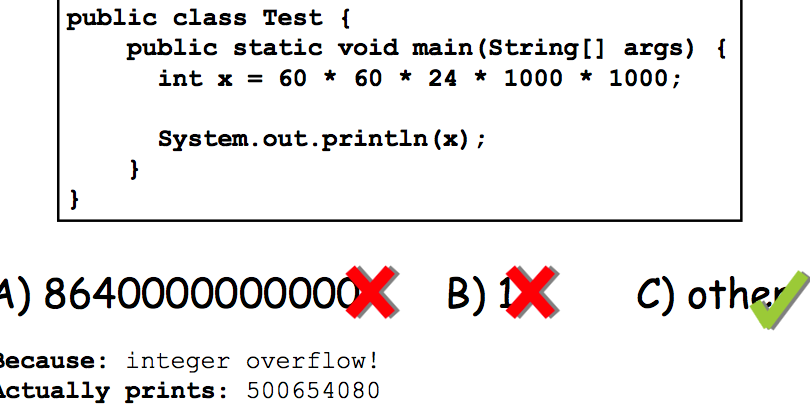




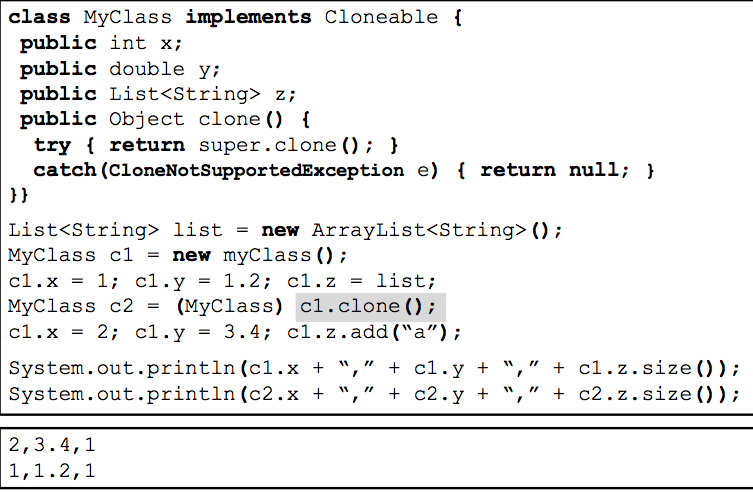


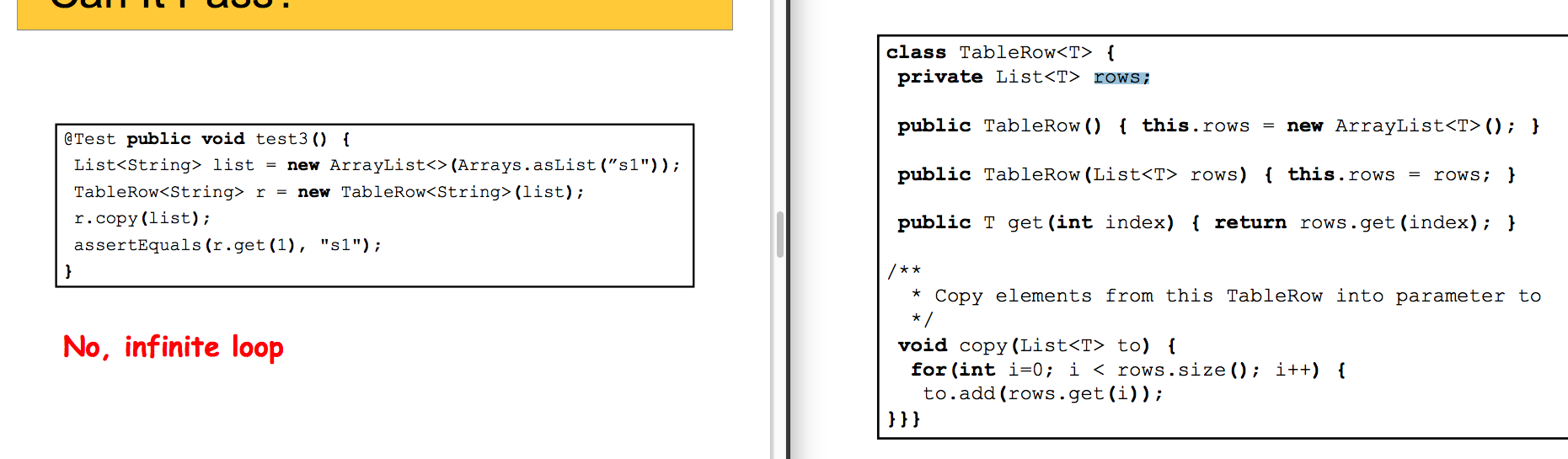


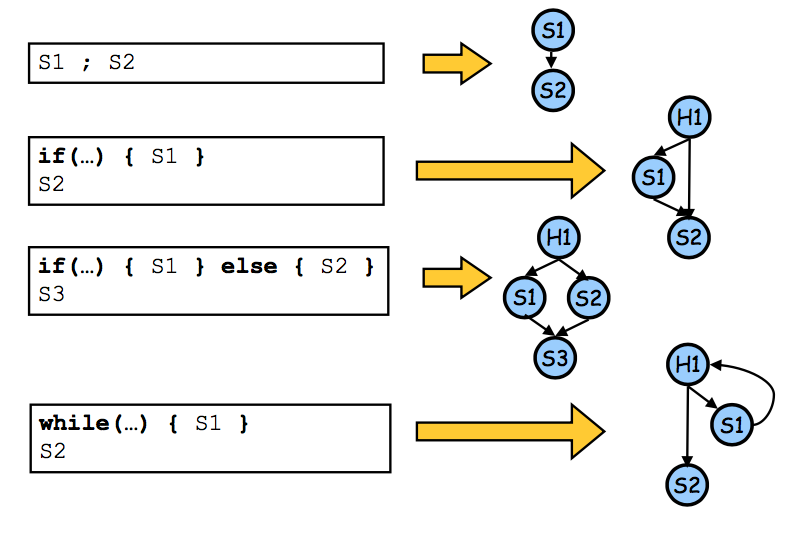
­

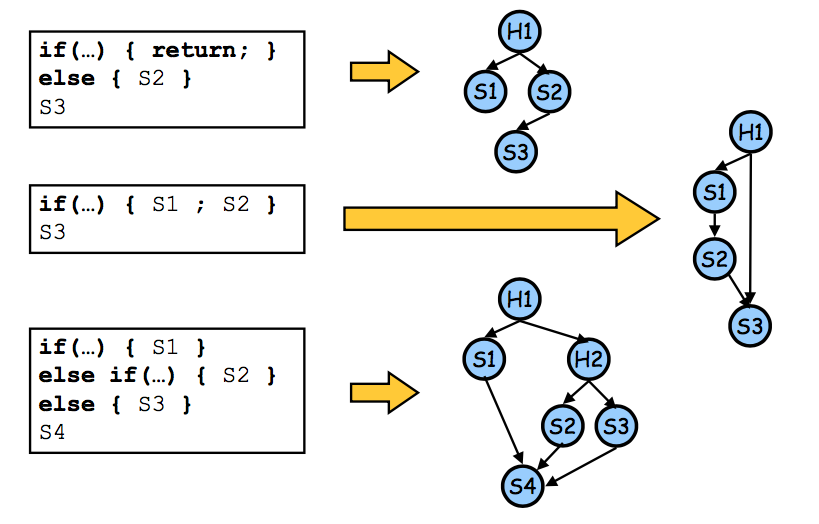


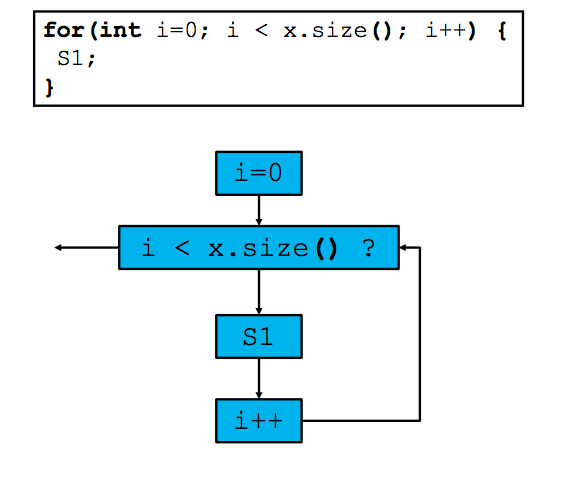
The int data type is a 32-bit signed two's complement integer. It has a minimum value of -**2,147,483,648** and a maximum value of **2,147,483,647**

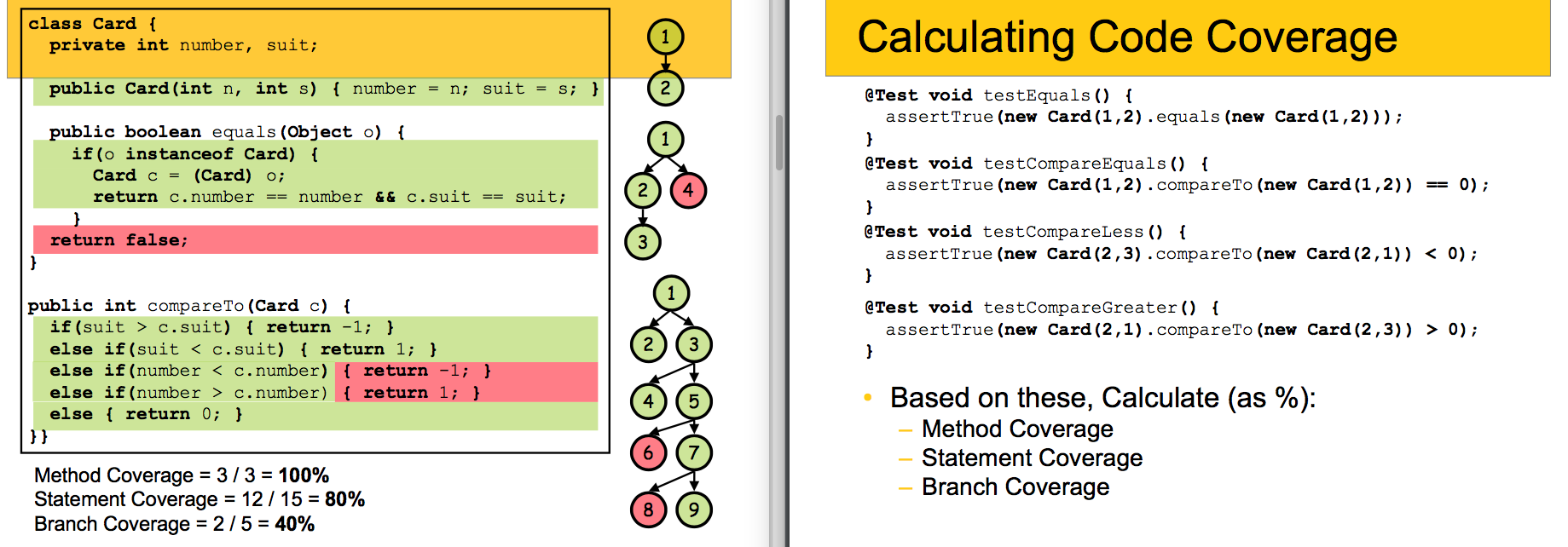


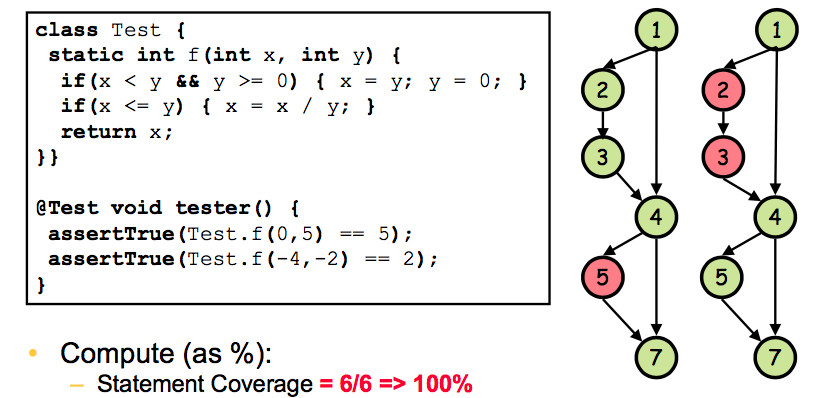


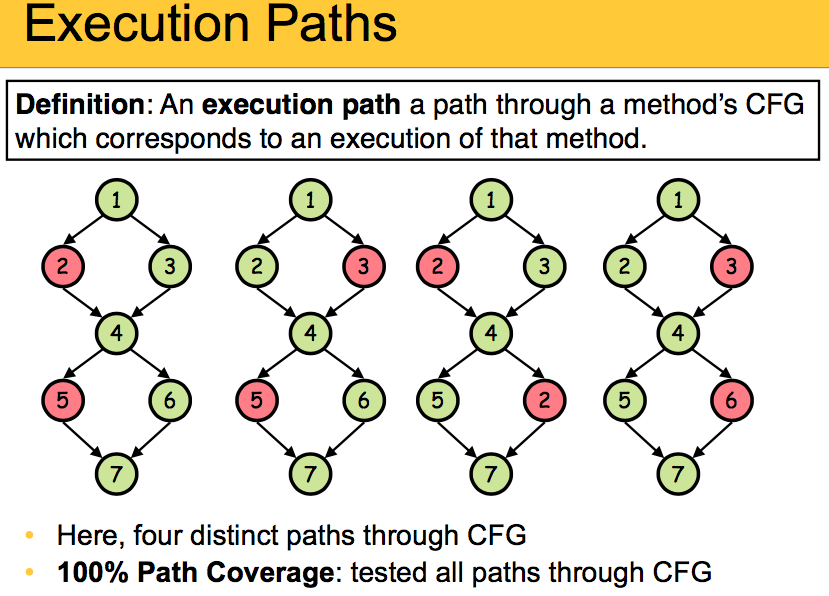


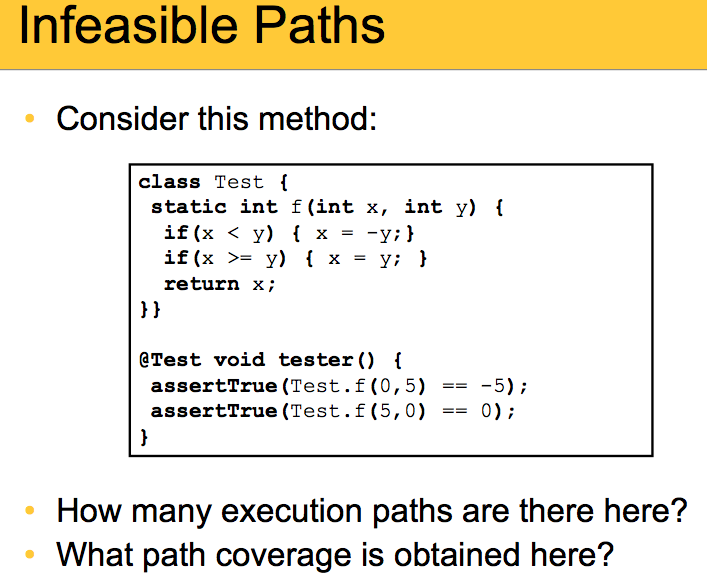


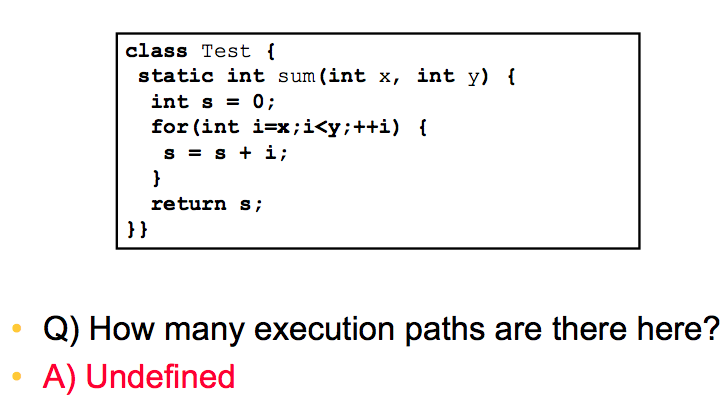


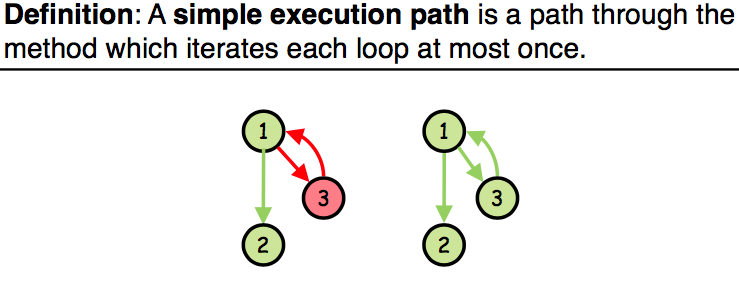


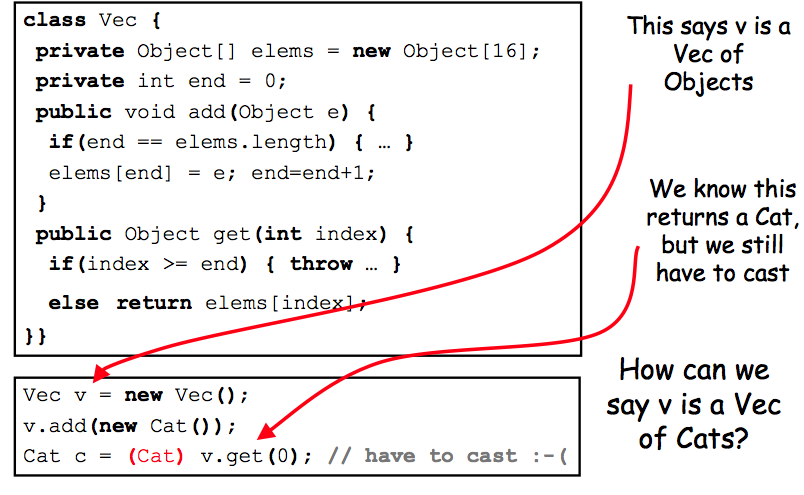


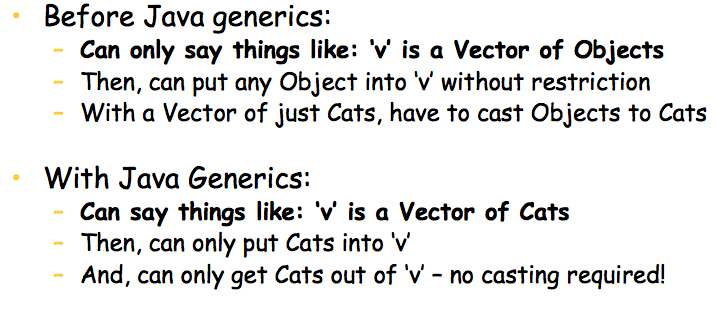


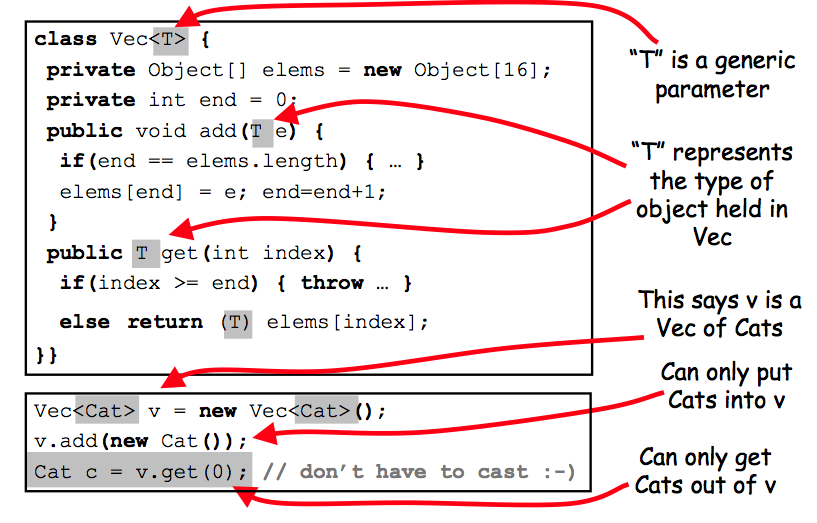


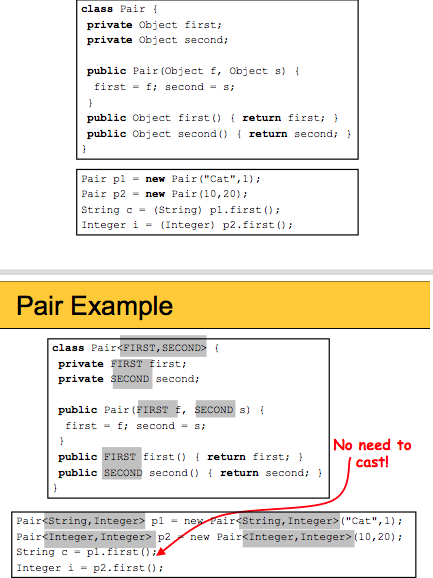


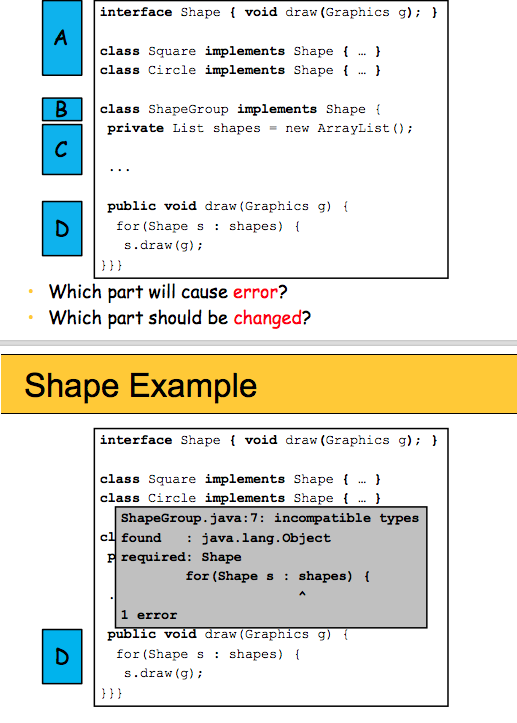


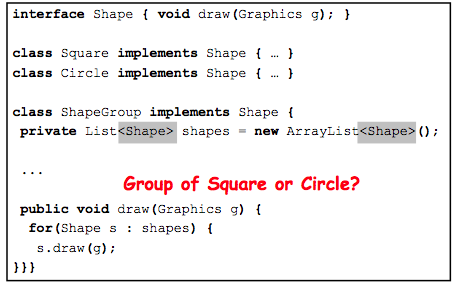


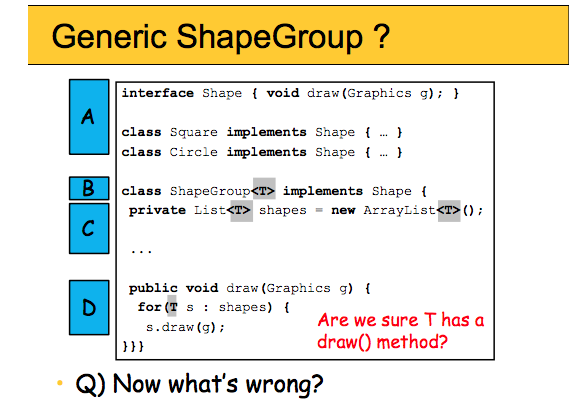


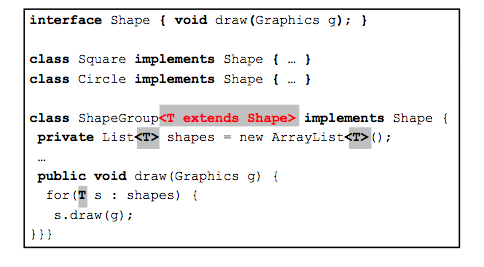


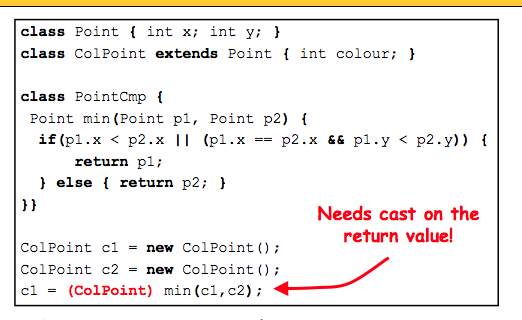


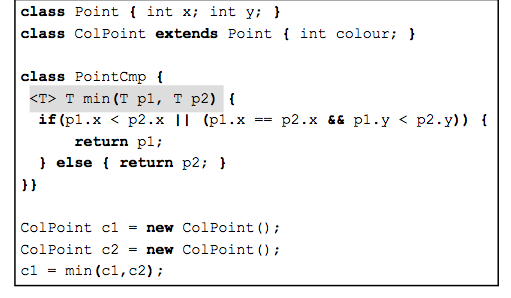


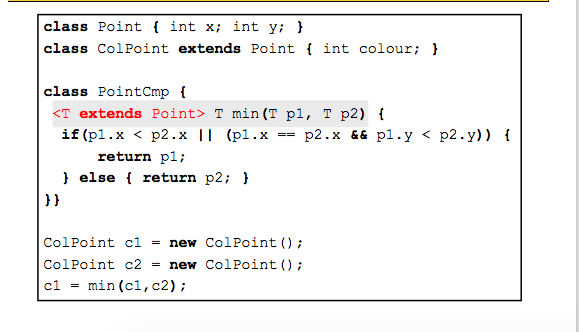


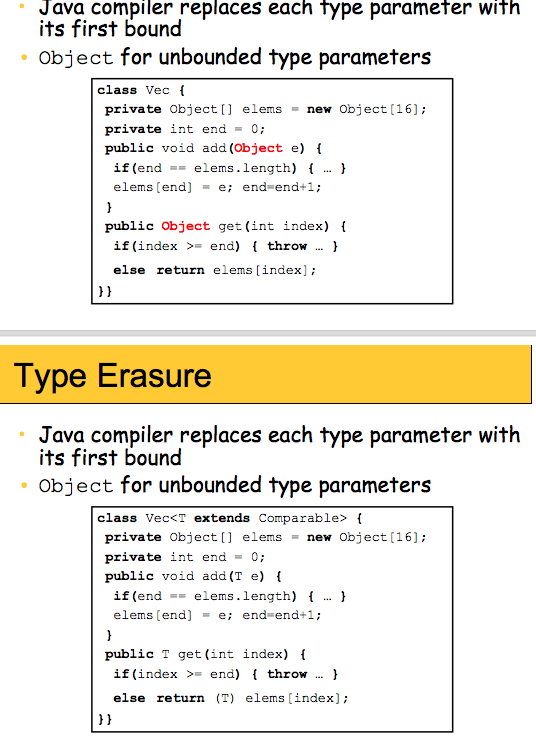


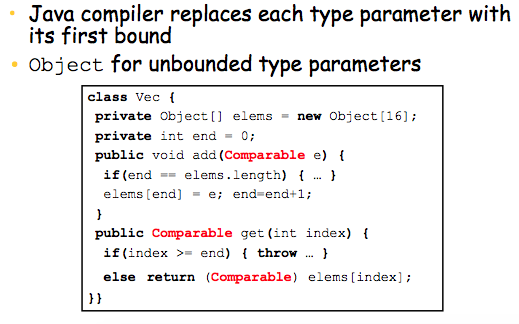


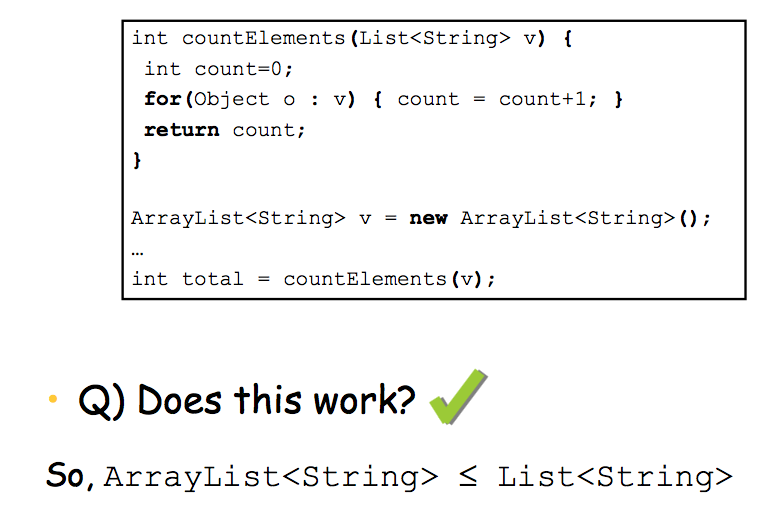


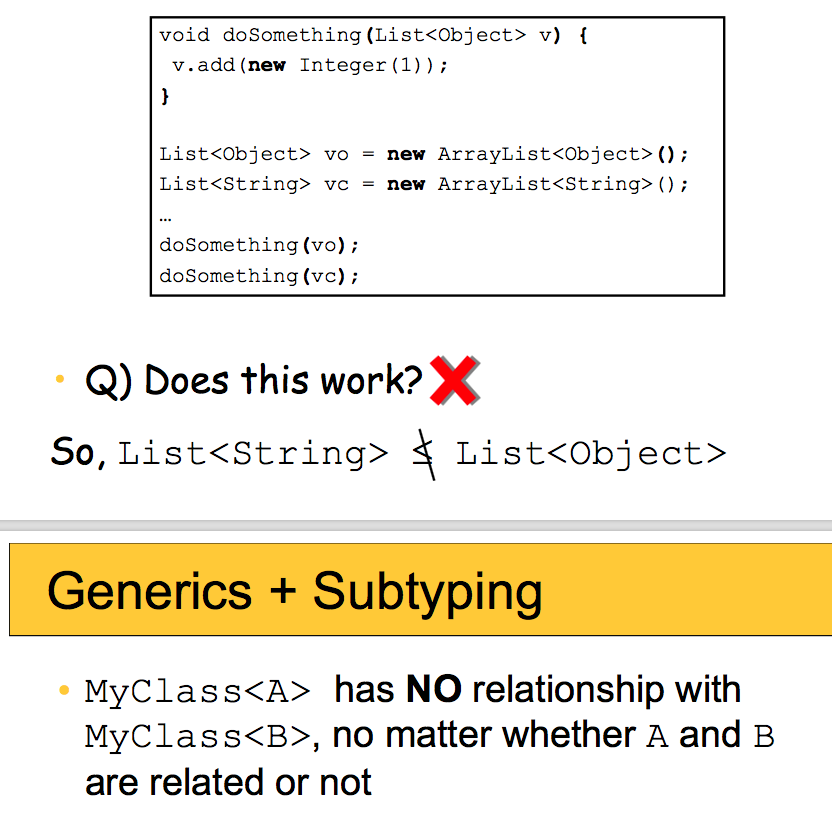


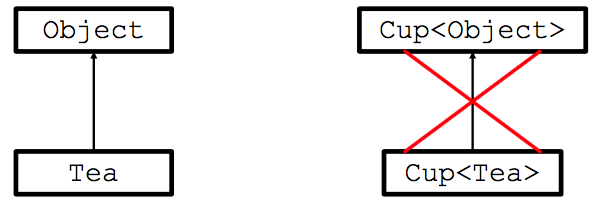


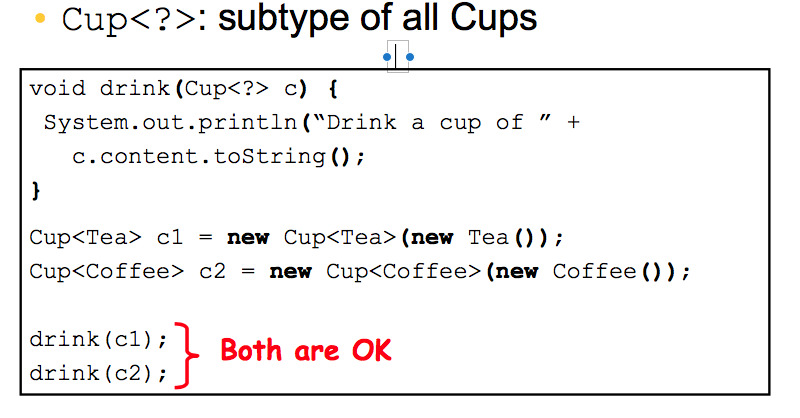


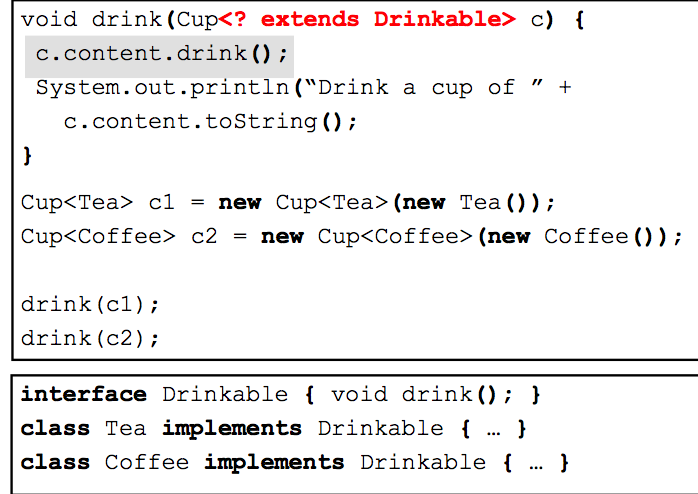


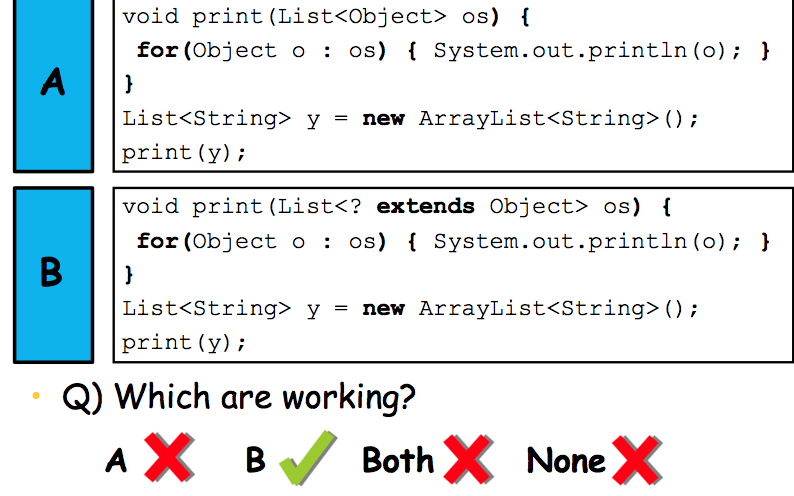


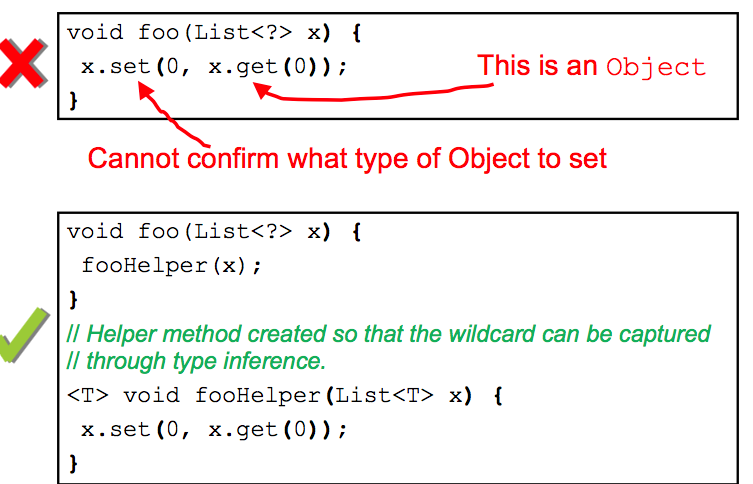


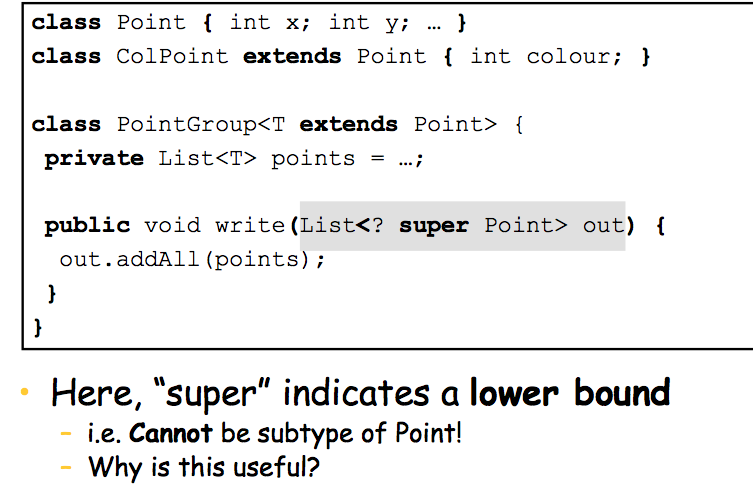


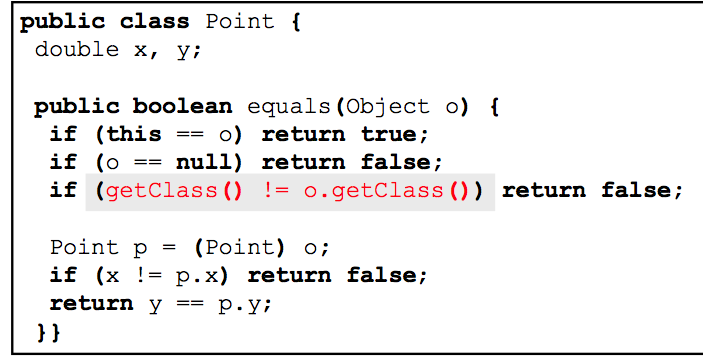


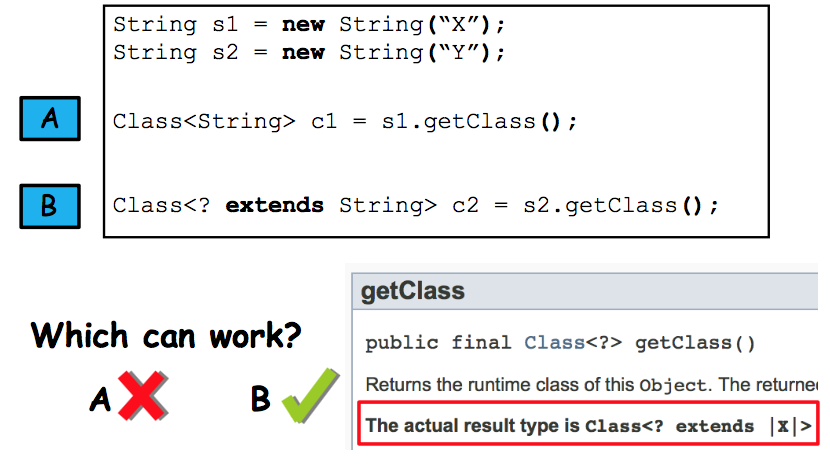


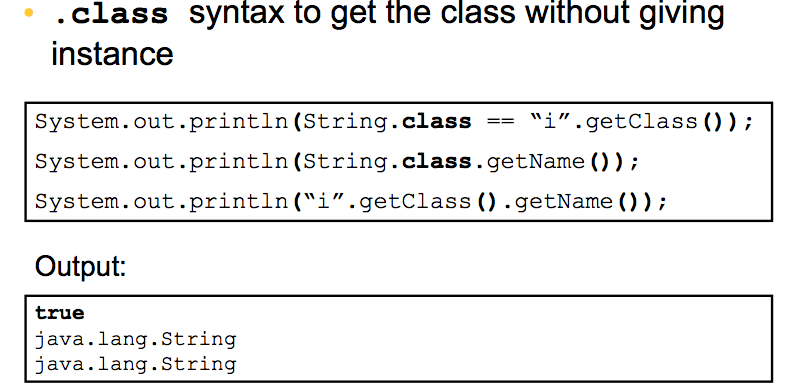


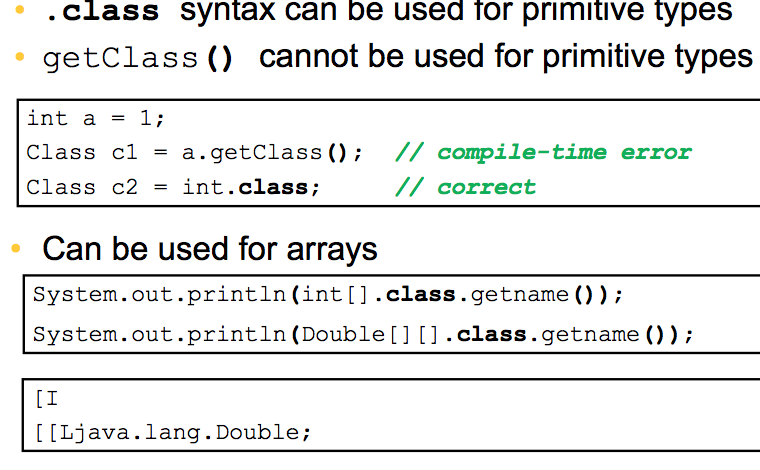


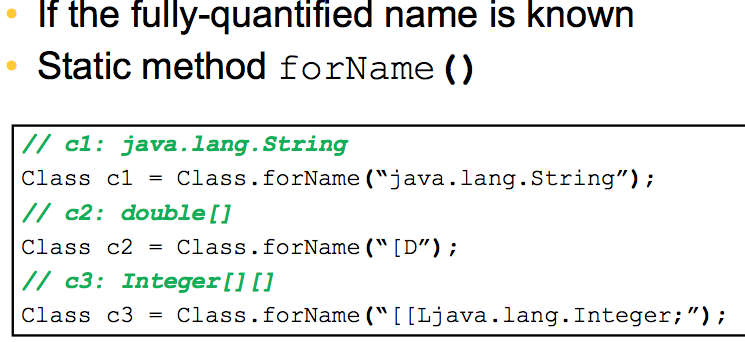


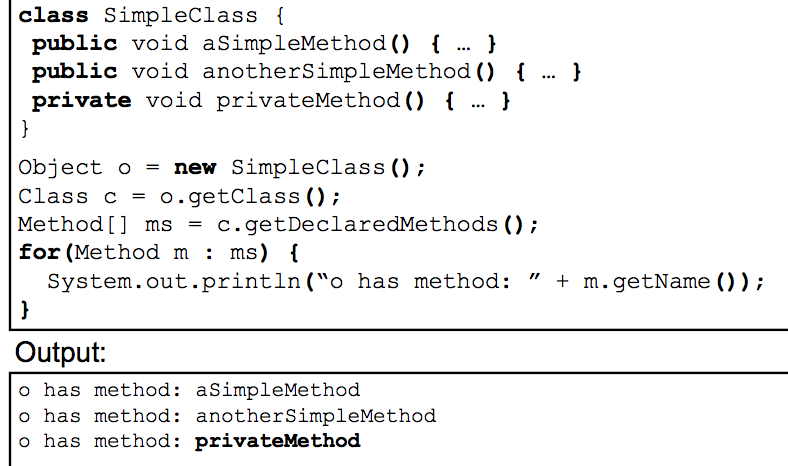


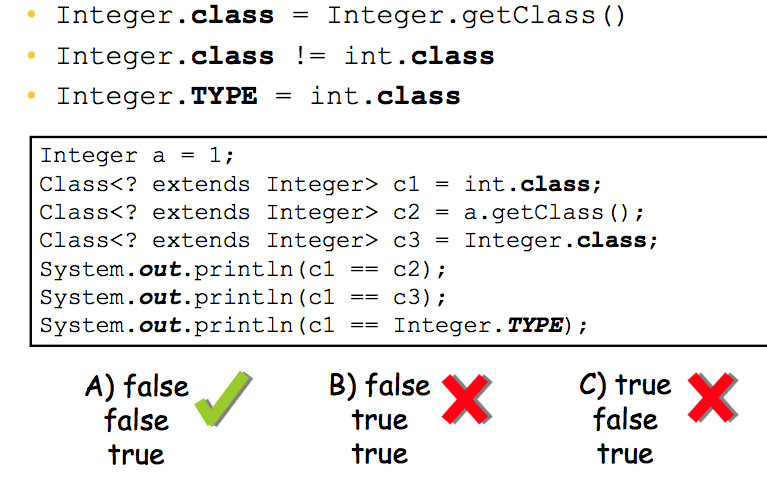


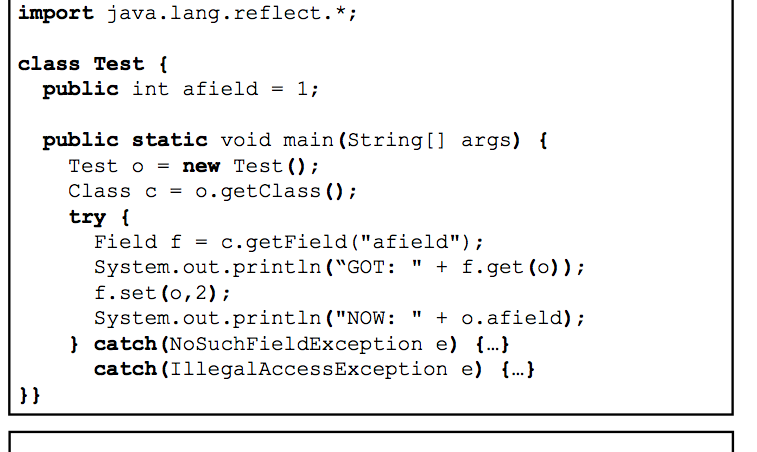






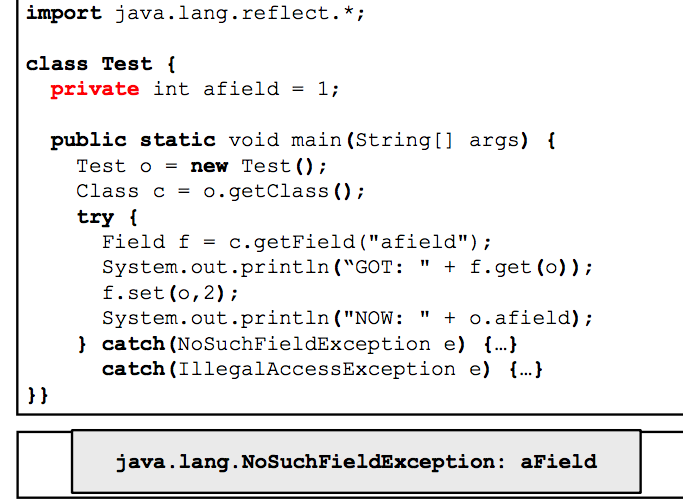


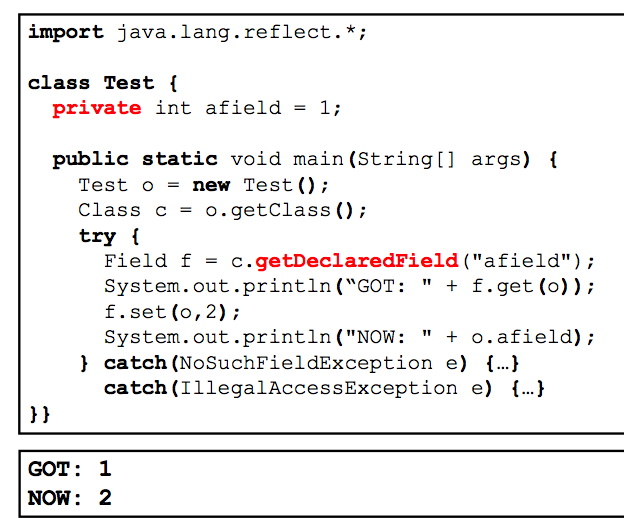


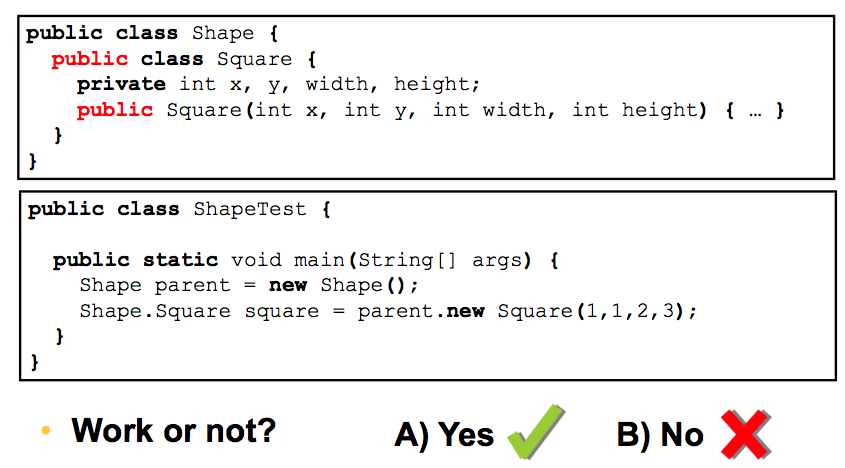


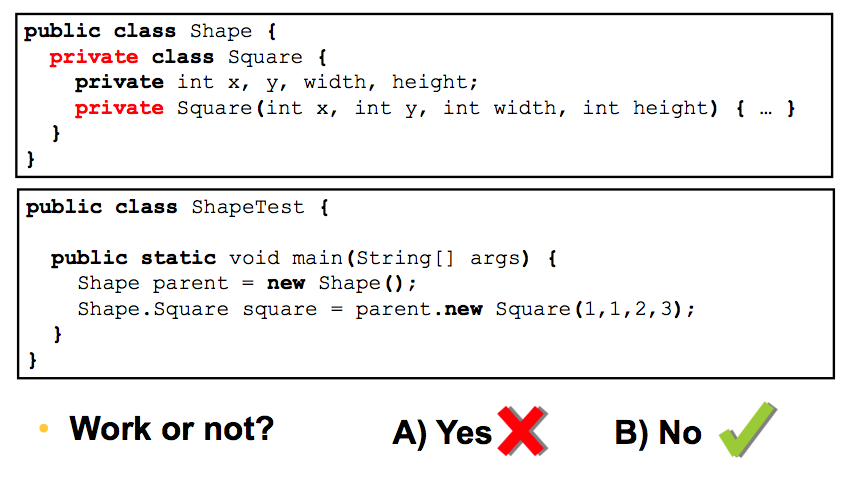
GOT: 1

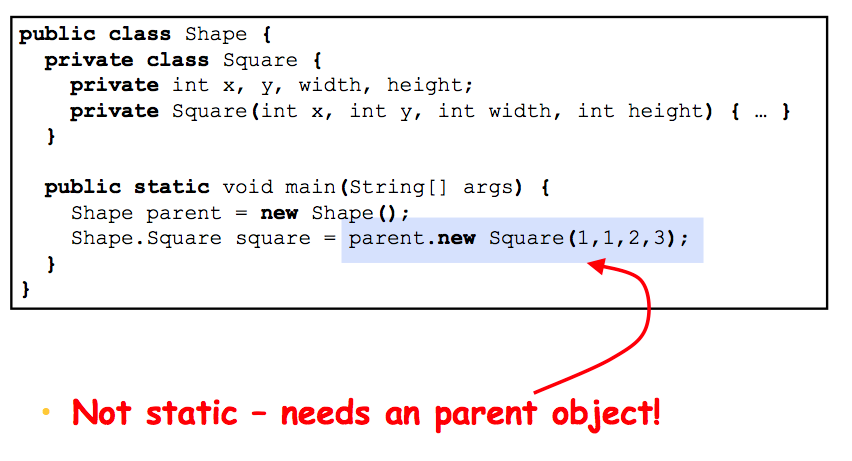
NOW: 2

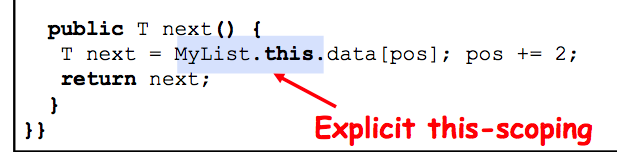


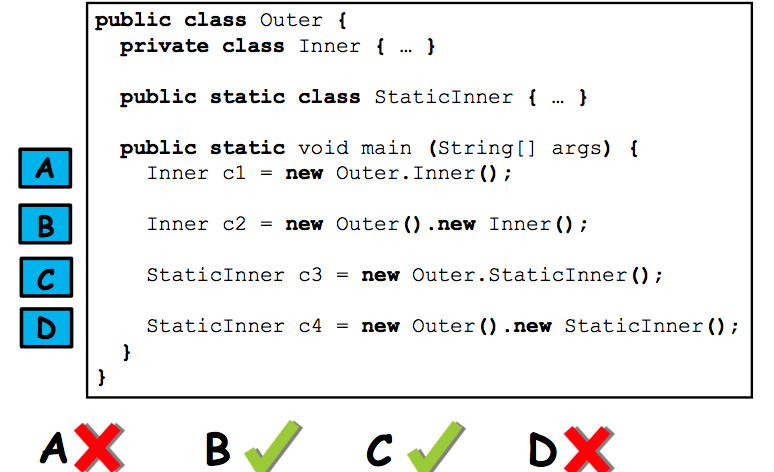


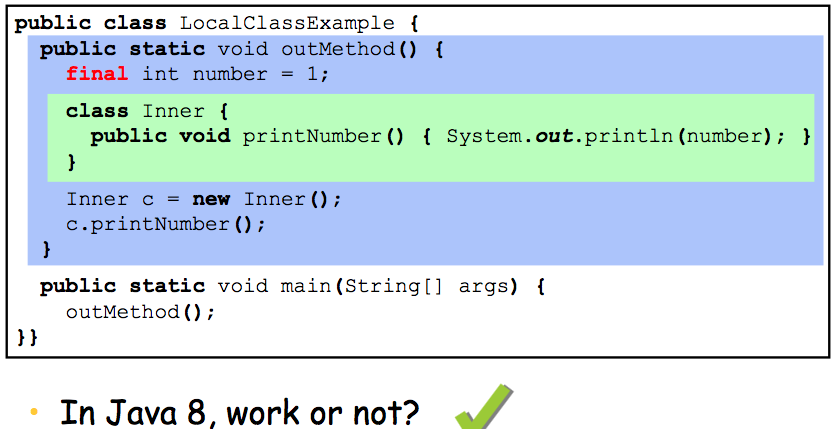


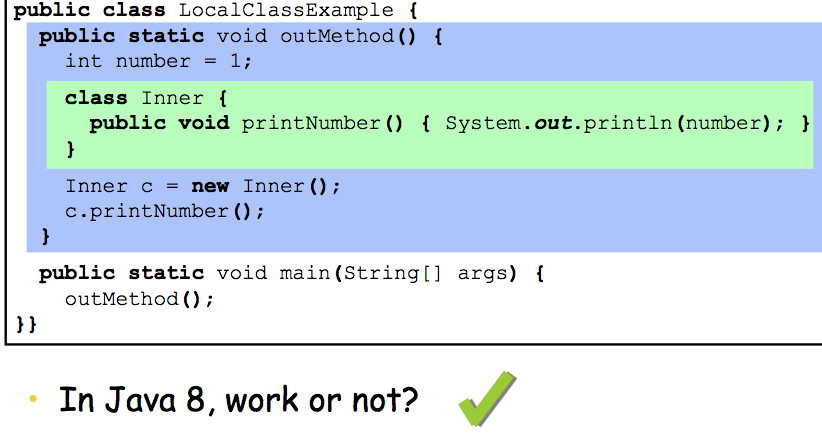


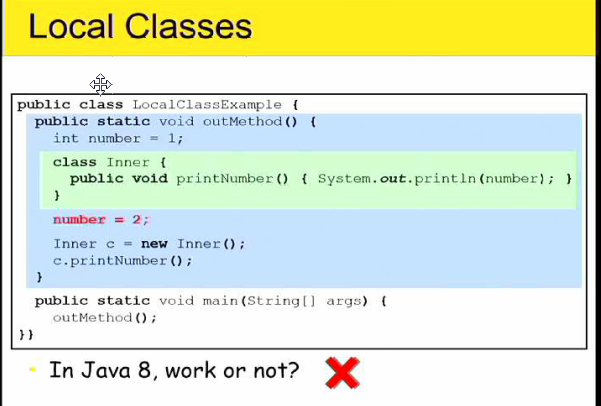


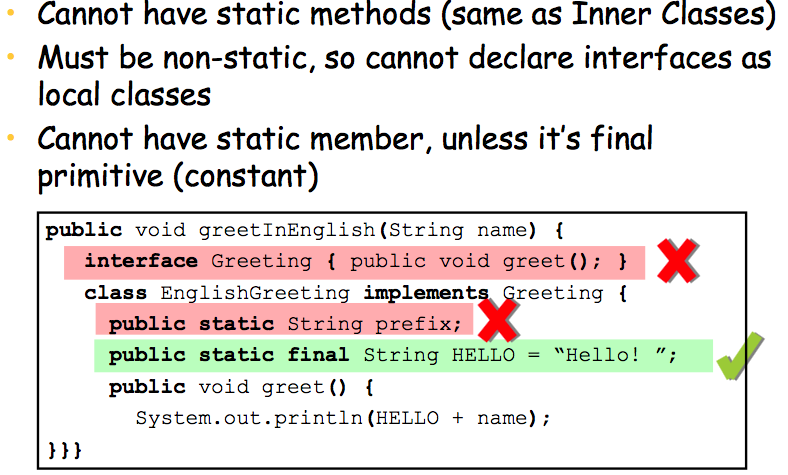


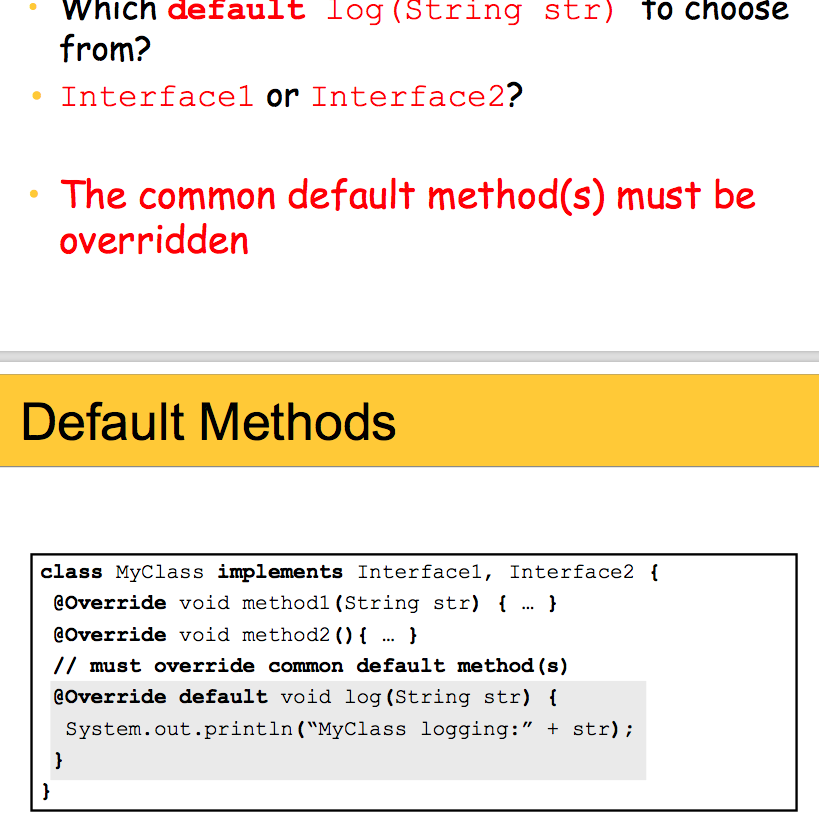


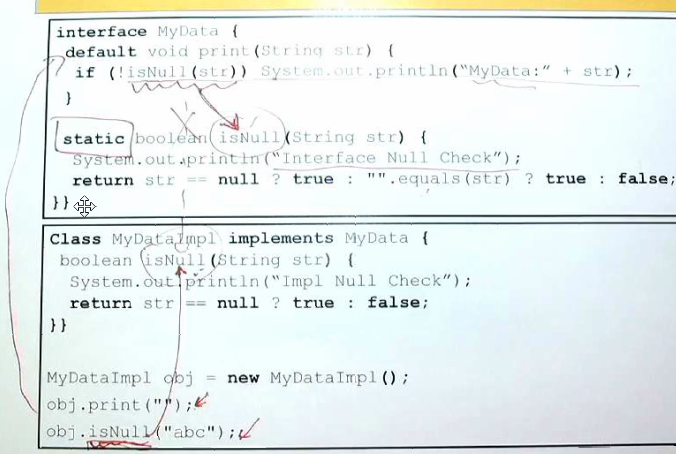


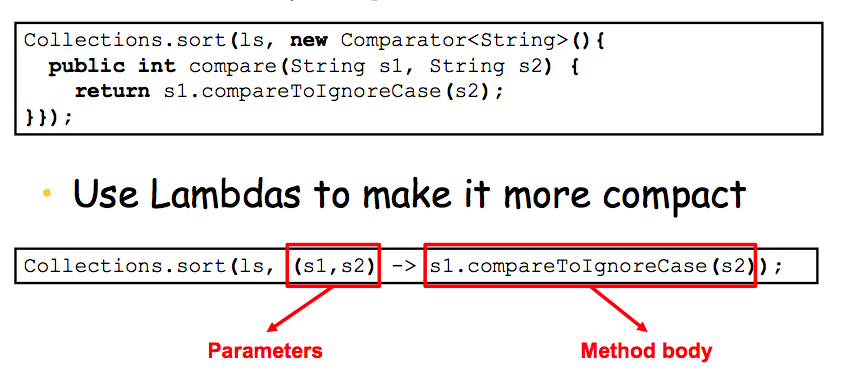


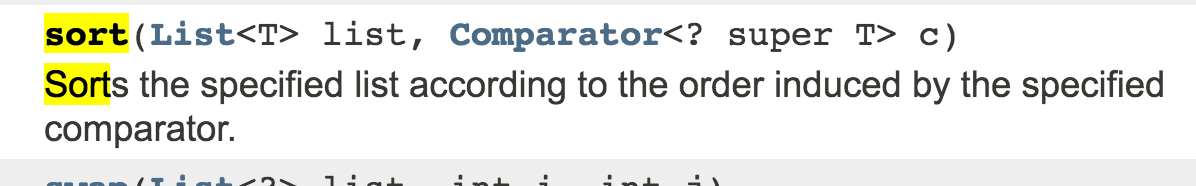


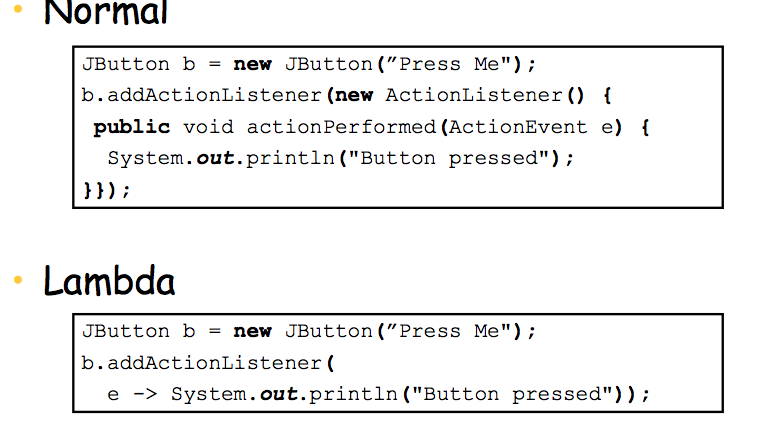


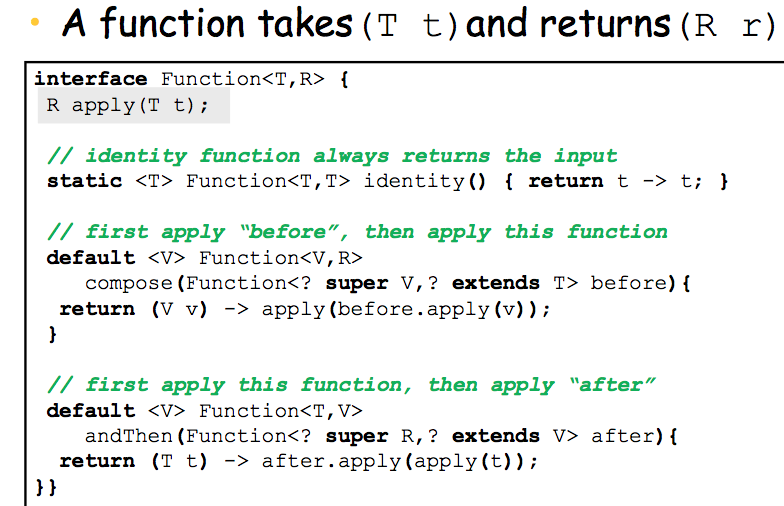


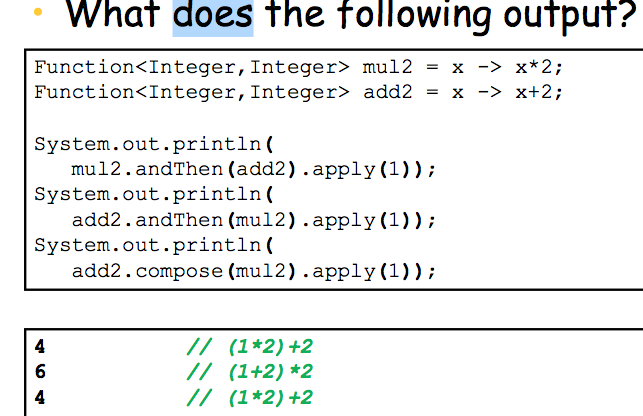


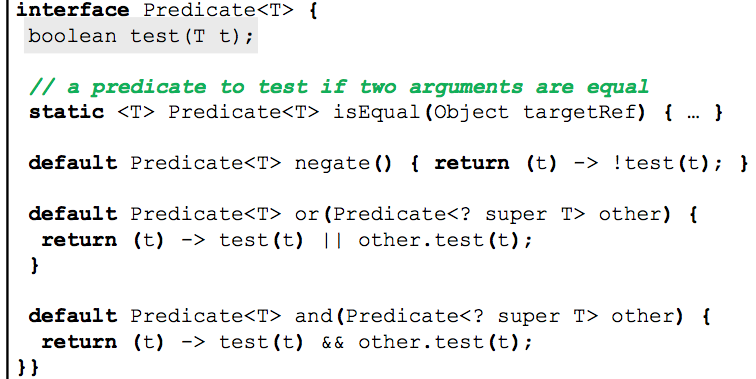


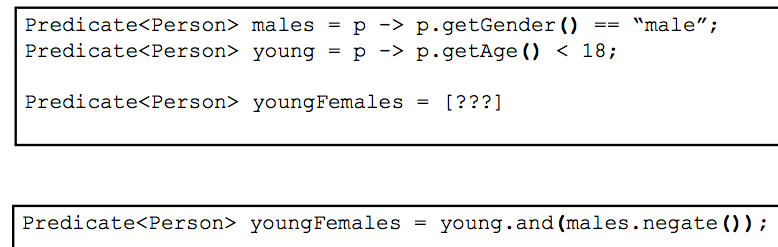


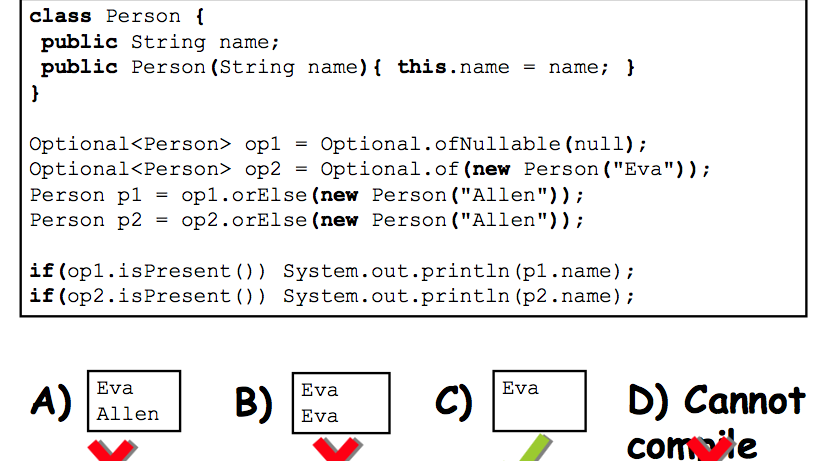


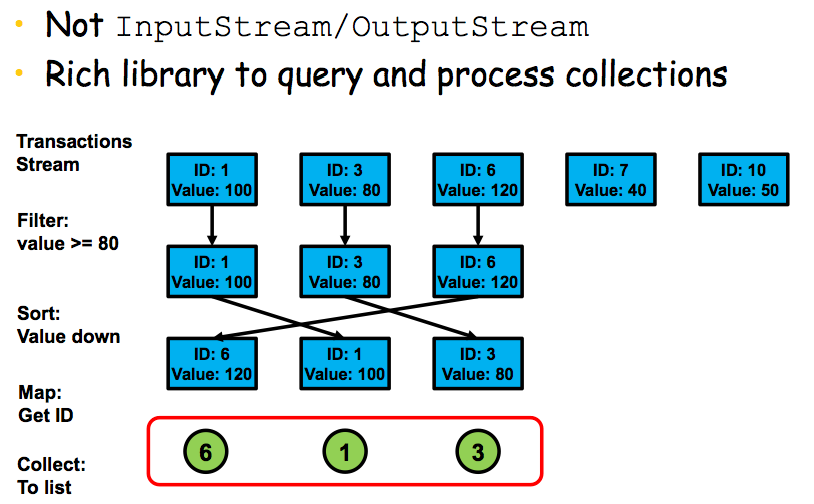


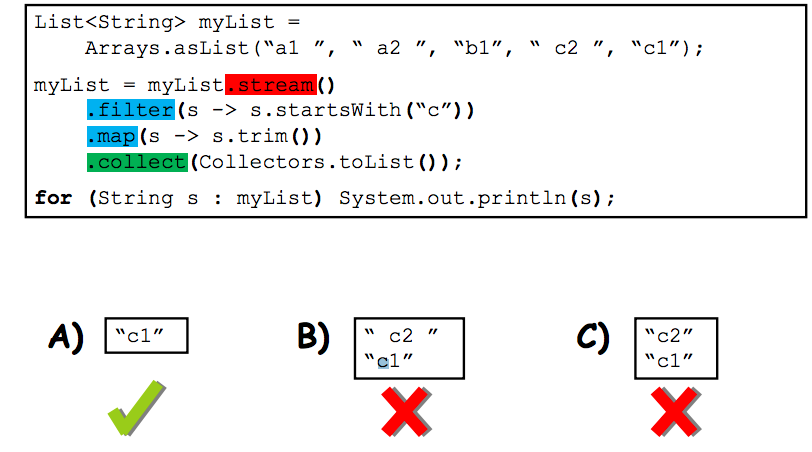


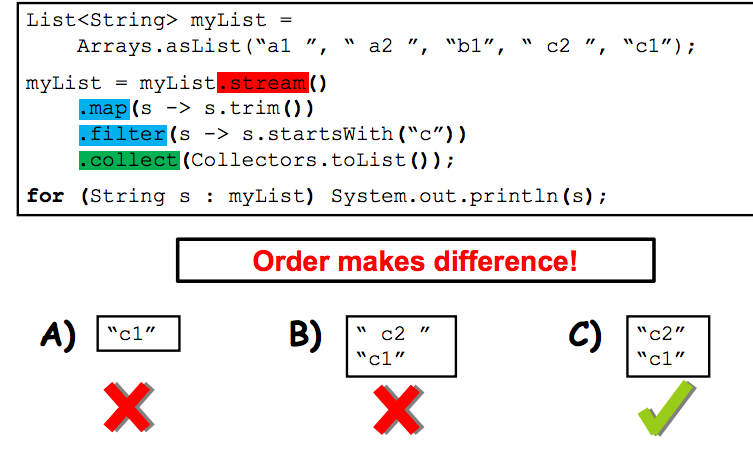












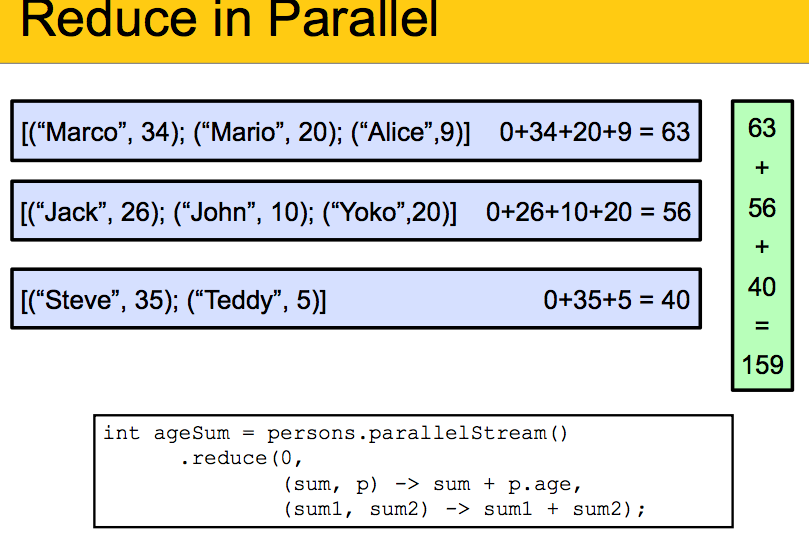
This is equivalent to:

T result = identity;

for (T element : this stream)

result = accumulator.apply(result, element)

return result;



优先级由高到低依次为：this.show(O)、super.show(O)、this.show((super)O)、super.show((super)O)

比如④，a2.show(b)，a2是一个引用变量，类型为A，则this为a2，b是B的一个实例，于是它到类A里面找show(B obj)方法，没有找到，于是到A的super(超类)找，而A没有超类，因此转到第三优先级this.show((super)O)，this仍然是a2，这里O为B，(super)O即(super)B即A，因此它到类A里面找show(A obj)的方法，类A有这个方法，但是由于a2引用的是类B的一个对象，B覆盖了A的show(A obj)方法，因此最终锁定到类B的show(A obj)，输出为"B and A”。

（一）相关类

1. **class** A ...{
2. **public** String show(D obj)...{
3. **return** ("A and D");
4. }
5. **public** String show(A obj)...{
6. **return** ("A and A");
7. }
8. }
9. **class** B **extends** A...{
10. **public** String show(B obj)...{
11. **return** ("B and B");
12. }
13. **public** String show(A obj)...{
14. **return** ("B and A");
15. }
16. }
17. **class** C **extends** B...{}
18. **class** D **extends** B...{}

（二）问题：以下输出结果是什么？

1. A a1 = **new** A();
2. A a2 = **new** B();
3. B b = **new** B();
4. C c = **new** C();
5. D d = **new** D();
6. System.out.println(a1.show(b));   ①
7. System.out.println(a1.show(c));   ②
8. System.out.println(a1.show(d));   ③
9. System.out.println(a2.show(b));   ④
10. System.out.println(a2.show(c));   ⑤
11. System.out.println(a2.show(d));   ⑥
12. System.out.println(b.show(b));    ⑦
13. System.out.println(b.show(c));    ⑧
14. System.out.println(b.show(d));    ⑨

（三）答案

①   A and A

②   A and A

③   A and D

④   B and A

⑤   B and A

⑥   A and D

⑦   B and B

⑧   B and B

⑨   A and D