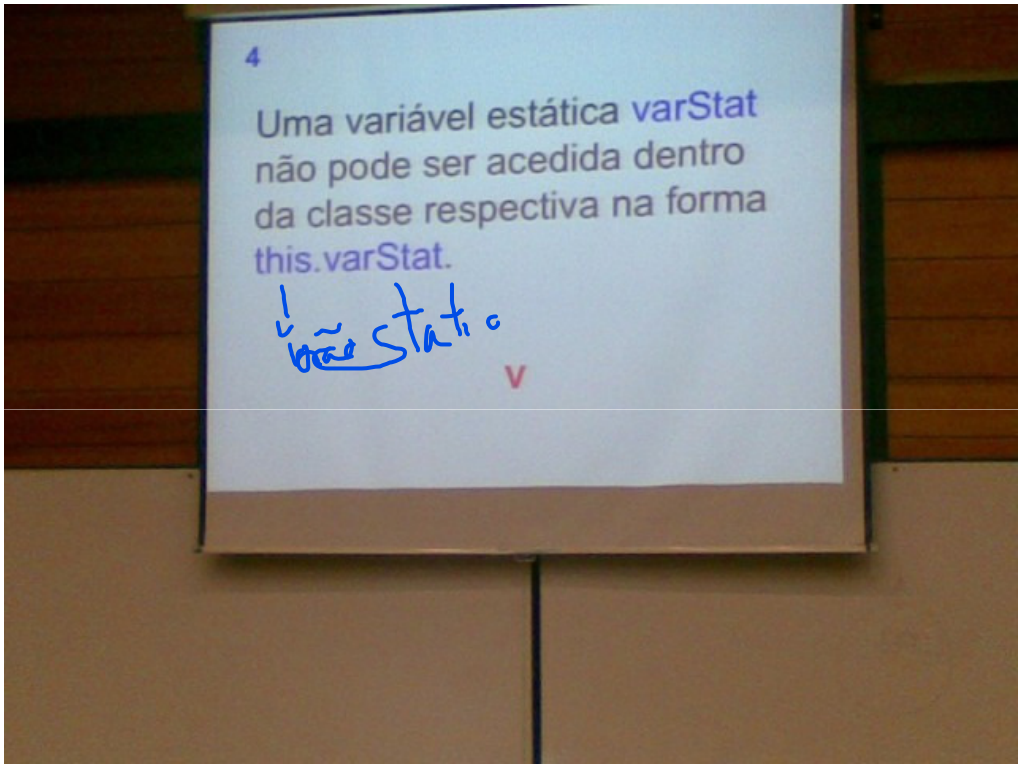
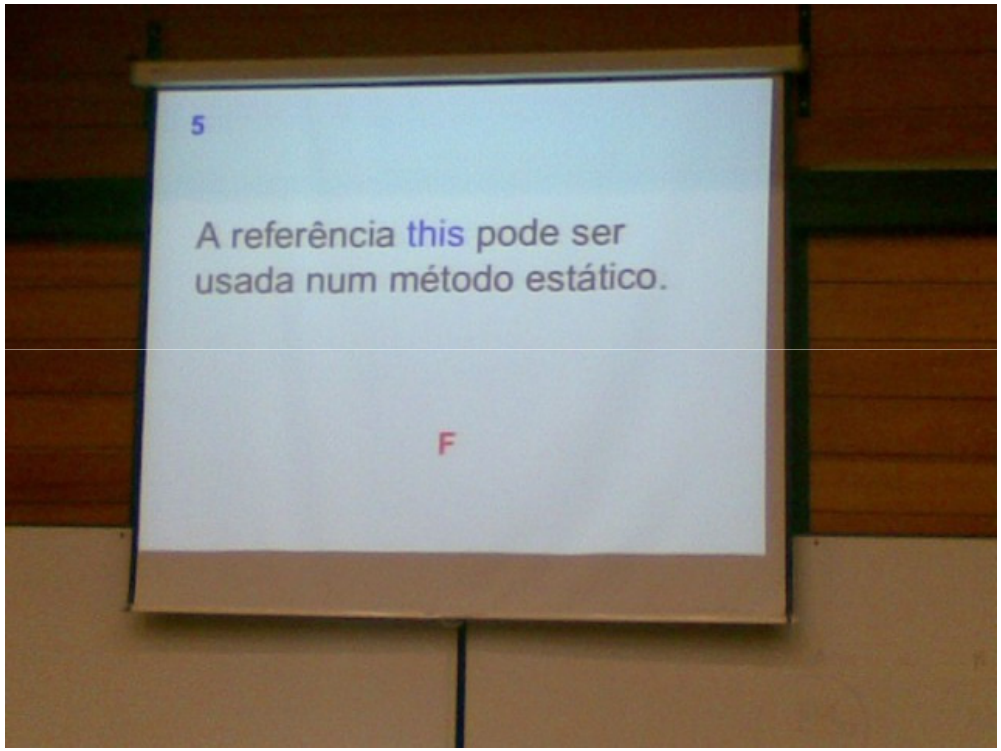
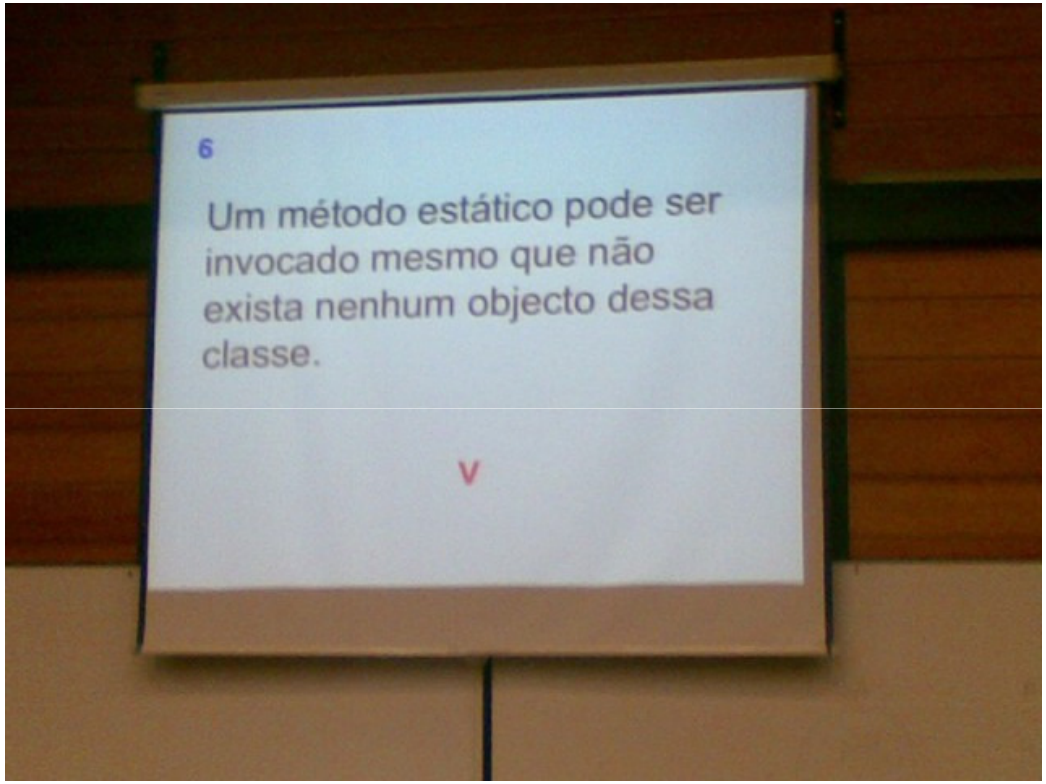
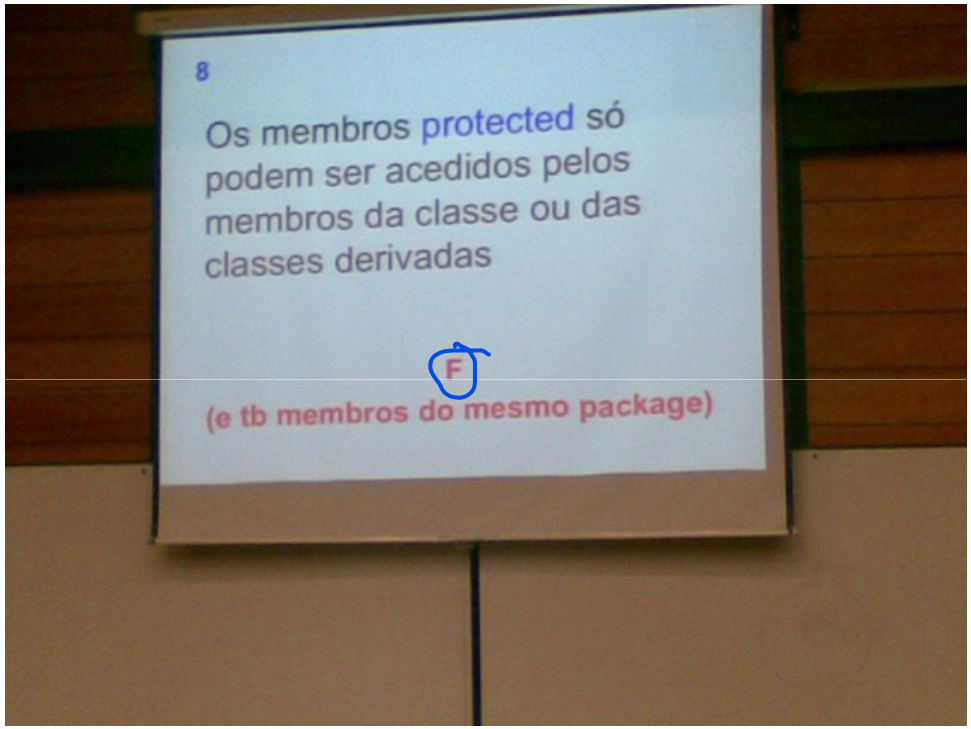
V/F’s
17/12/2008
JLO

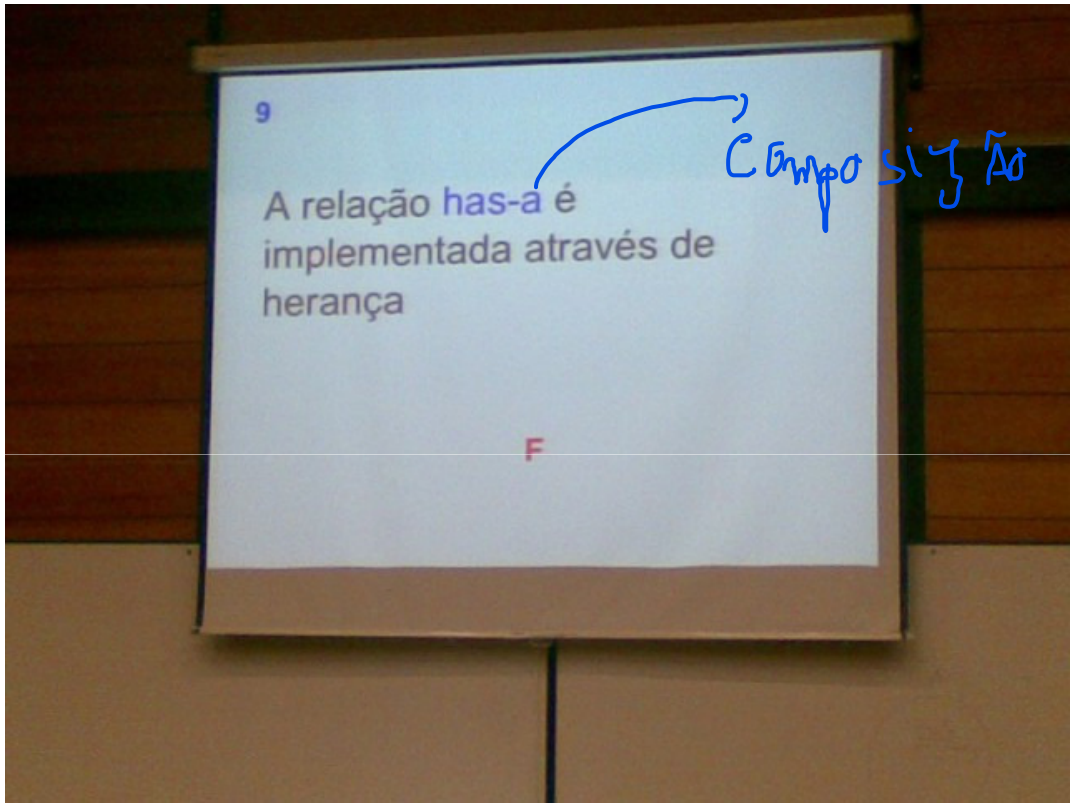

4
Uma variável estática varStat não pode ser acedida dentro da classe respectiva na forma this.varStat.
não static
V

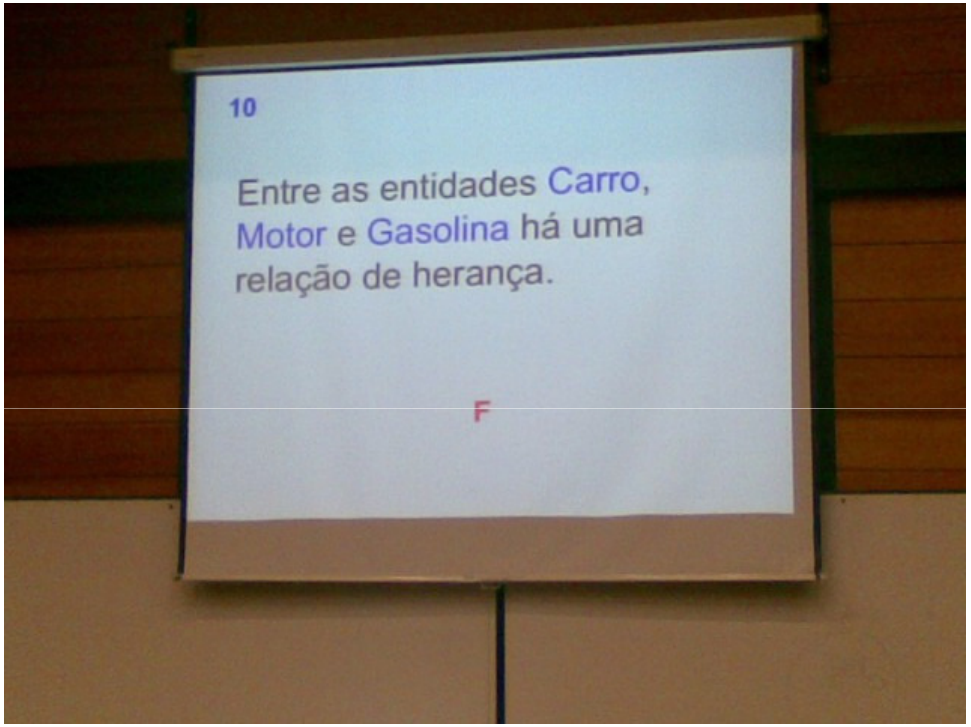

5
A referência this pode ser usada num método estático.
F

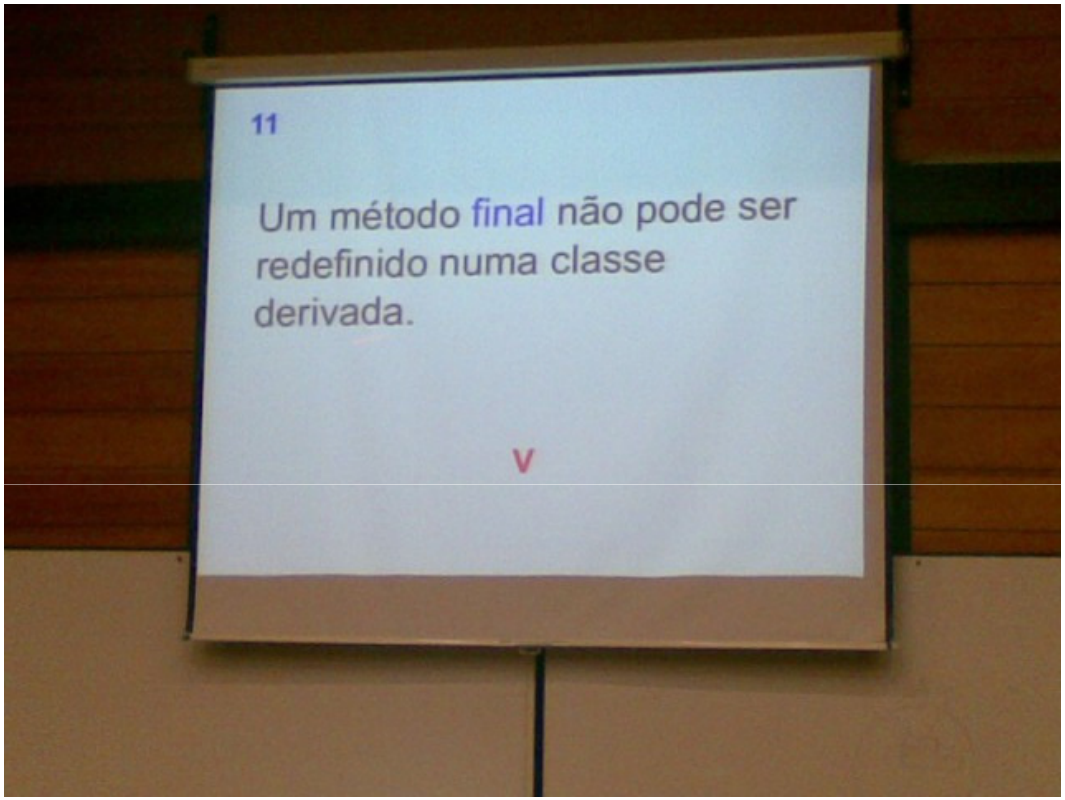

6
Um método estático pode ser invocado mesmo que não exista nenhum objecto dessa classe.
V

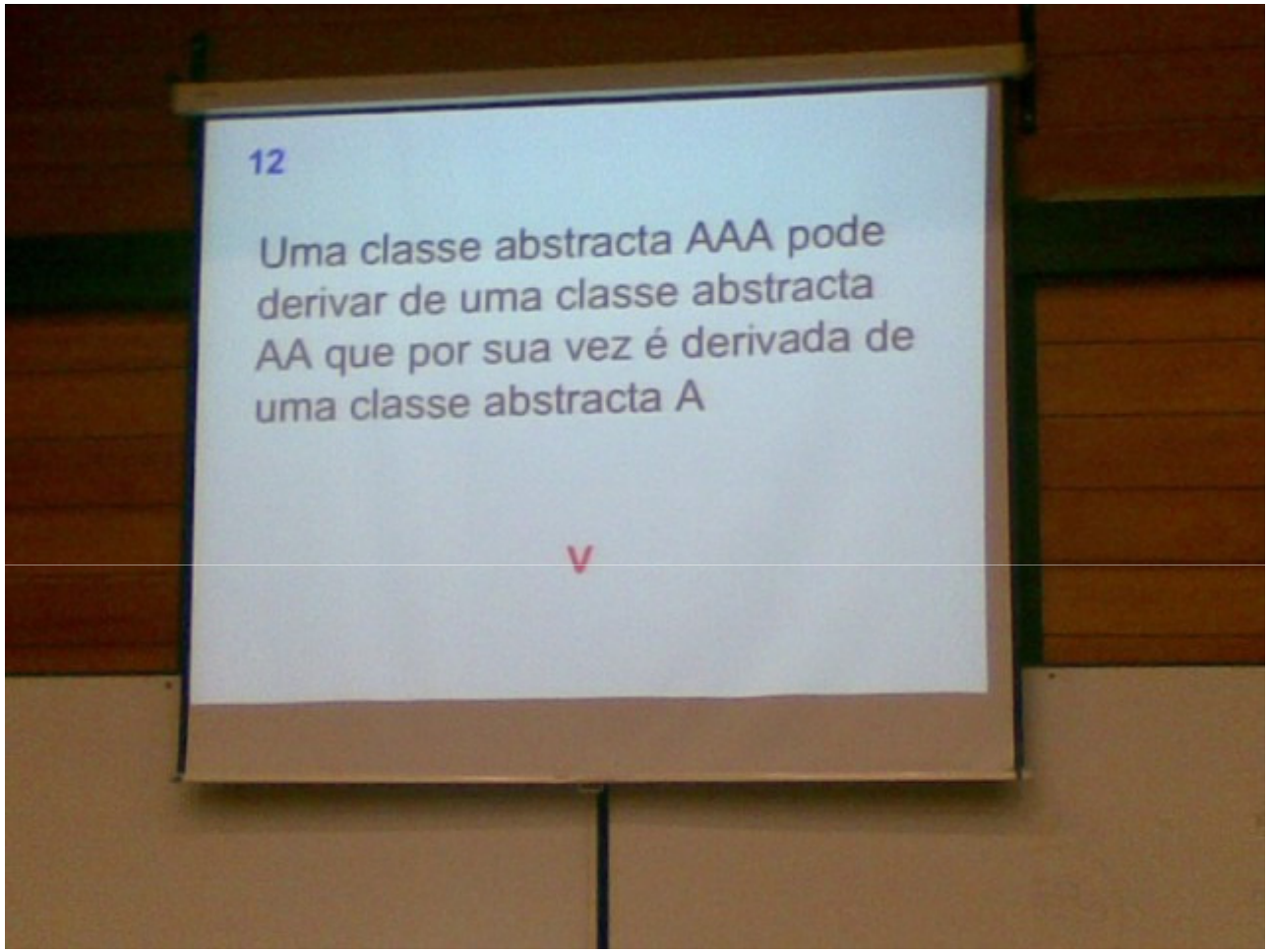



protected
é
a classe e classes extendidas e membros do
PACOTE

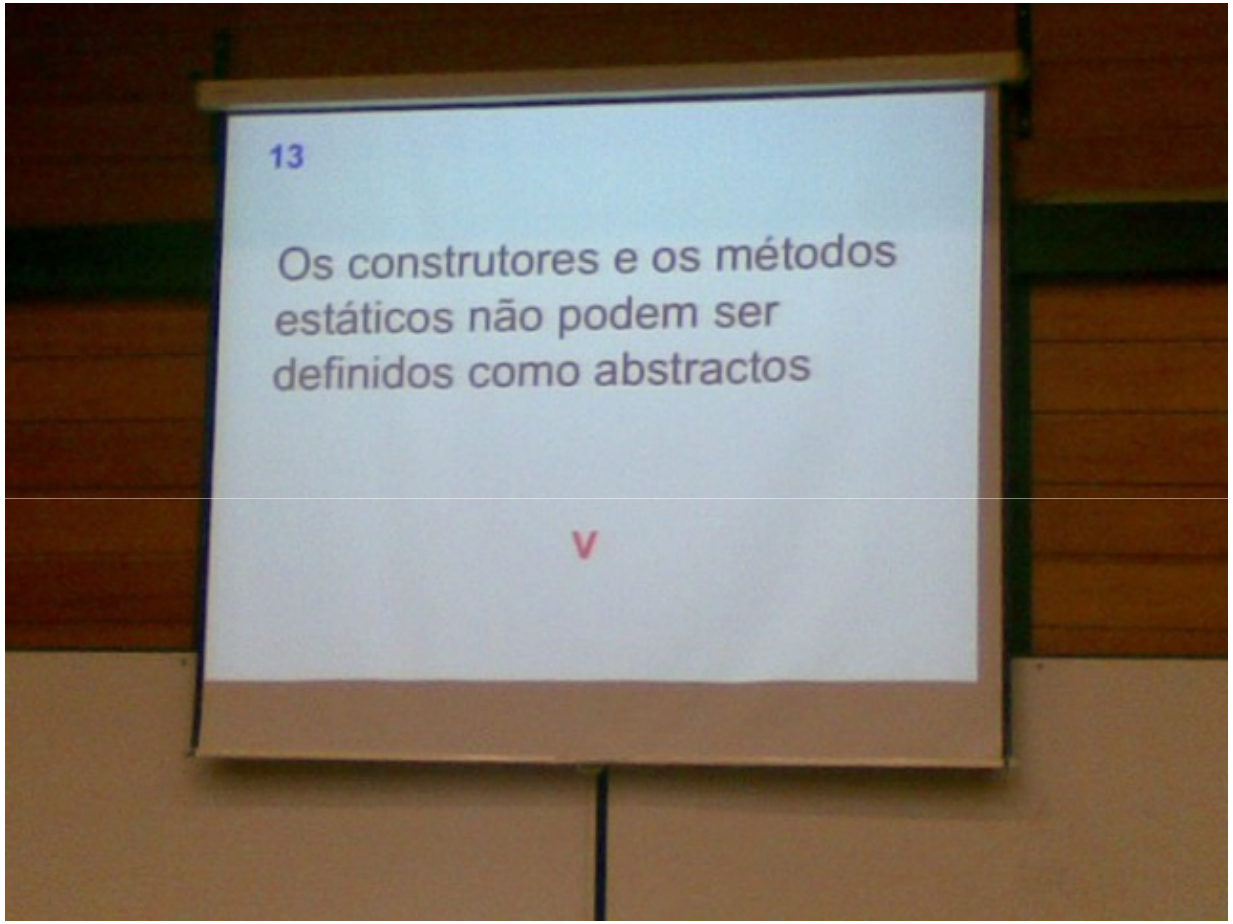

9
A relação has-a é implementada através de herança
F
Composição

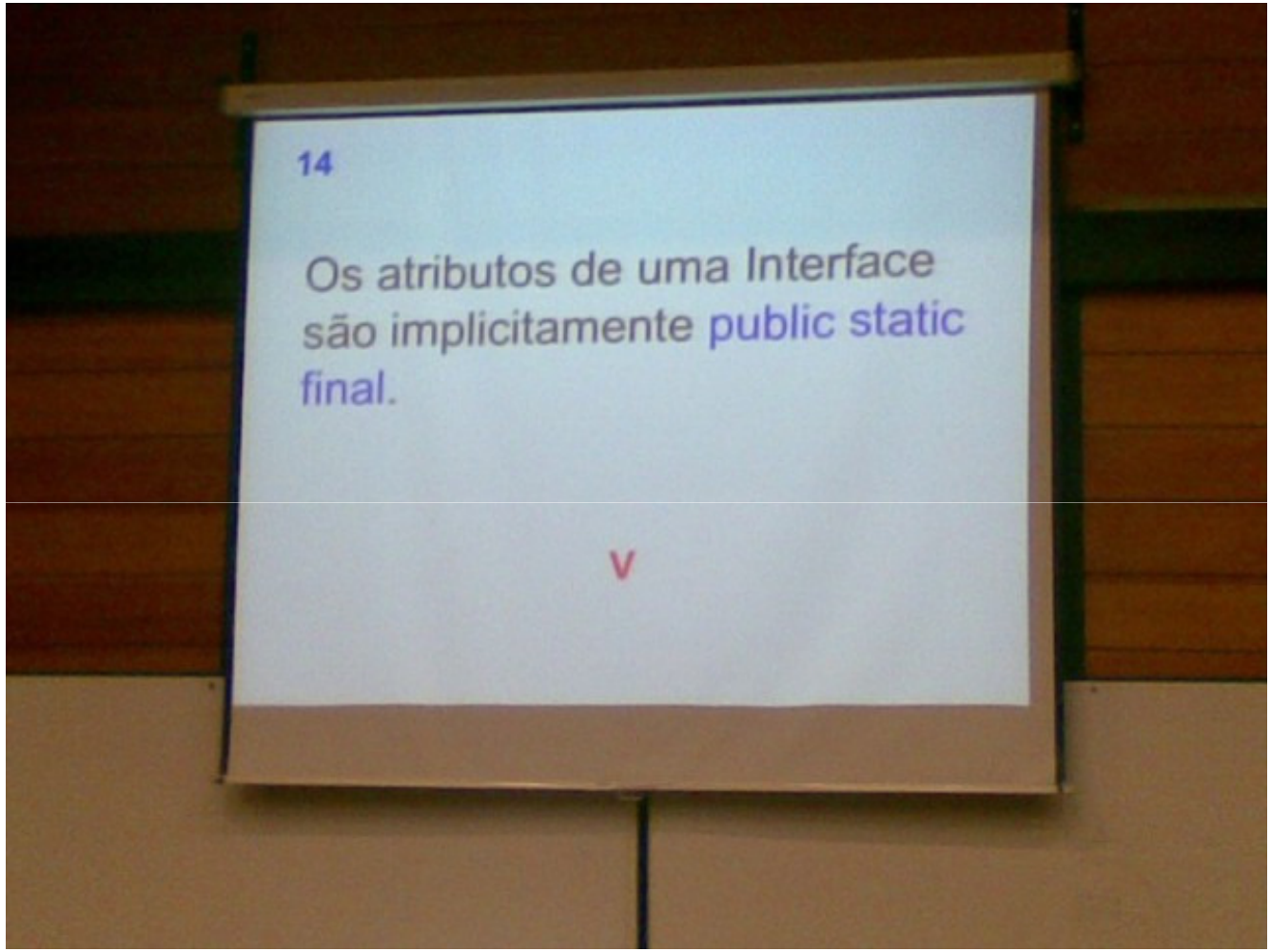

10
Entre as entidades Carro, Motor e Gasolina há uma relação de herança.
F

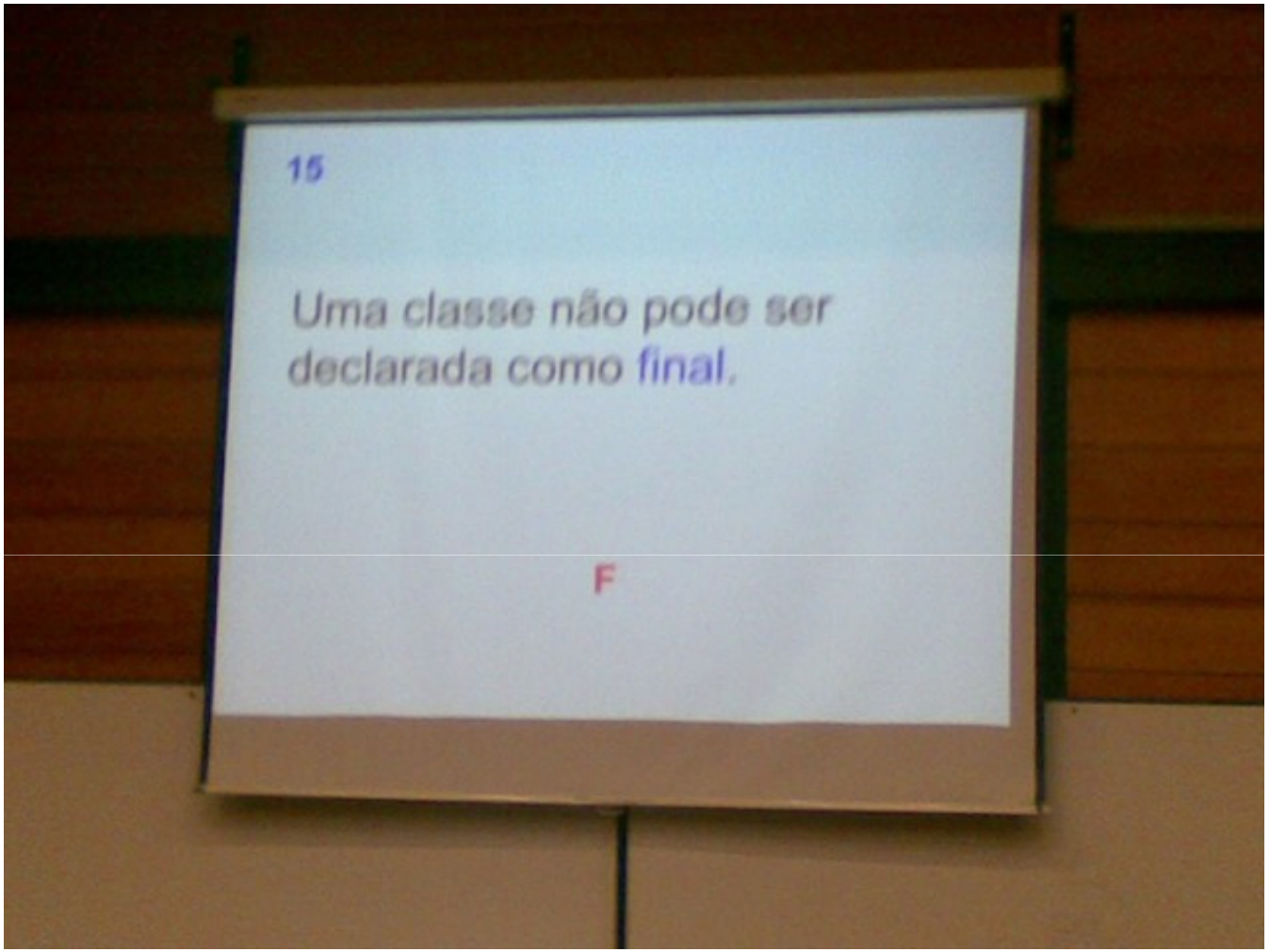

11
Um método final não pode ser redefinido numa classe derivada.
V



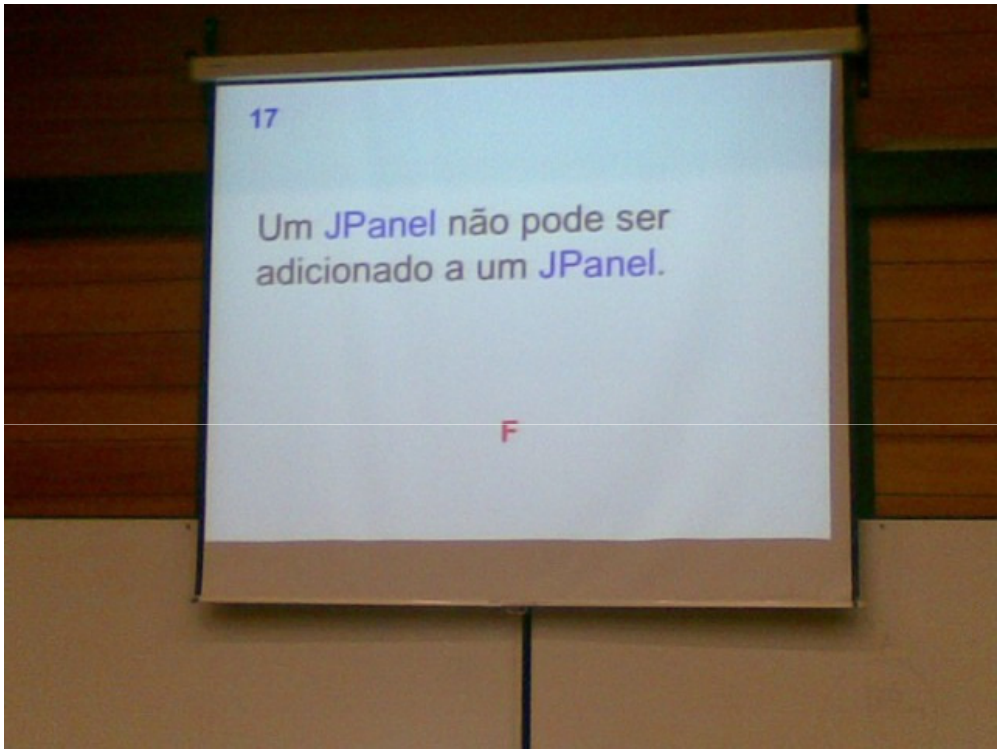

abs podem derivar de abs

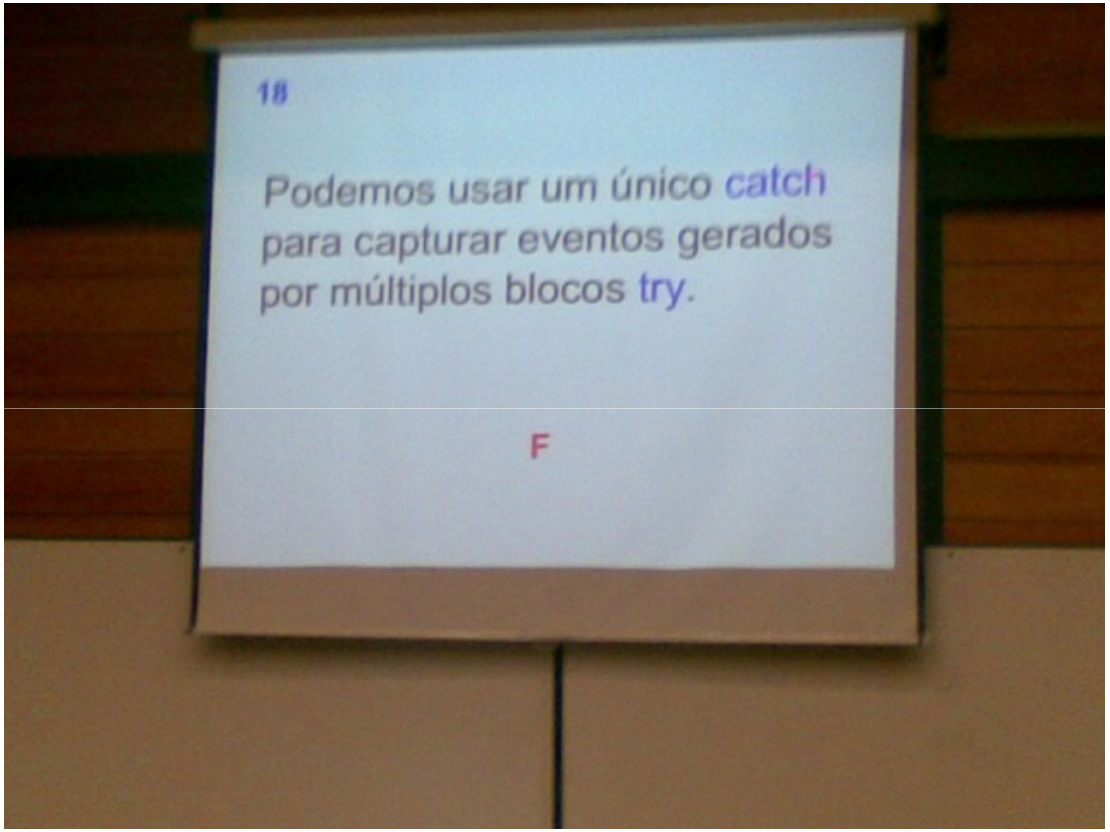

13
Os construtores e os métodos estáticos não podem ser definidos como abstractos
V


14
Os atributos de uma Interface são implicitamente public static final.
V


15
Uma classe não pode ser declarada como final.
F


16
Um JPanel é um JComponent.


17
Um JPanel não pode ser adicionado a um JPanel.
F

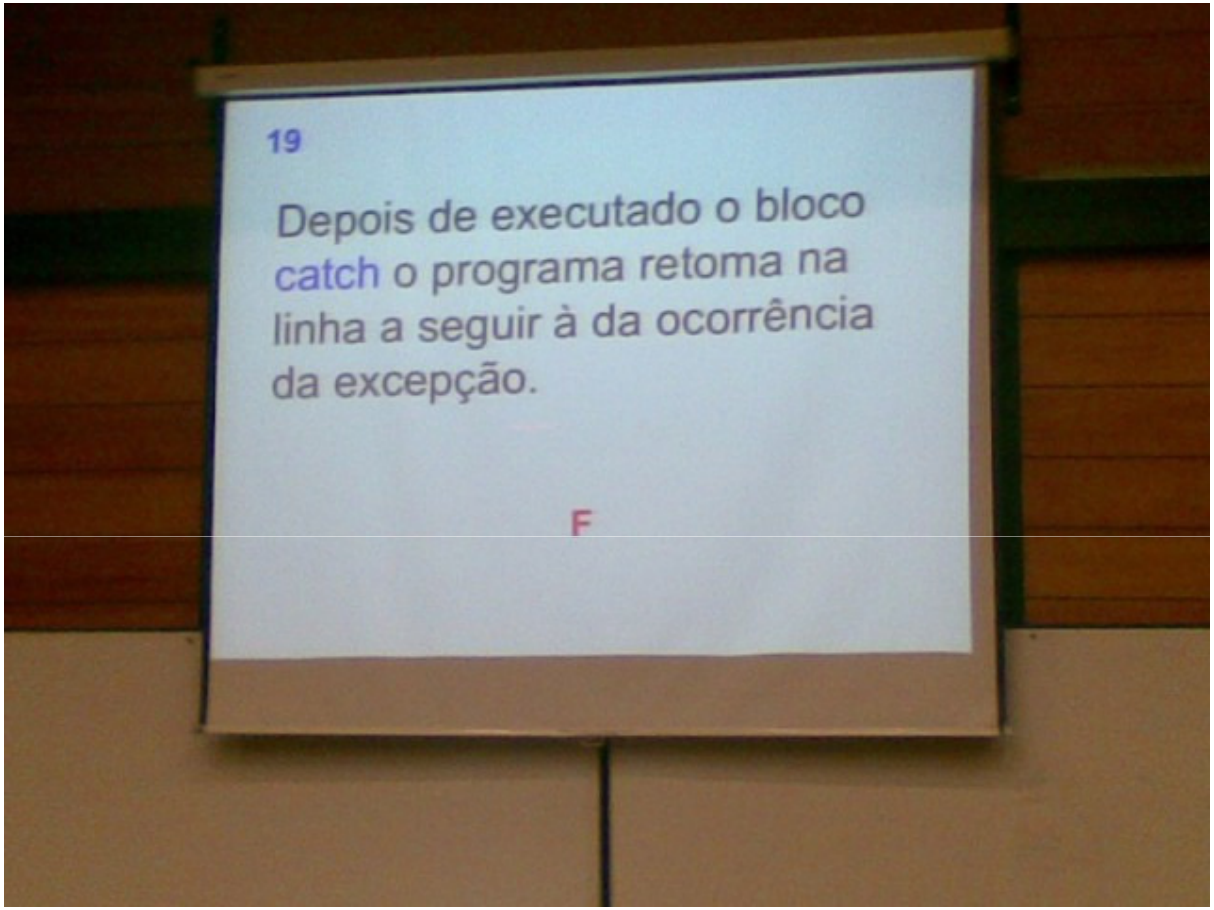


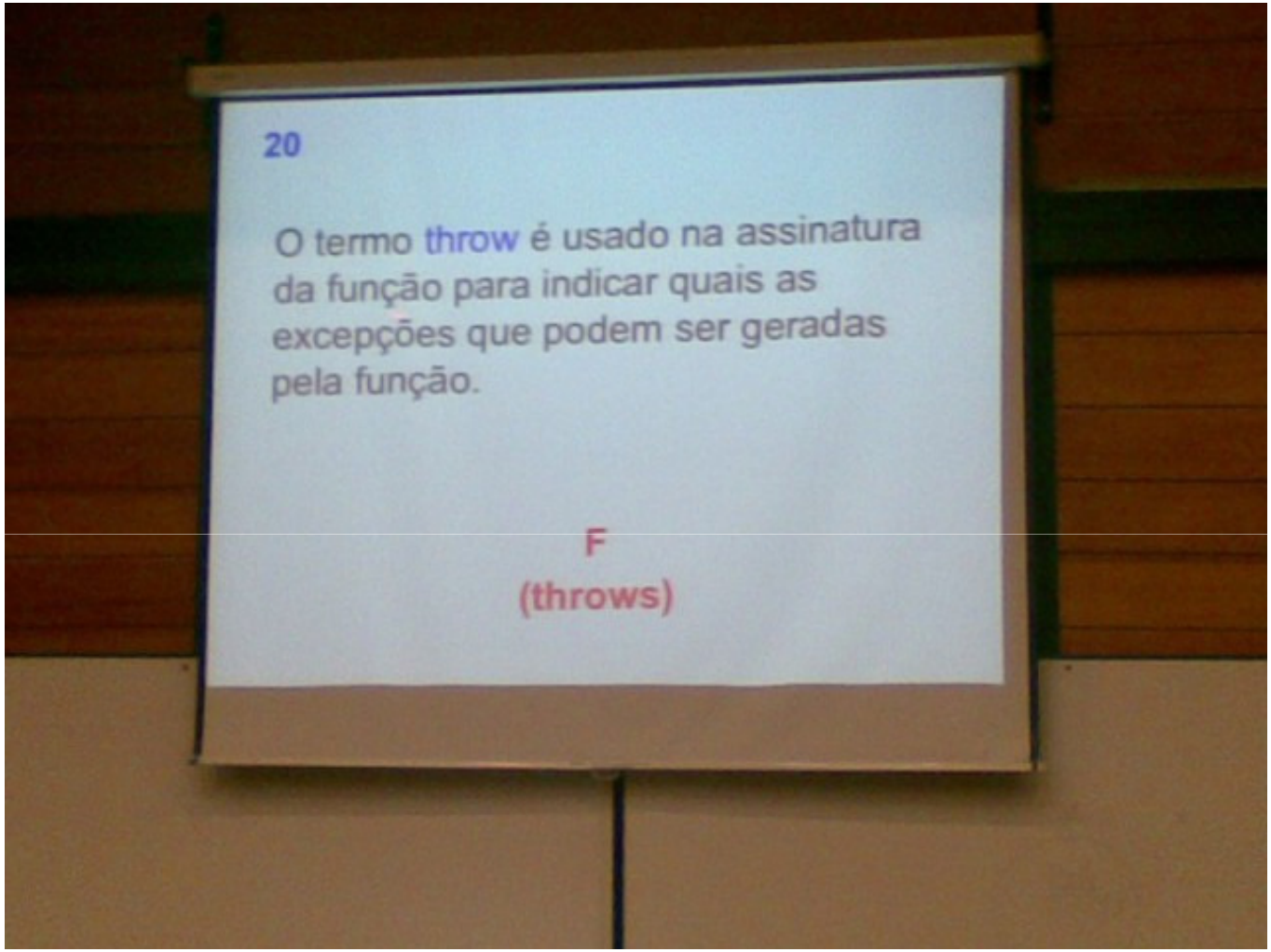

try -> vários catches

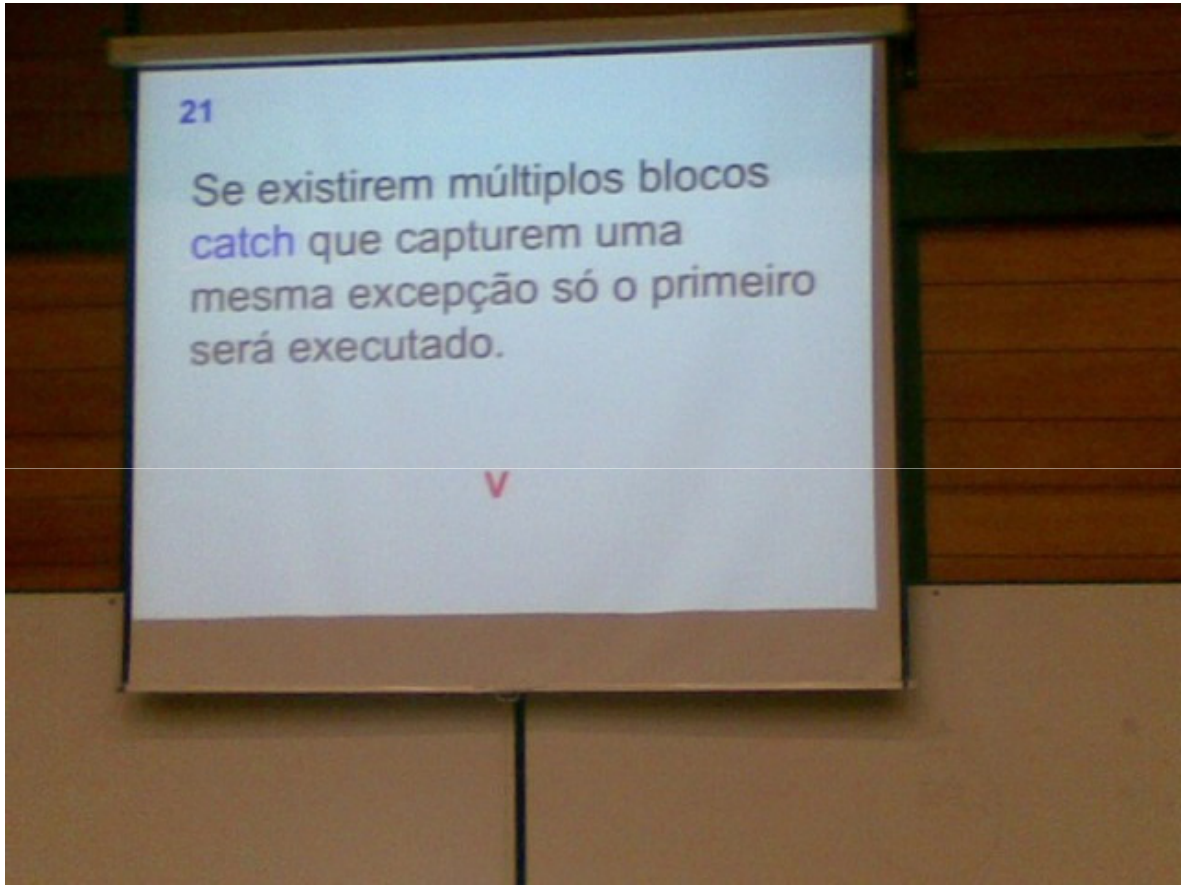
catch }

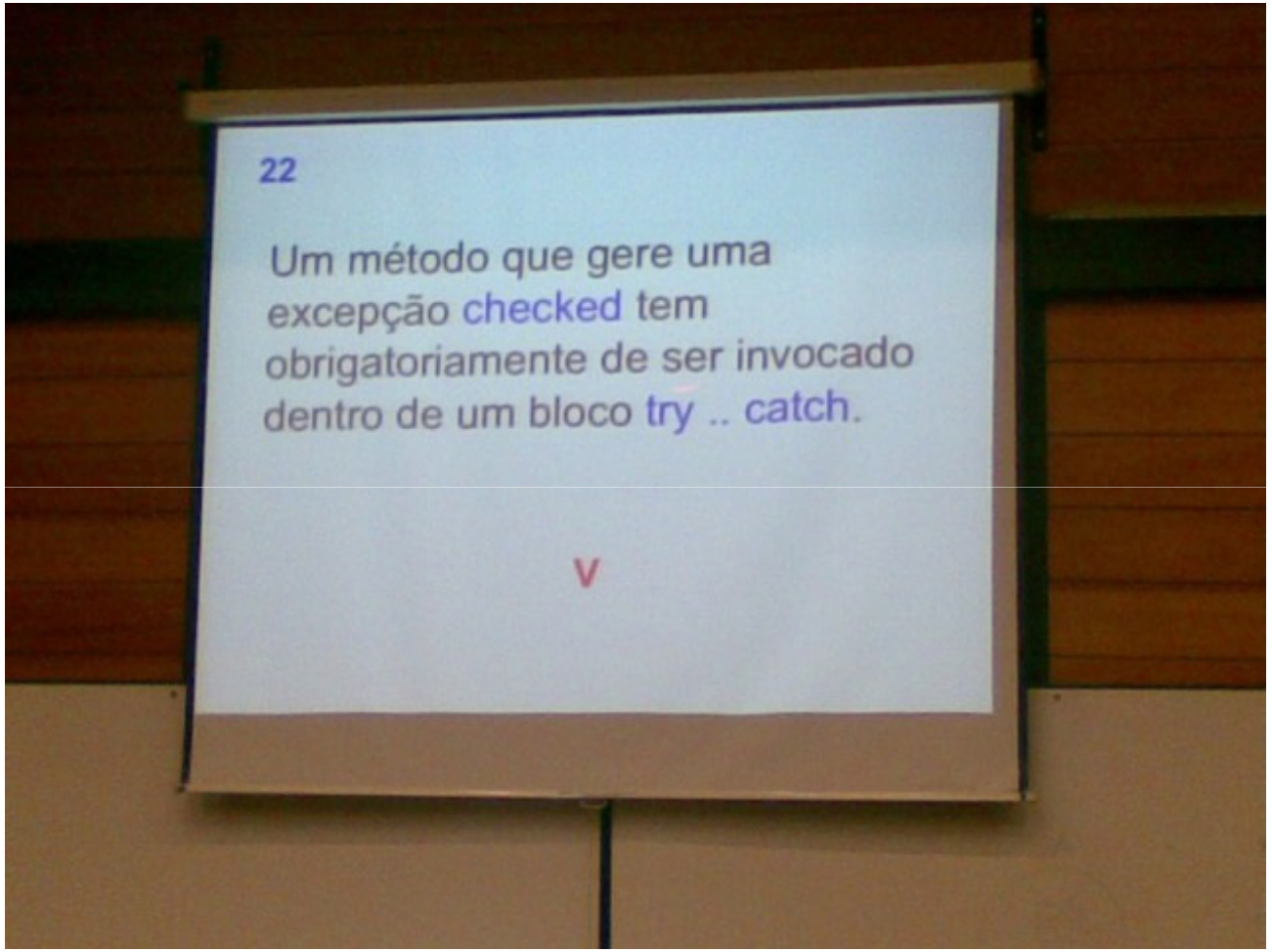
só

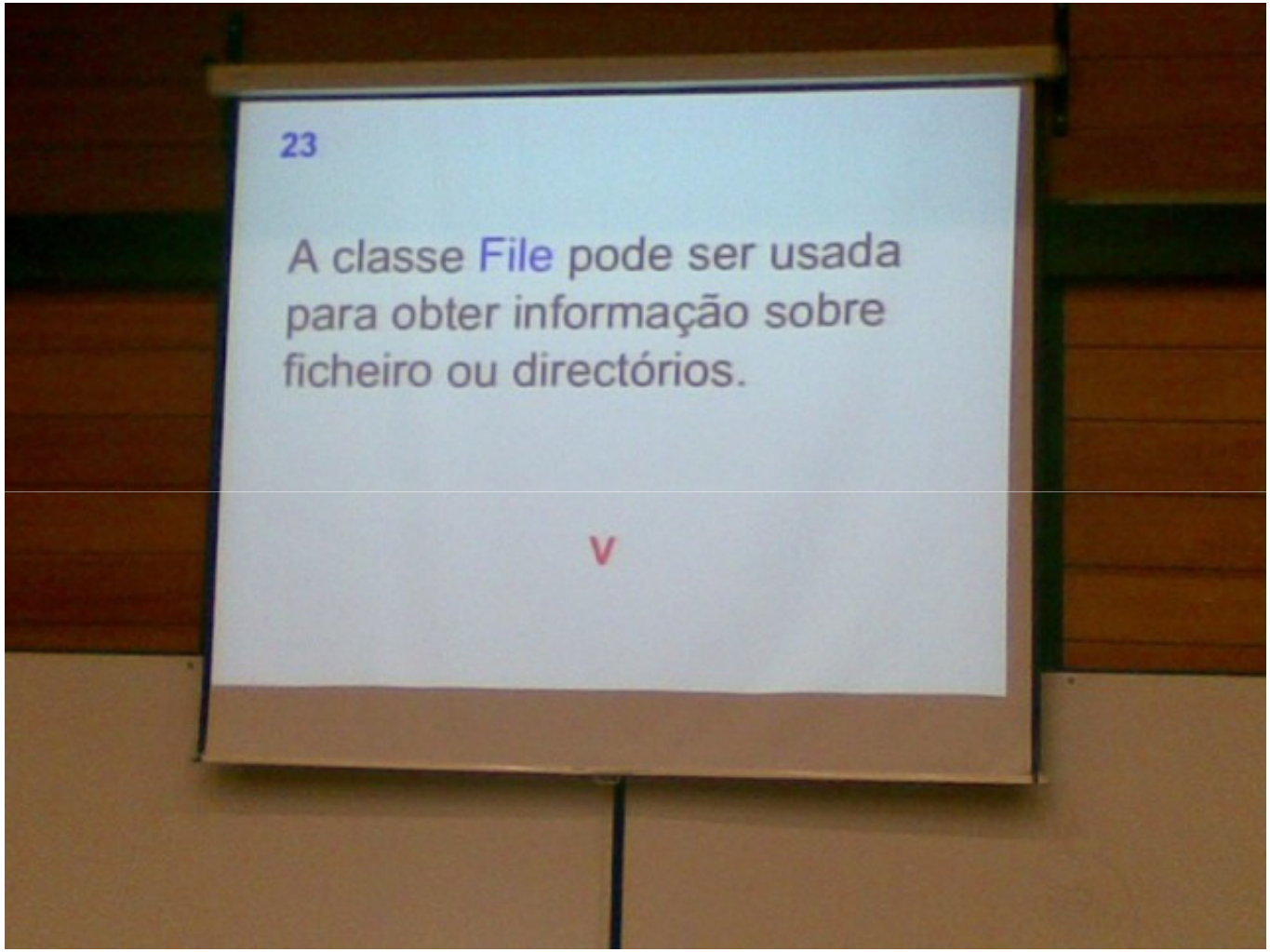
1 try

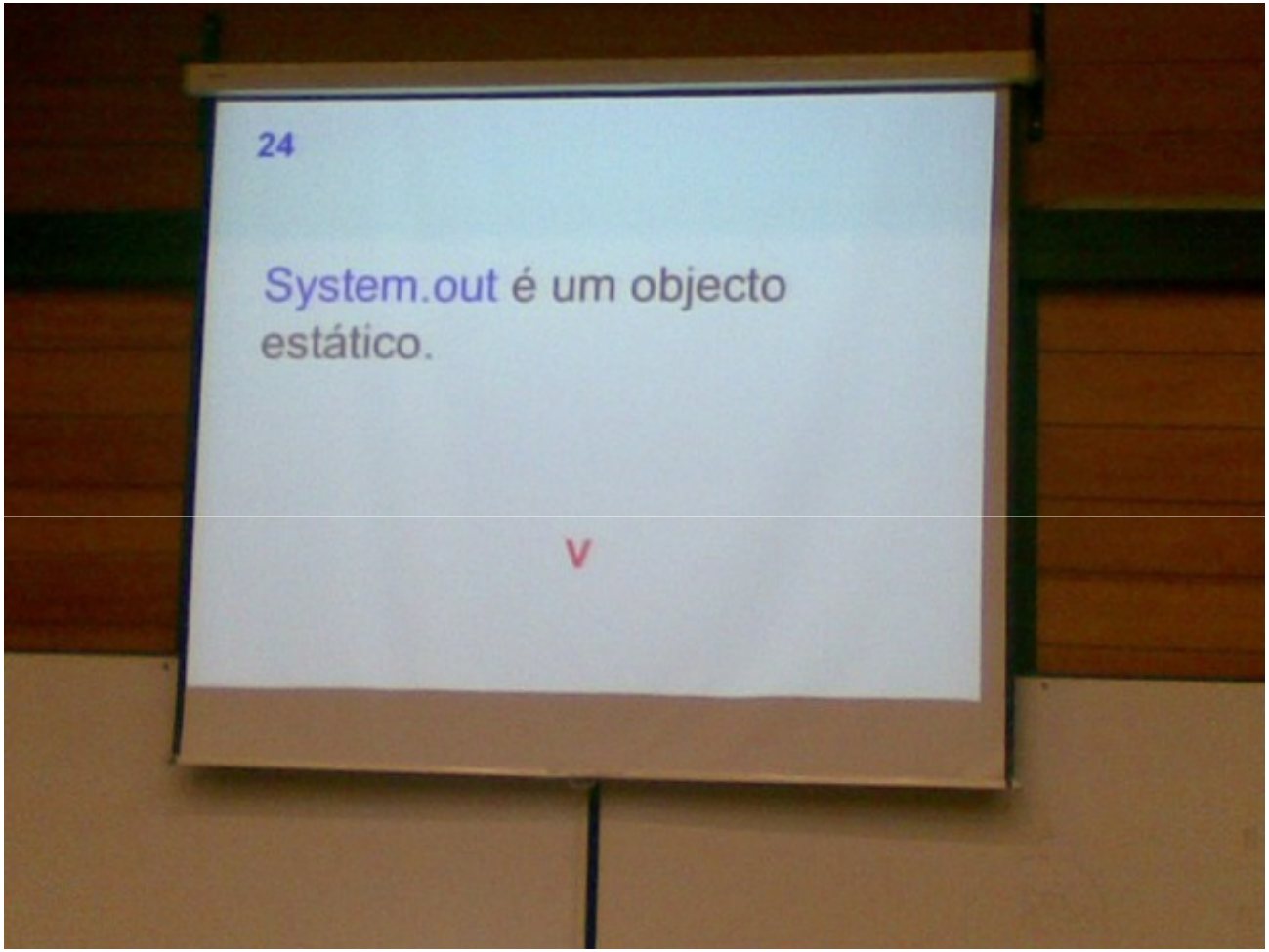

19
Depois de executado o bloco catch o programa retoma na linha a seguir à da ocorrência da excepção.
F

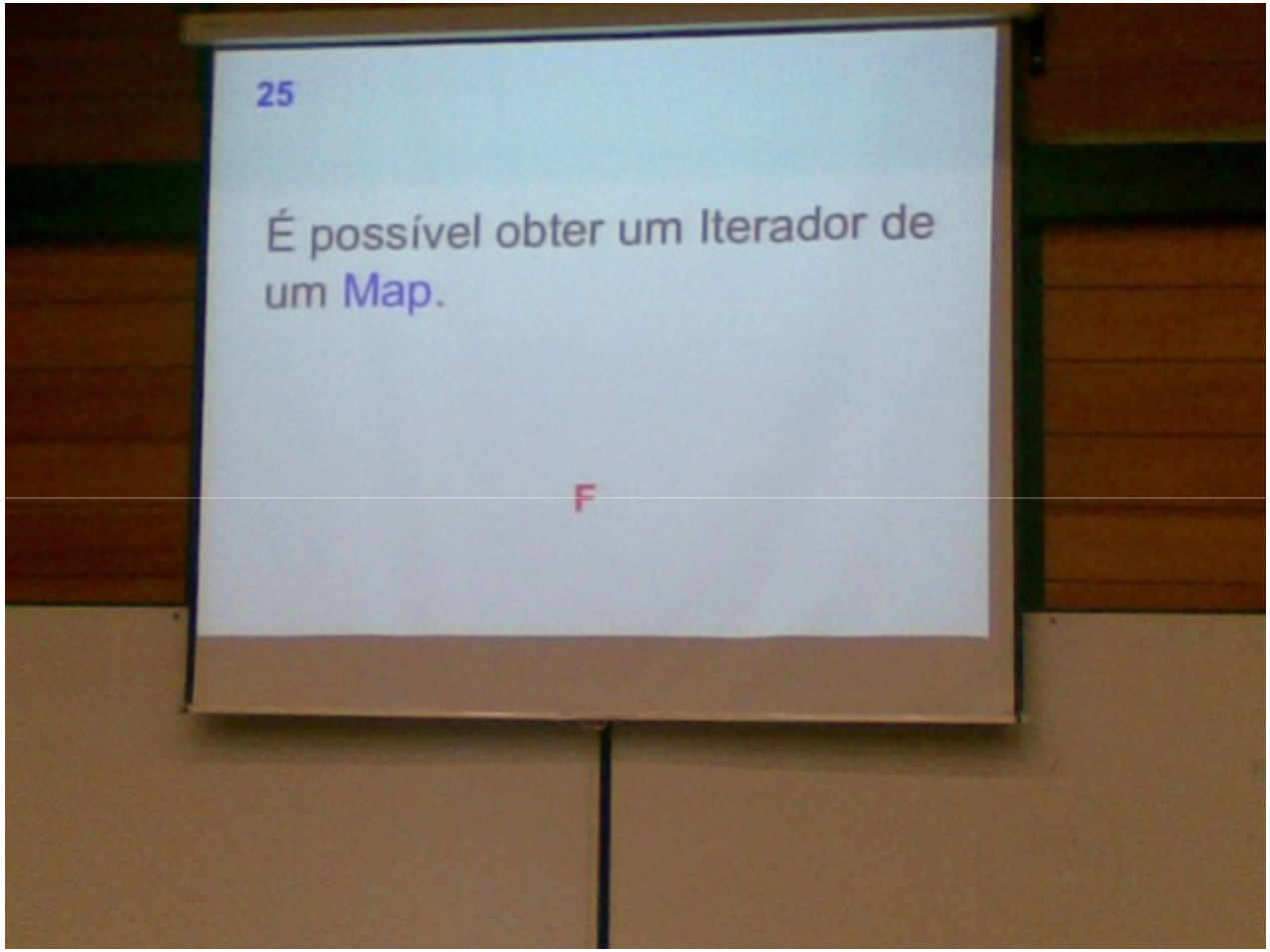

20
O termo throw é usado na assinatura da função para indicar quais as excepções que podem ser geradas pela função.
F
(throws)

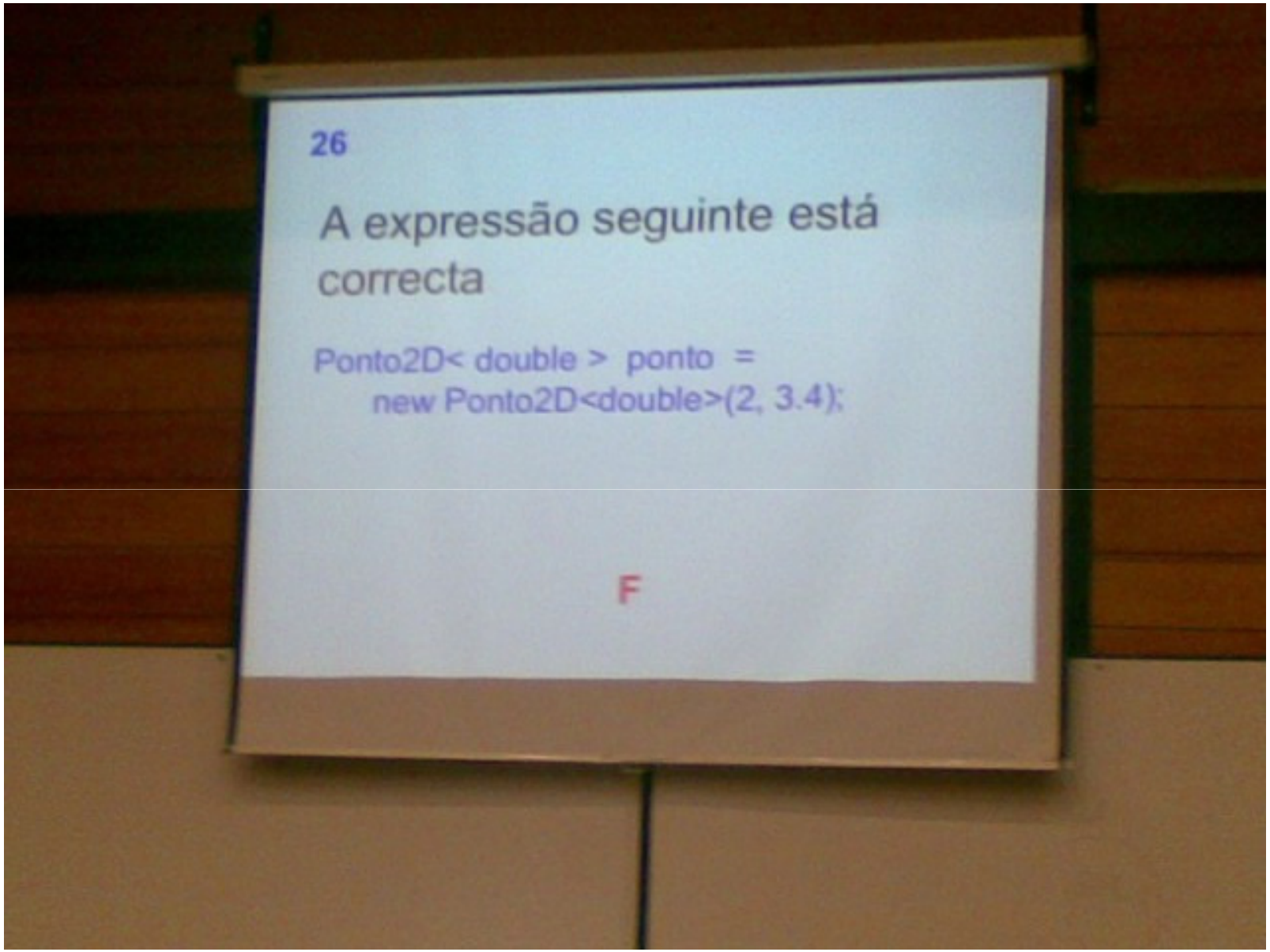

21
Se existirem múltiplos blocos catch que capturem uma mesma excepção só o primeiro será executado.
V

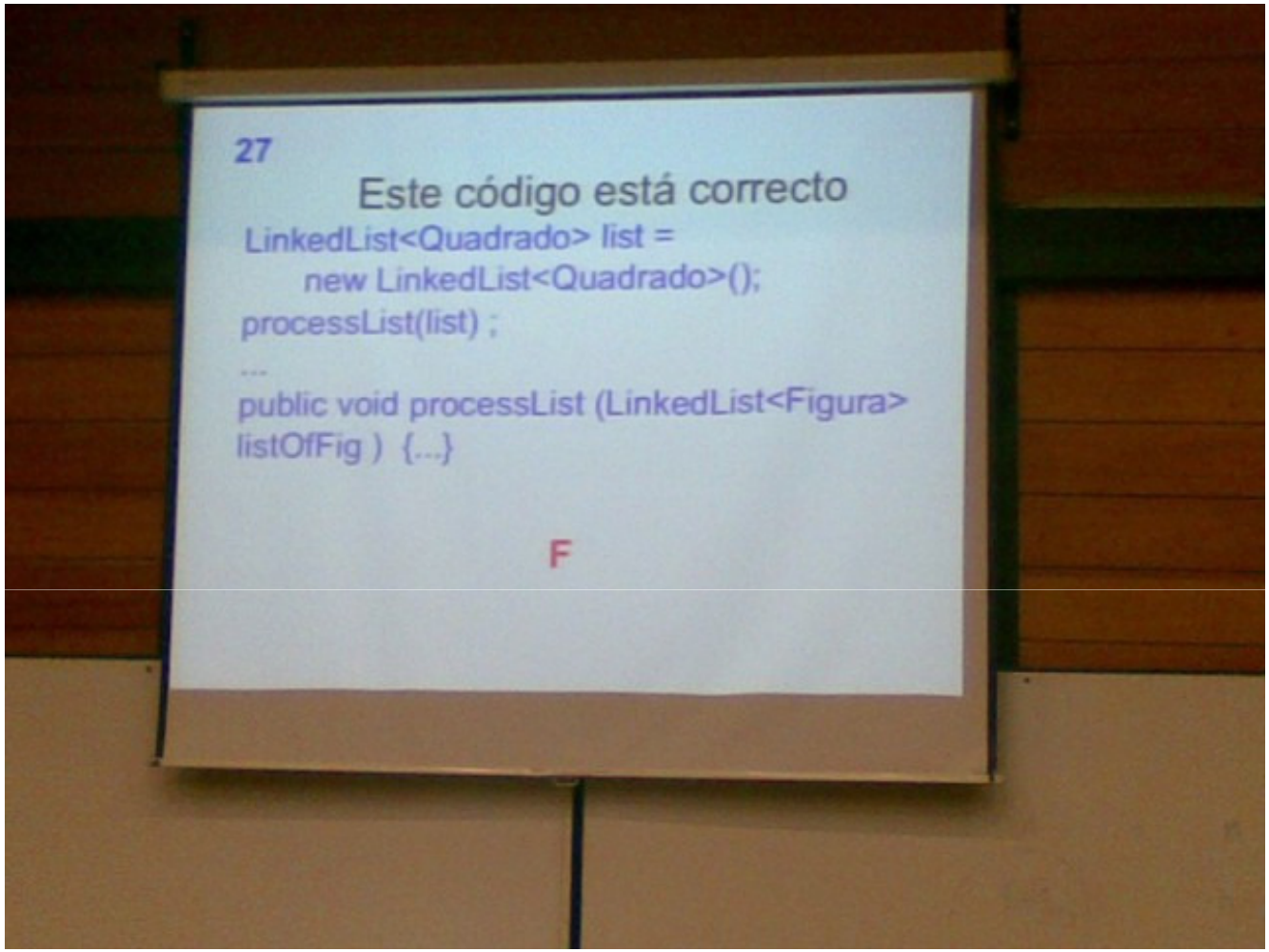

22
Um método que gere uma excepção checked tem obrigatoriamente de ser invocado dentro de um bloco try .. catch.
V

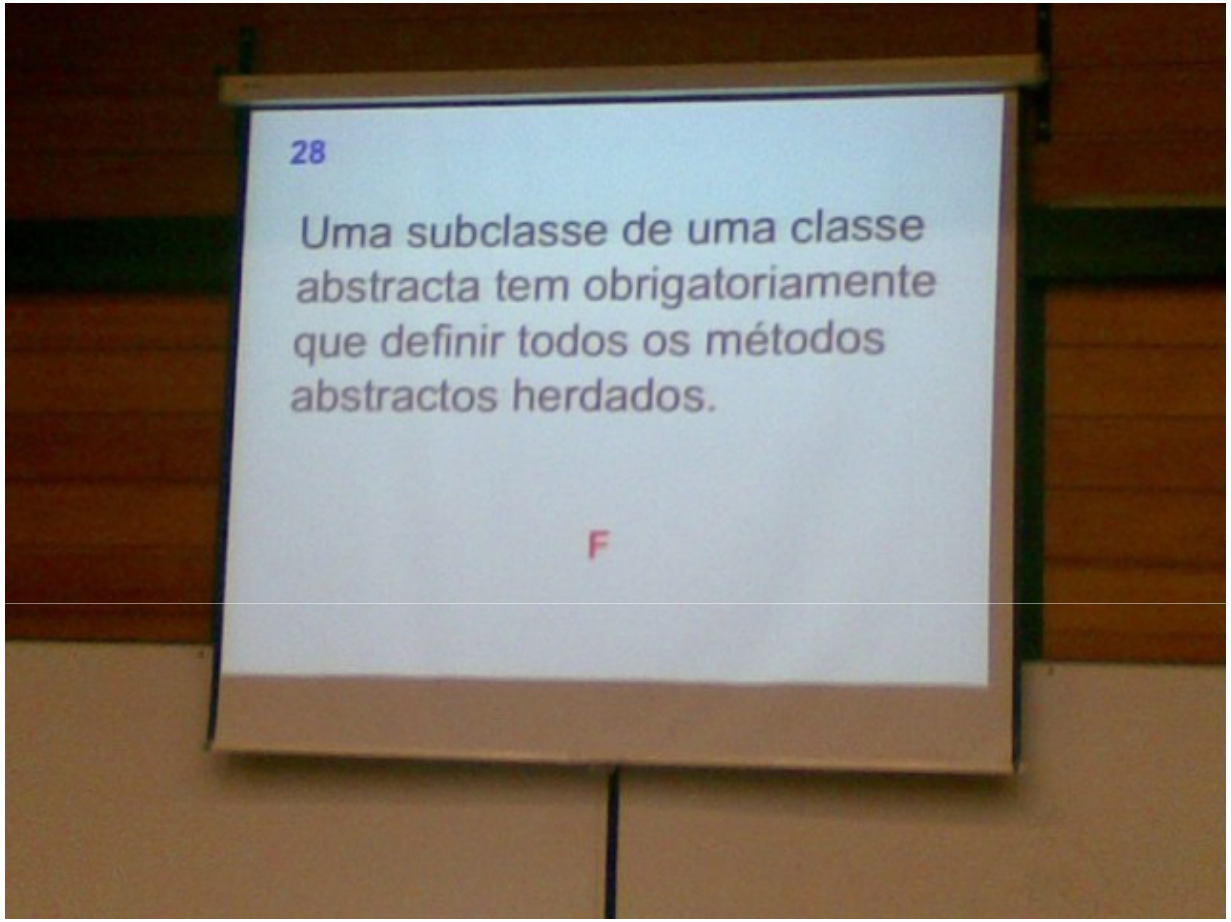

23
A classe File pode ser usada para obter informação sobre ficheiro ou directórios.
V

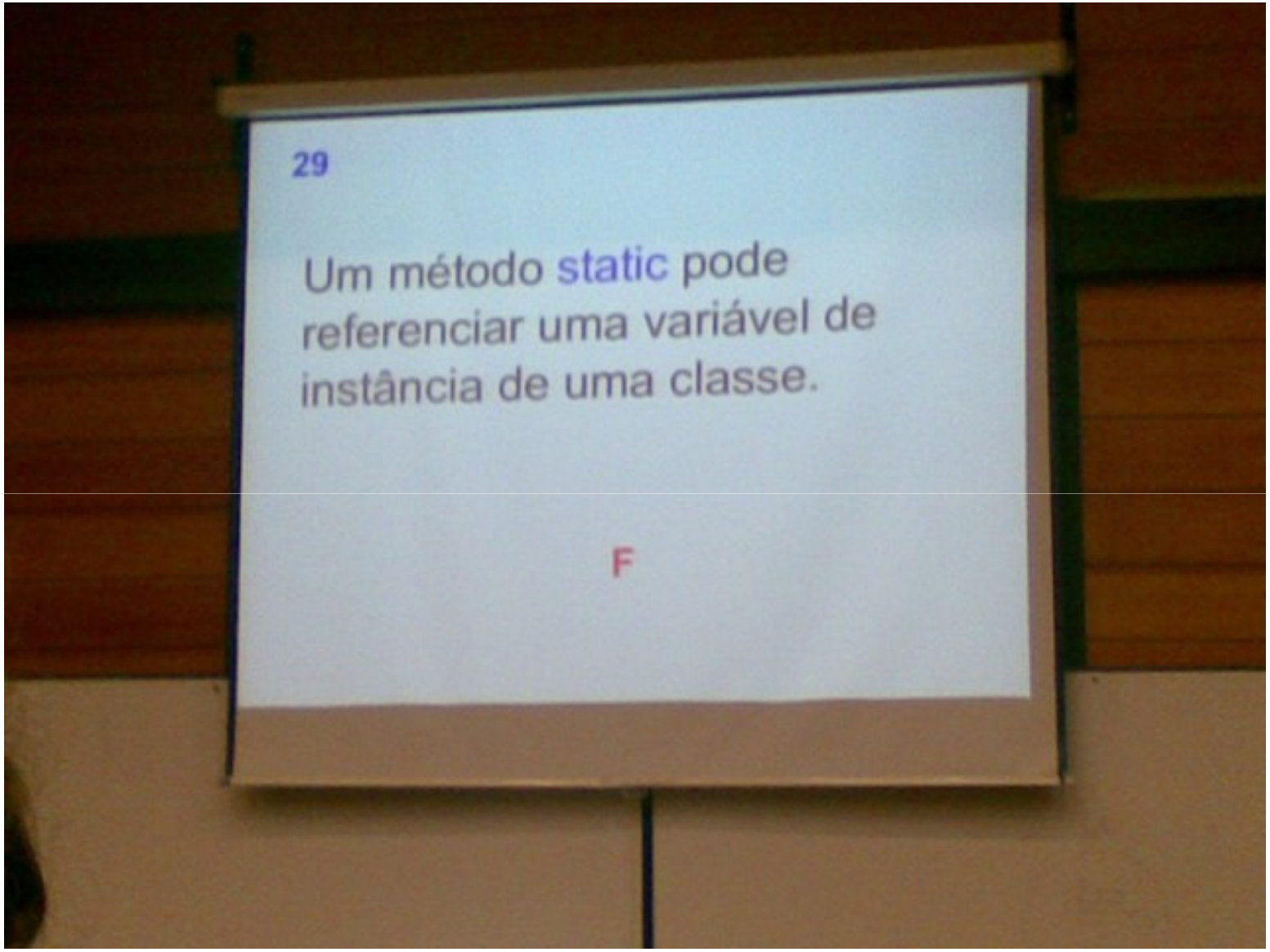

24
System.out é um objecto estático.
V

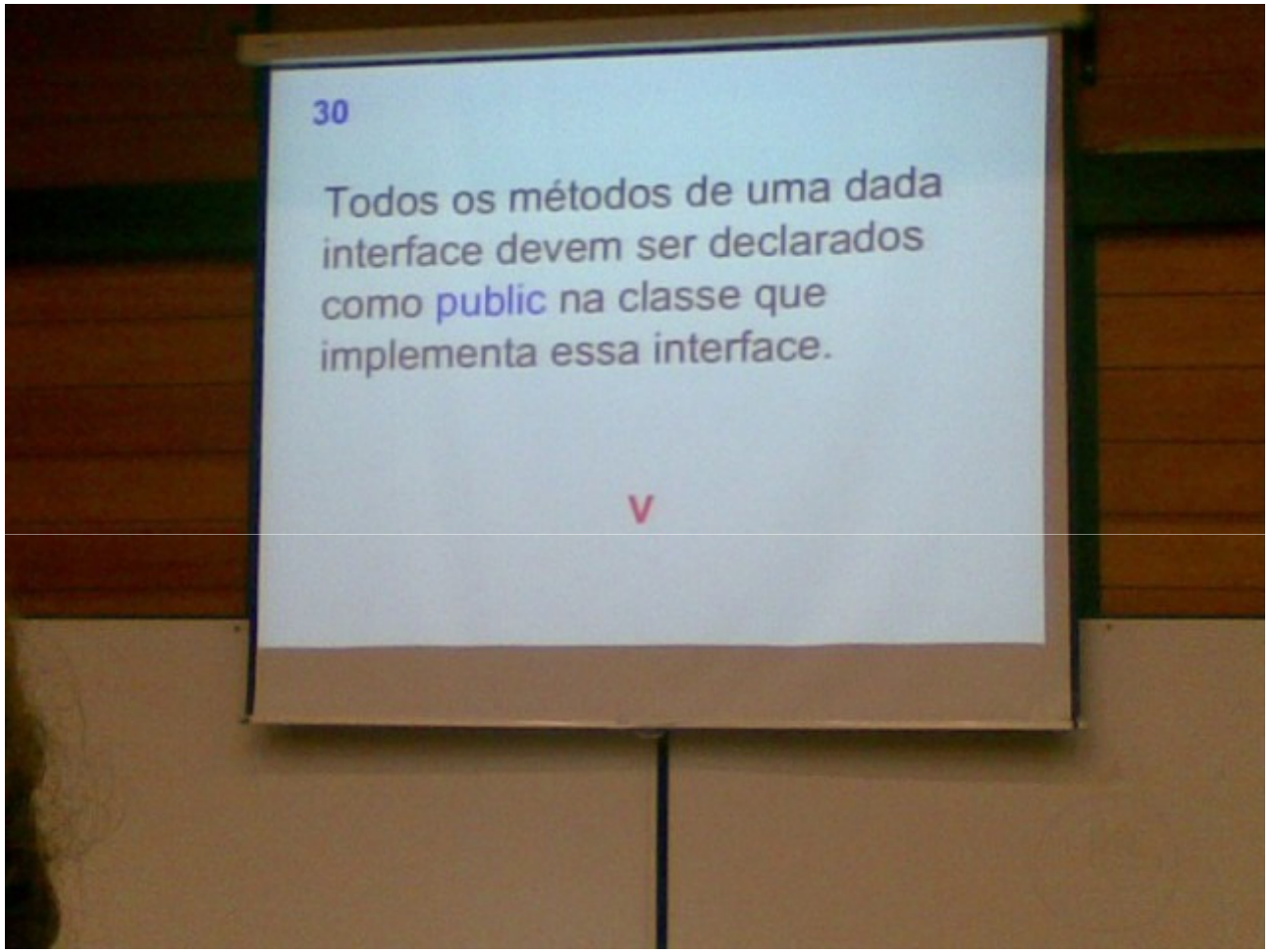

25
É possível obter um Iterador de um Map.
F


26
A expressão seguinte está correcta
Ponto2D< double > ponto =
new Ponto2D<double>(2, 3.4);
F

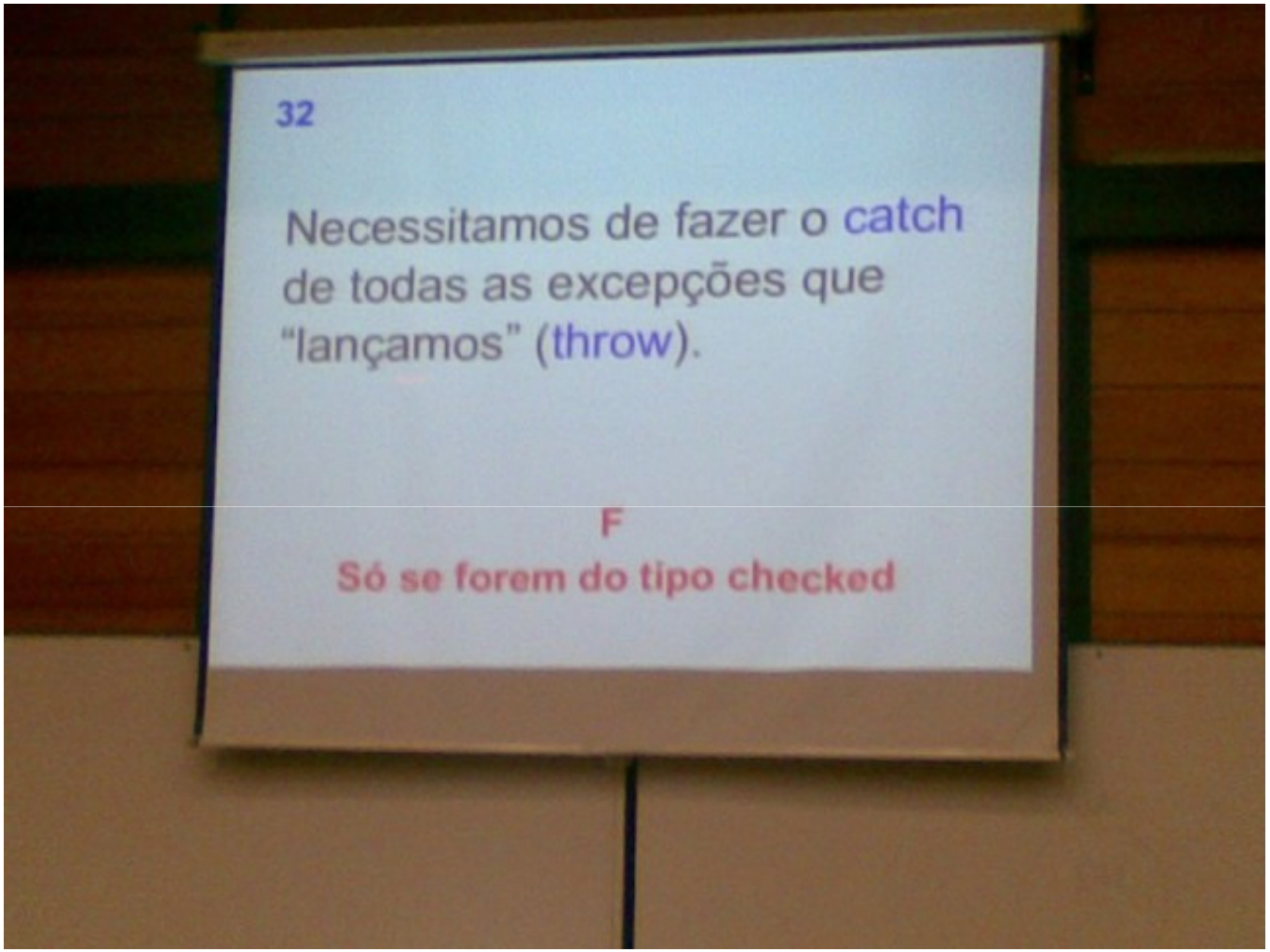

27
Este código está correcto
LinkedList<Quadrado> list =
new LinkedList<Quadrado>();
processList(list) ;
...
public void processList (LinkedList<Figura>
listOfFig ) {...}
F

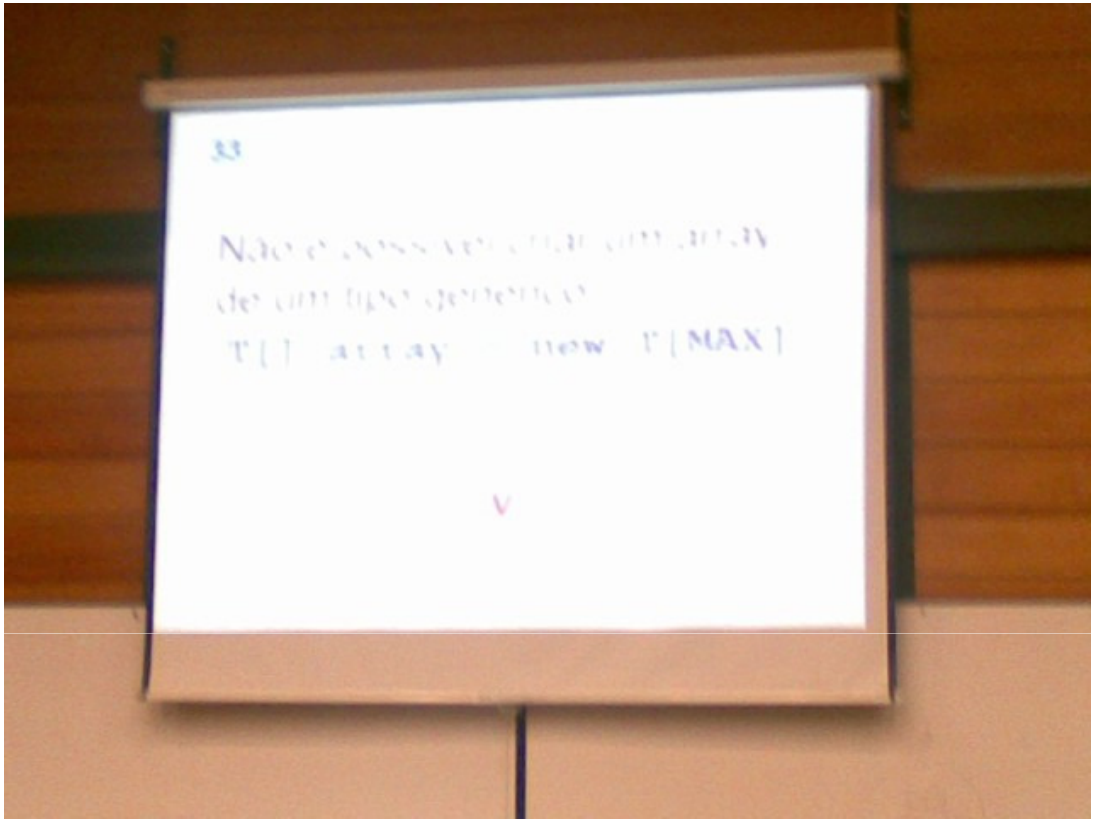

28
Uma subclasse de uma classe abstracta tem obrigatoriamente que definir todos os métodos abstractos herdados.
F


29
Um método static pode referenciar uma variável de instância de uma classe.
F


30
Todos os métodos de uma dada interface devem ser declarados como public na classe que implementa essa interface.
V


31
As implementações de Collections são Serializable.
List<Figura> ll = new LinkedList<Figura>();
for (int i = 0; i < 10; i++)
ll.add(randObject(3));
ObjectOutputStream objectOut =
new ObjectOutputStream(
new FileOutputStream("c:/tmp/out.bin"));
objectOut.writeObject(ll);
V


32
Necessitamos de fazer o catch de todas as excepções que "lançamos" (throw).
F
Só se forem do tipo checked




“Não é possível criar um array de um tipo genérico

T[] array = new T[MAX];”

V

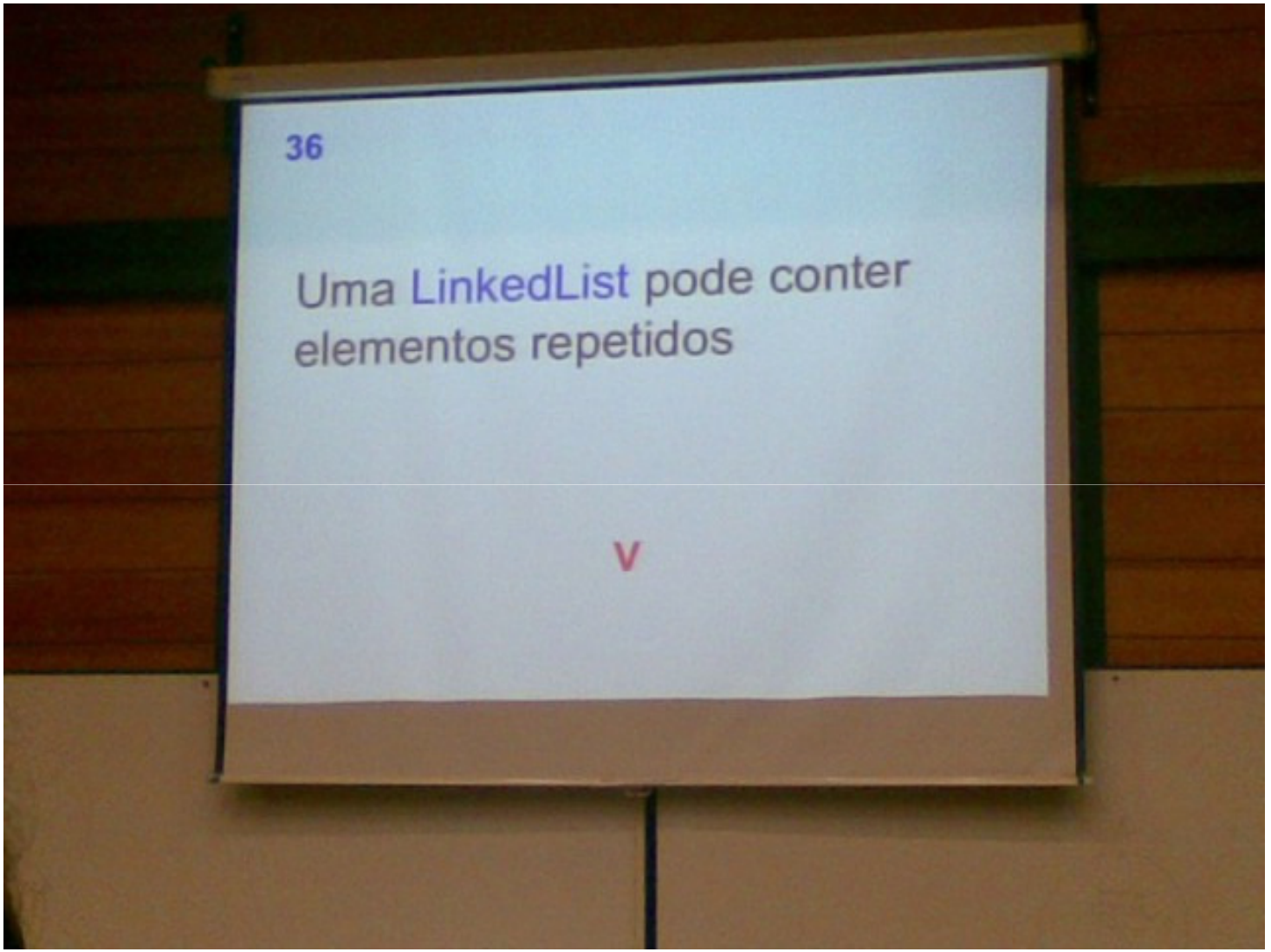
- 34

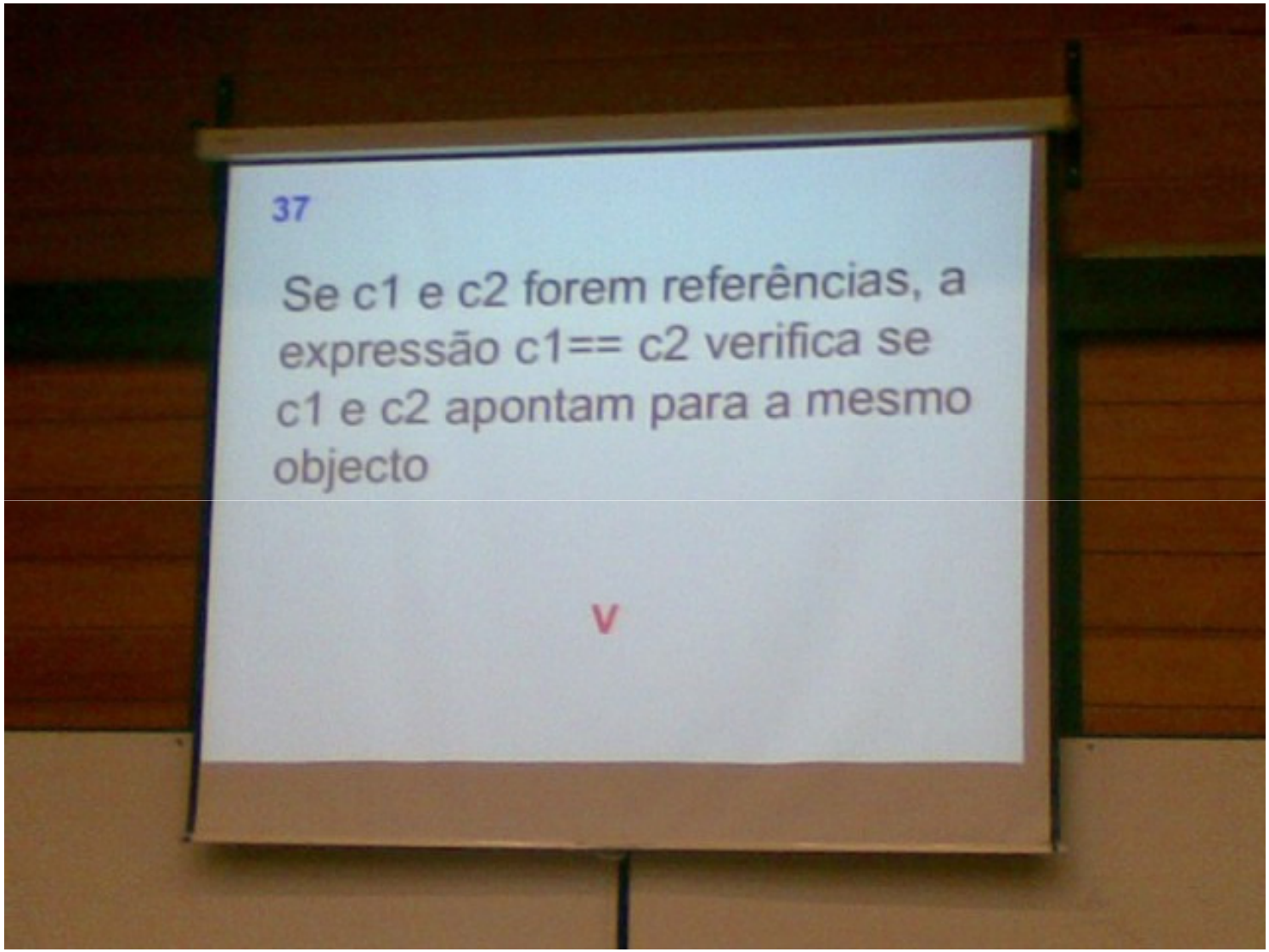
  “Classes Serializable podem ler e escrever”

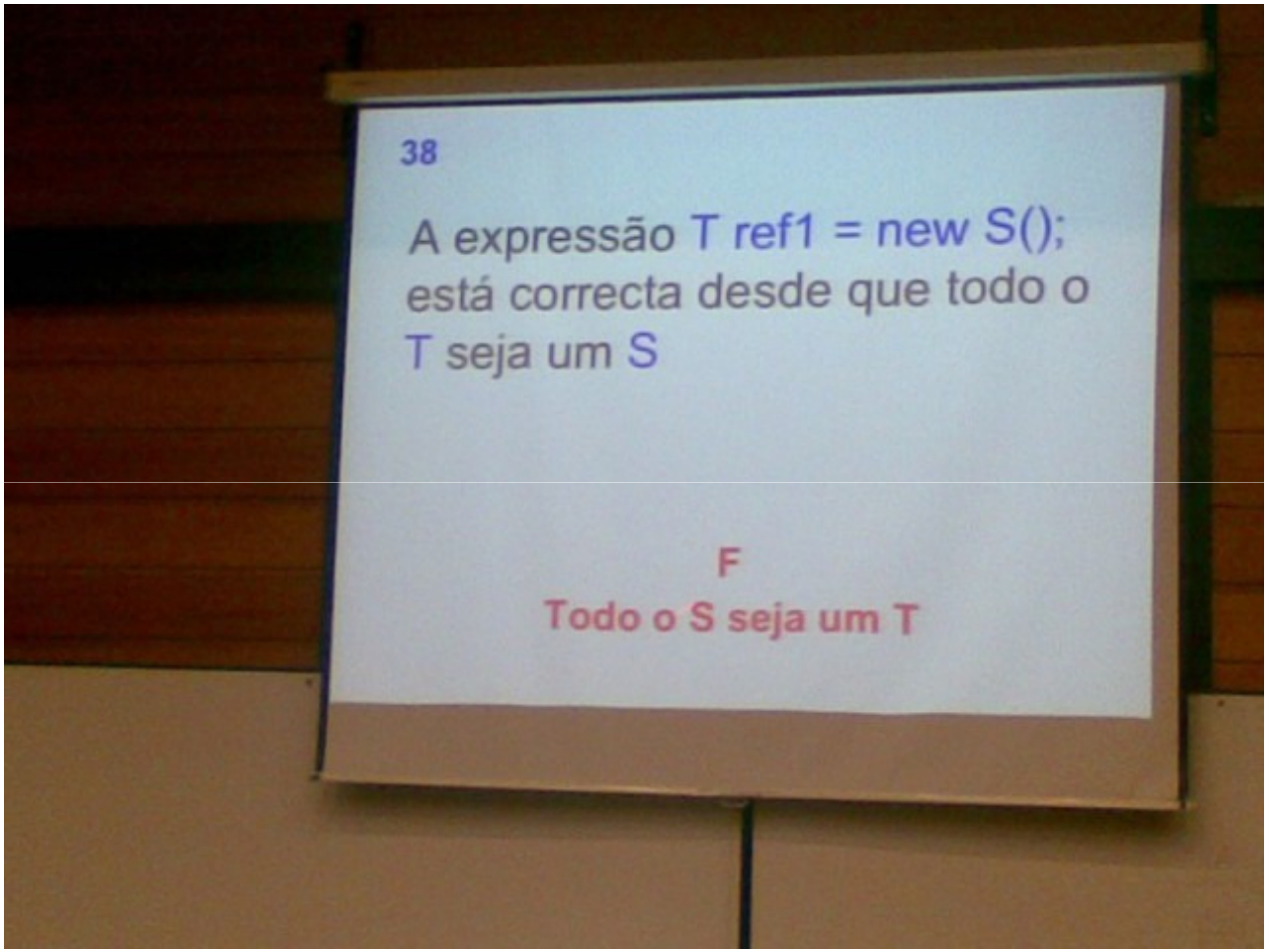
  (algo do género)

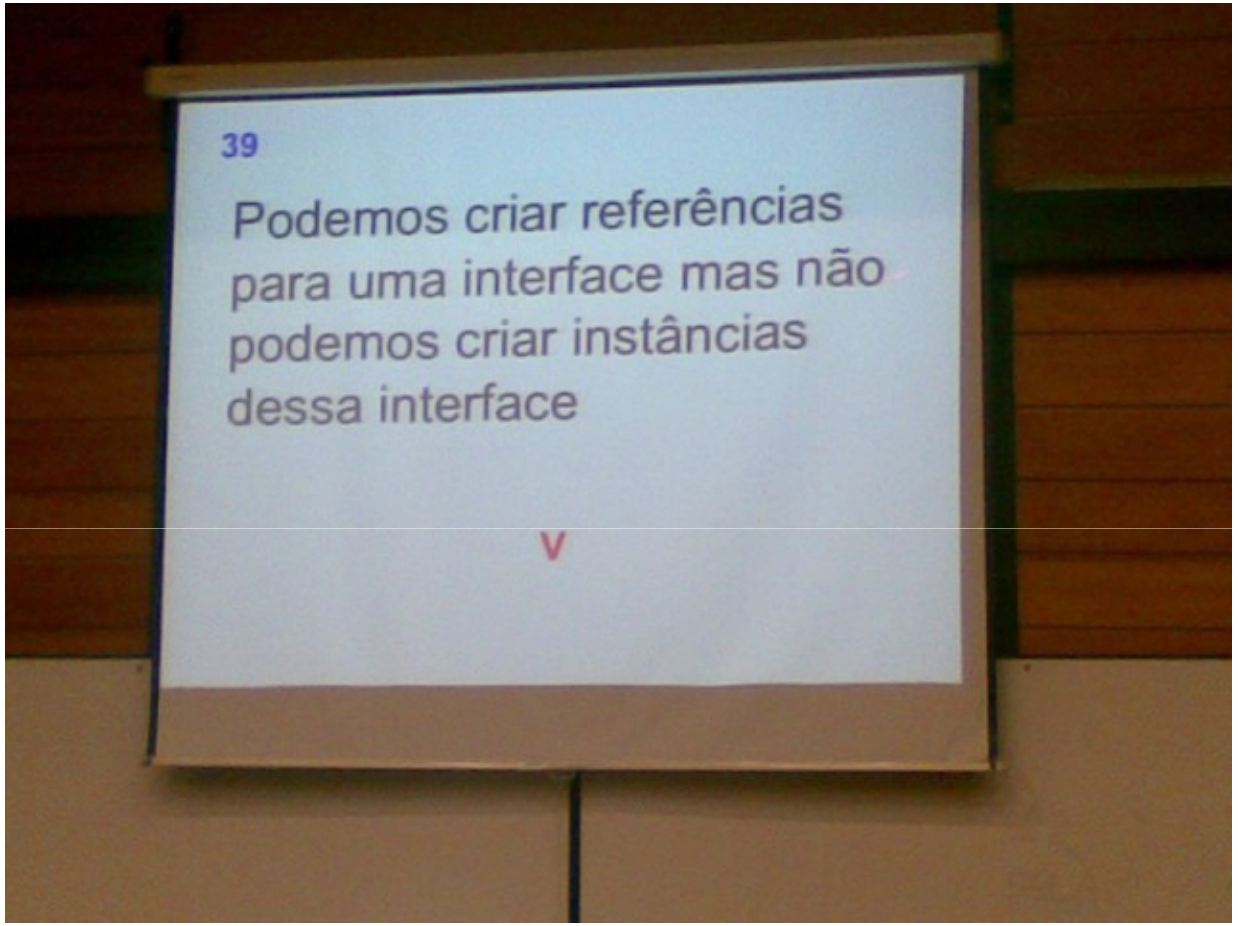
**V**

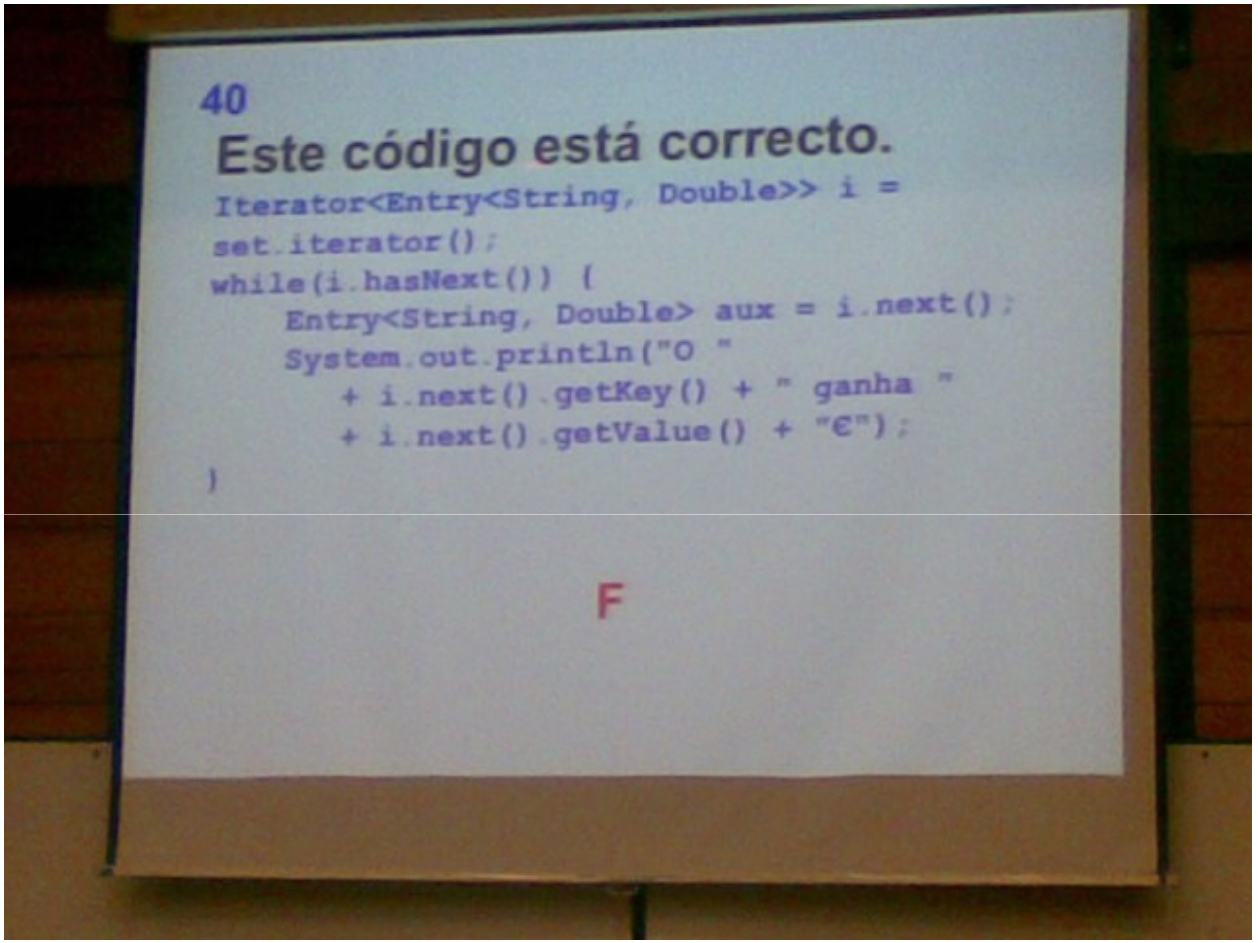

35
O ficheiro Manifest é usado pelo Eclipse para gerir cada projecto
F
Ficheiros Jar


36
Uma LinkedList pode conter elementos repetidos
V


37
Se c1 e c2 forem referências, a expressão c1== c2 verifica se c1 e c2 apontam para a mesmo objecto
V


38
A expressão T ref1 = new S();
está correcta desde que todo o
T seja um S
F
Todo o S seja um T


39
Podemos criar referências para uma interface mas não podemos criar instâncias dessa interface
V


40
Este código está correcto.
Iterator<Entry<String, Double>> i =
set.iterator();
while(i.hasNext()) {
Entry<String, Double> aux = i.next();
System.out.println("O "
+ i.next().getKey() + " ganha "
+ i.next().getValue() + "€");
}
F


FIM