CSE 2911: Usability of Programming Languages ("Programmers Are People Too")

Michael Coblenz

Today

- Discuss "Language Wars" paper
- Designing and conducting qualitative studies (part 1; part 2 next time)
  - Brief overview of running studies
  - Then focus on *usability studies*

Language Wars

- Overall impressions
- What constitutes *evidence?*
  - "Further, Boo allows the programmer to turn off the static type system (so-called Duck Typing), a decision not supported by the literature on type systems."
- How many languages do we need?
- Which RQs should we focus on?

Research Methods

Or: How We Can Obtain *Evidence*

Key Takeaway: Methods Answer Specific Questions

CATEGORIES OF METHODS

- Qualitative methods
  - Focus is on *depth* of data
  - Does not imply *no quantities*
- Quantitative methods
  - Focus is on *statistical analysis* of data

STAGES

- I don't know what I'm doing.
    - What problems are there to solve?
    - What hypotheses are worth testing?
- I have a tool. Let's make it better.
- I have a tool. Can people use it?
- I have a tool. Let's try to show that it IS better.

Qualitative studies

GENERATING HYPOTHESES

NOT JUST ANY HYPOTHESES…

- Want to only test hypotheses that are probably true.
- You can publish a paper even if all you have is a hypothesis!
  - (if it is well-justified)
- And what if your 🧠 is empty?

QUALITATIVE STUDIES

- Want to understand something we don't understand yet.
  - What problems do factory workers have?
  - What is it like to write code for Indy 500 cars?
  - What usability problems do people have when they use my "awesome" system?

KINDS OF QUALITATIVE STUDIES

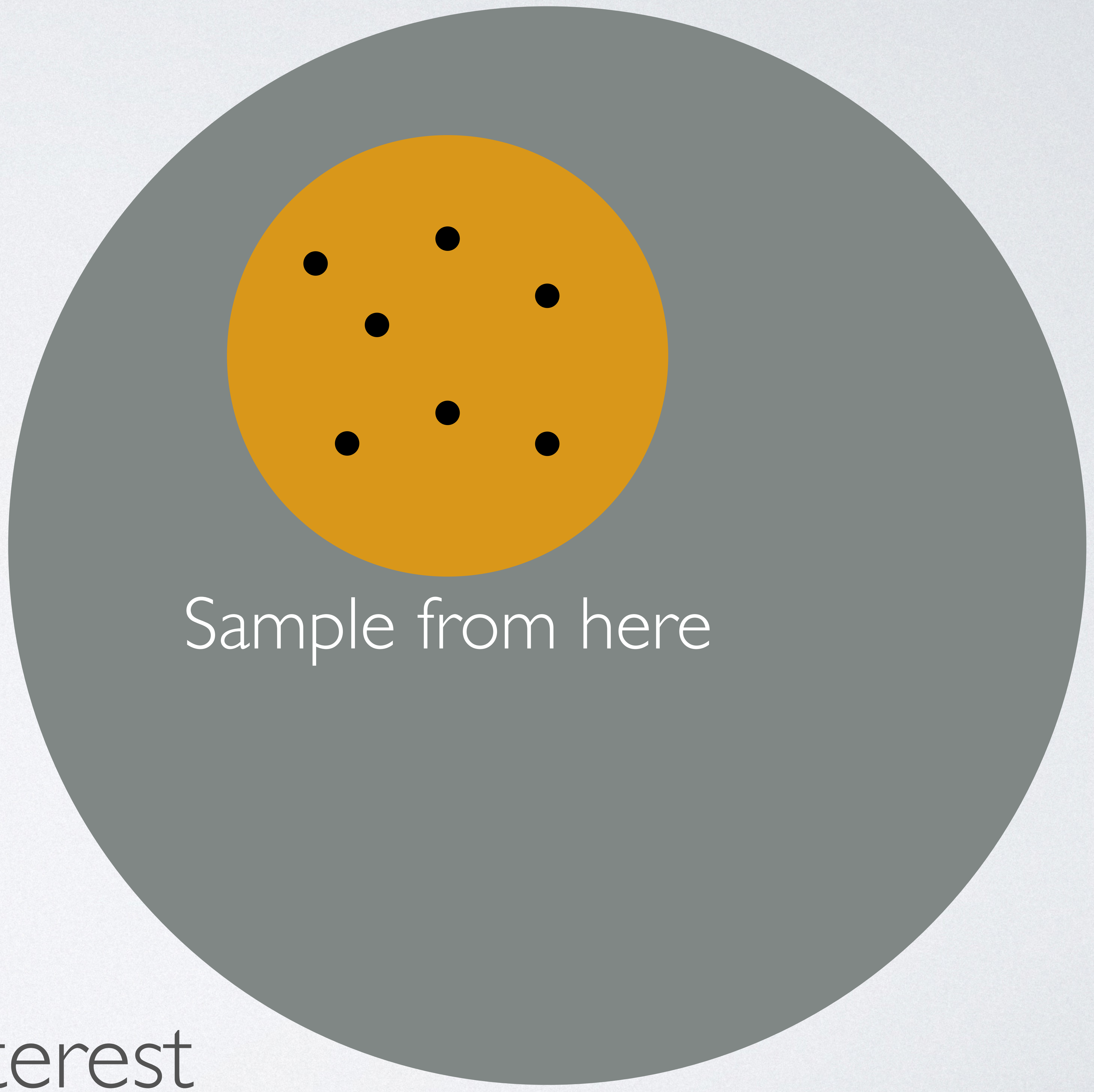
Study	Purpose
Interviews	Learn from experts independently
Focus groups	Learn from experts, stimulating conversation
Surveys	Generalize experiences
Think-aloud usability studies	Identify challenges
Corpus study	Learn from existing data

GENERALIZABILITY

If you want to argue your results generalize to X, then ideally you should sample from X.

Plan B: argue X is similar to the population you sampled from.

Examples?



THINK-ALOUD USABILITY STUDIES

- Give people tasks and observe what happens.
- NOT experiments
- NOT controlled
- NOT comparative
- Just want to see what problems people encounter.
- Follow "think-aloud" protocol

USABILITY STUDIES CAN SHOW

- X% of my participants completed the task in 30 minutes.
- Participants encountered the following problems…
- Only participants who knew X were able to do the task.

USABILITY STUDIES CANNOT SHOW

- My system is better than an existing system.

USABILITY STUDY OVERVIEW

- Running usability studies requires:
  - Ethics approval
  - Recruiting
  - Training
  - Task design
  - Data collection/analysis

ETHICS REVIEW

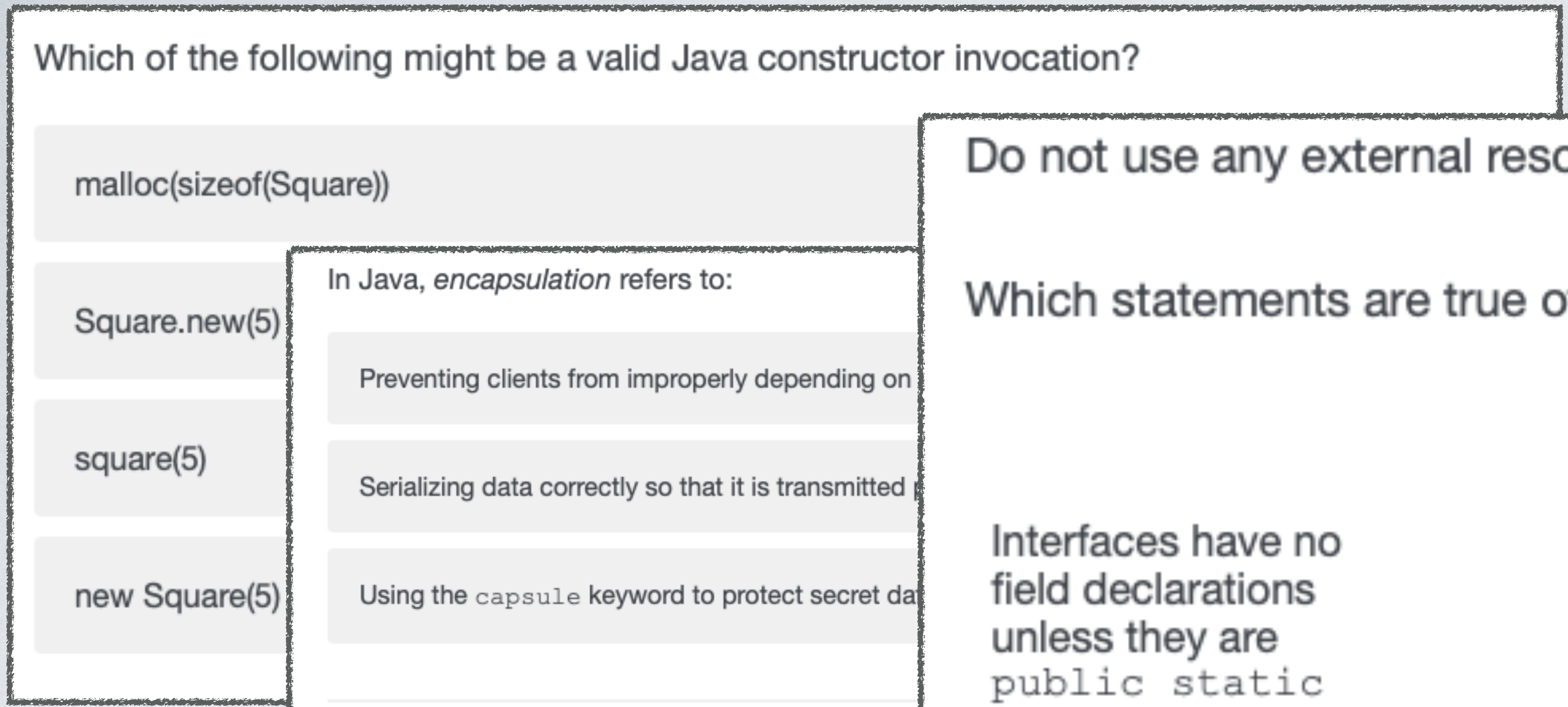
- For research: need to submit proposal to Institutional Review Board (IRB)
- For class: no need to get IRB approval (IRB only supervises *research*)

ETHICS

- What if incentive is too high?
- What if incentive is too low?
    - IRB reviews incentives
- What if recruitment is misleading?
    - IRB reviews recruitment materials

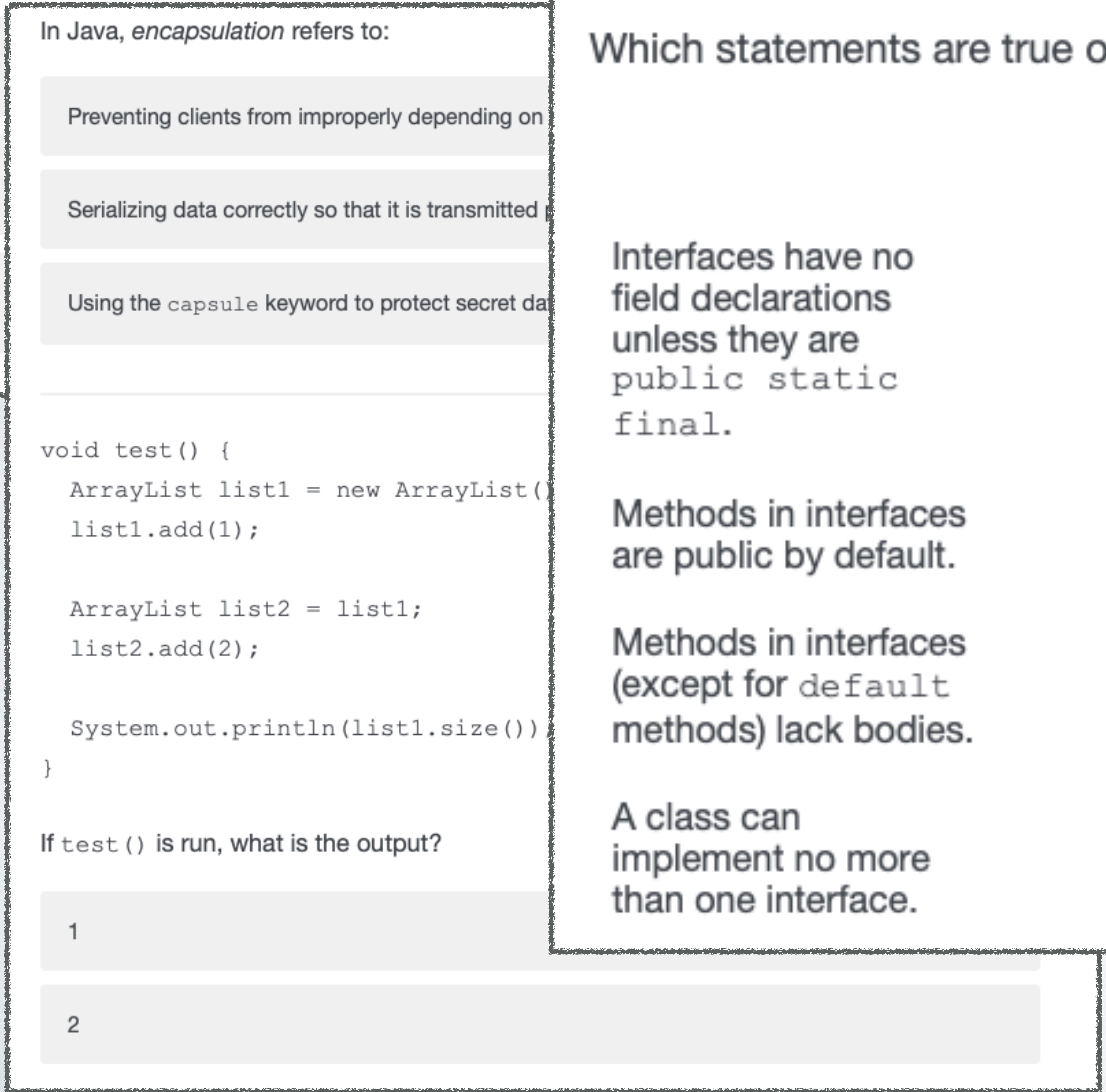
PARTICIPANT PRE-SCREENING

- Can issue a pre-test to avoid wasting time on unqualified participants.
- Issues:
    - How will you incentivize people to take the test?
    - Can you use the test results in your research?



Which of the following might be a valid Java constructor invocation?

- malloc(sizeof(Square))
- Square.new(5)
- square(5)
- new Square(5)



In Java, *encapsulation* refers to:

- Preventing clients from improperly depending on
- Serializing data correctly so that it is transmitted
- Using the `capsule` keyword to protect secret da

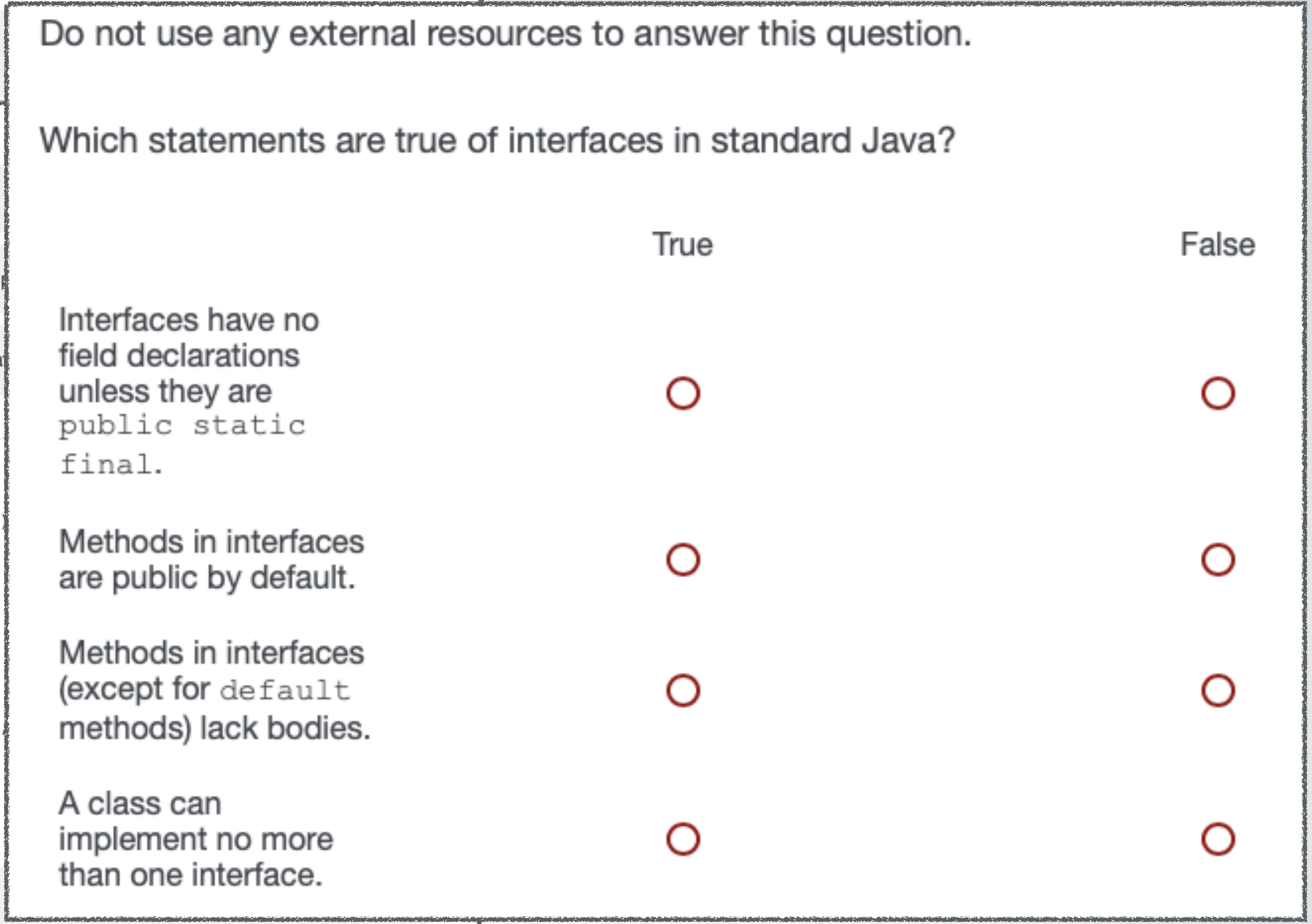
```
void test() {
  ArrayList list1 = new ArrayList()
  list1.add(1);

  ArrayList list2 = list1;
  list2.add(2);

  System.out.println(list1.size())
}
```

If `test()` is run, what is the output?

- 1
- 2



Do not use any external resources to answer this question.

Which statements are true of interfaces in standard Java?

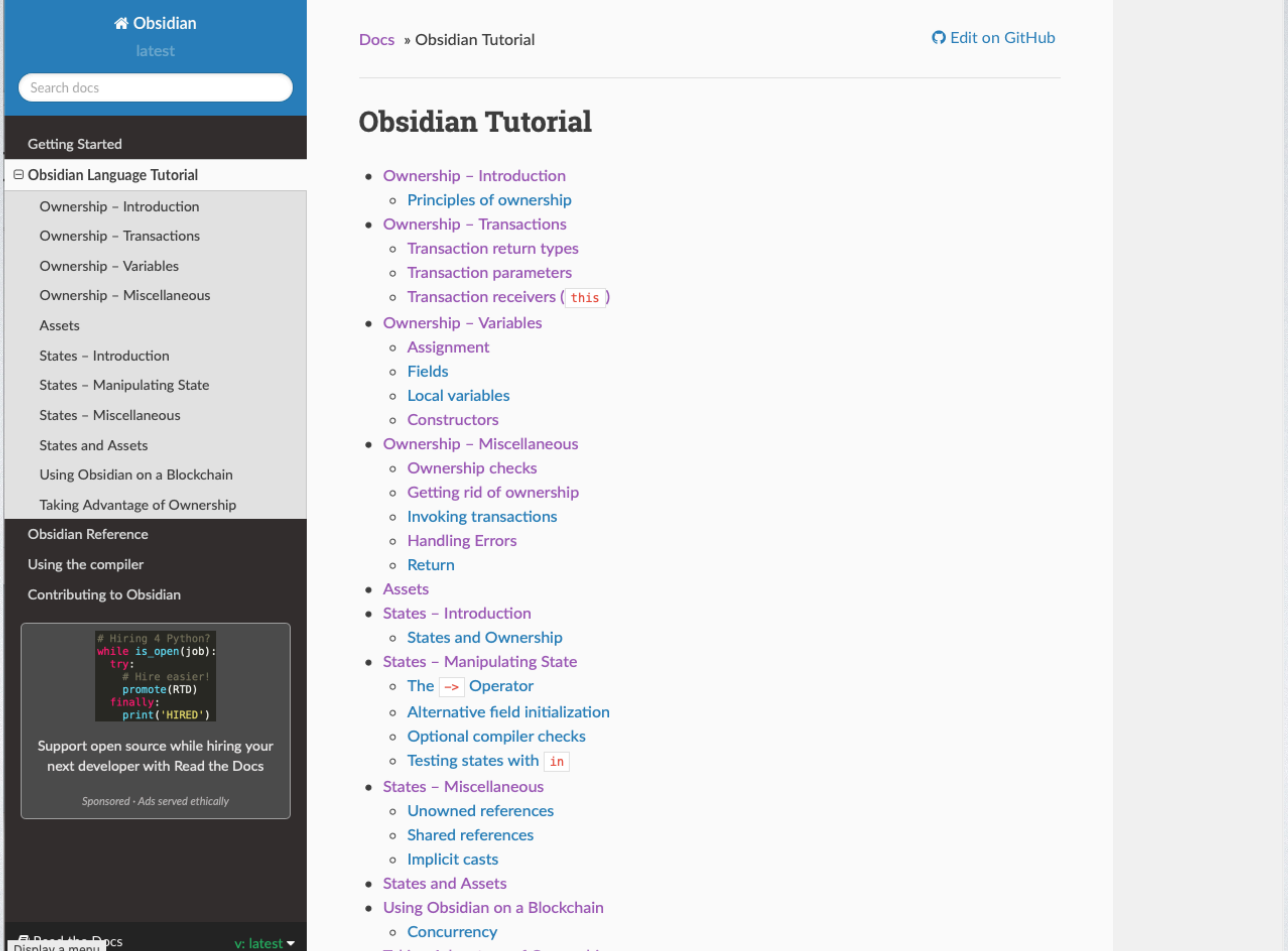
	True	False
Interfaces have no field declarations unless they are `public static final`.	○	○
Methods in interfaces are public by default.	○	○
Methods in interfaces (except for `default` methods) lack bodies.	○	○
A class can implement no more than one interface.	○	○

DEMOGRAPHICS

- Collect information if you want it!
- Programming experience? Languages?
- If they tell you, you can use it…
- e.g. Gender___________

TRAINING

- How will you prepare your participants?
- People don't read.
- People think they understand but in fact do not.
- Teach…and then assess.
- Or: decide that no training is necessary.



Obsidian
latest

Search docs

```
# Hiring 4 Python?
while is_open(job):
  try:
    # Hire easier!
    promote(RTD)
  finally:
    print('HIRED')
```

Support open source while hiring your next developer with Read the Docs

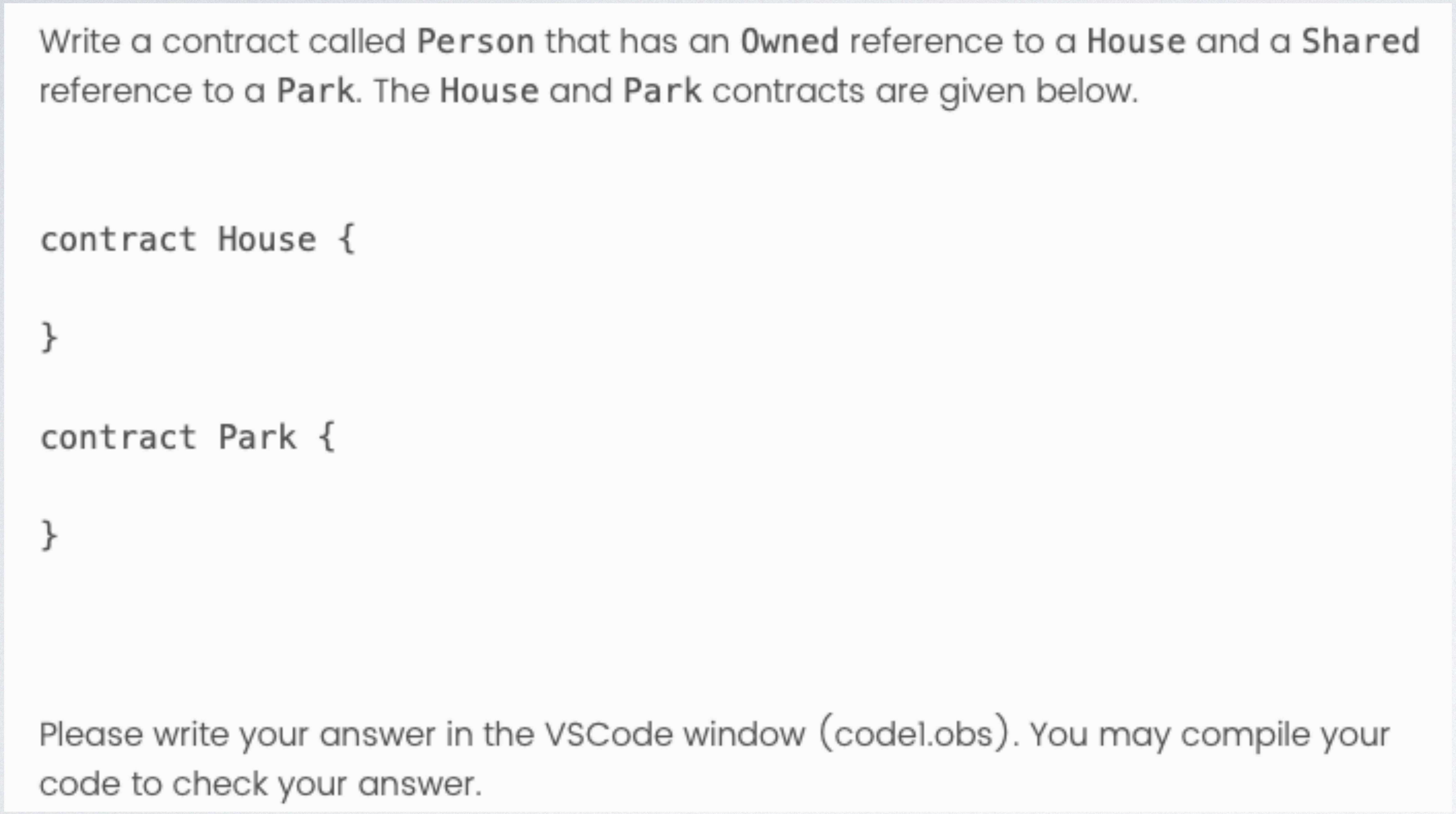
Sponsored · Ads served ethically

Read the Docs
Display a menu
v: latest

Docs » Obsidian Tutorial

Edit on GitHub

Obsidian Tutorial



Write a contract called **Person** that has an **Owned** reference to a **House** and a **Shared** reference to a **Park**. The **House** and **Park** contracts are given below.

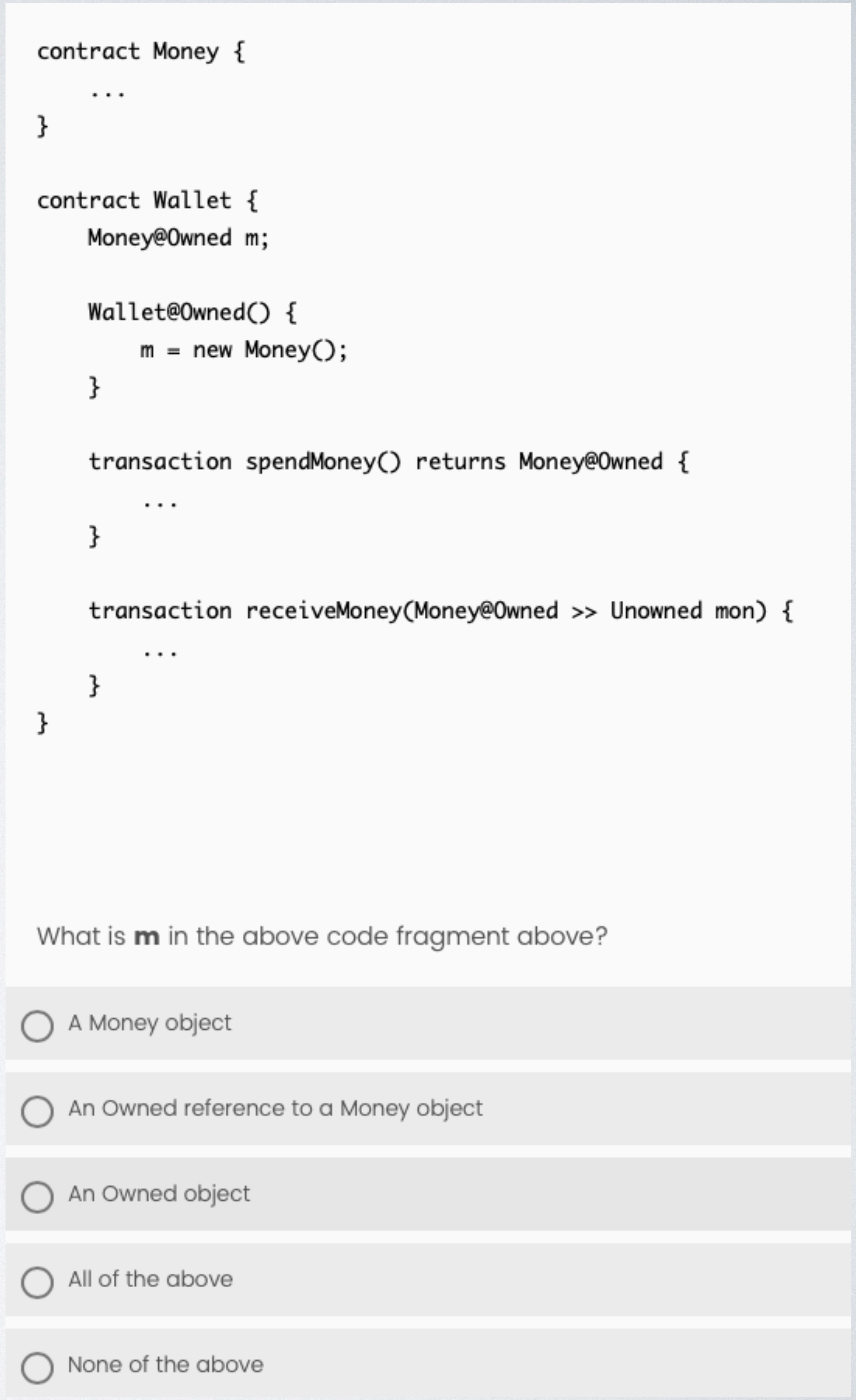
```
contract House {

}

contract Park {

}
```

Please write your answer in the VSCode window (code1.obs). You may compile your code to check your answer.



```
contract Money {
    ...
}

contract Wallet {
    Money@Owned m;

    Wallet@Owned() {
        m = new Money();
    }

    transaction spendMoney() returns Money@Owned {
        ...
    }

    transaction receiveMoney(Money@Owned >> Unowned mon) {
        ...
    }
}
```

What is **m** in the above code fragment above?

- ○ A Money object
- ○ An Owned reference to a Money object
- ○ An Owned object
- ○ All of the above
- ○ None of the above

TASKS

- This is the hardest part of study design.
- You will not get this right the first time.
- Solution: pilot repeatedly.
- But: you can use data from your "pilots" if you follow protocol.
- (a true "pilot" involves throwing the data out)
- What is the distribution over task times?