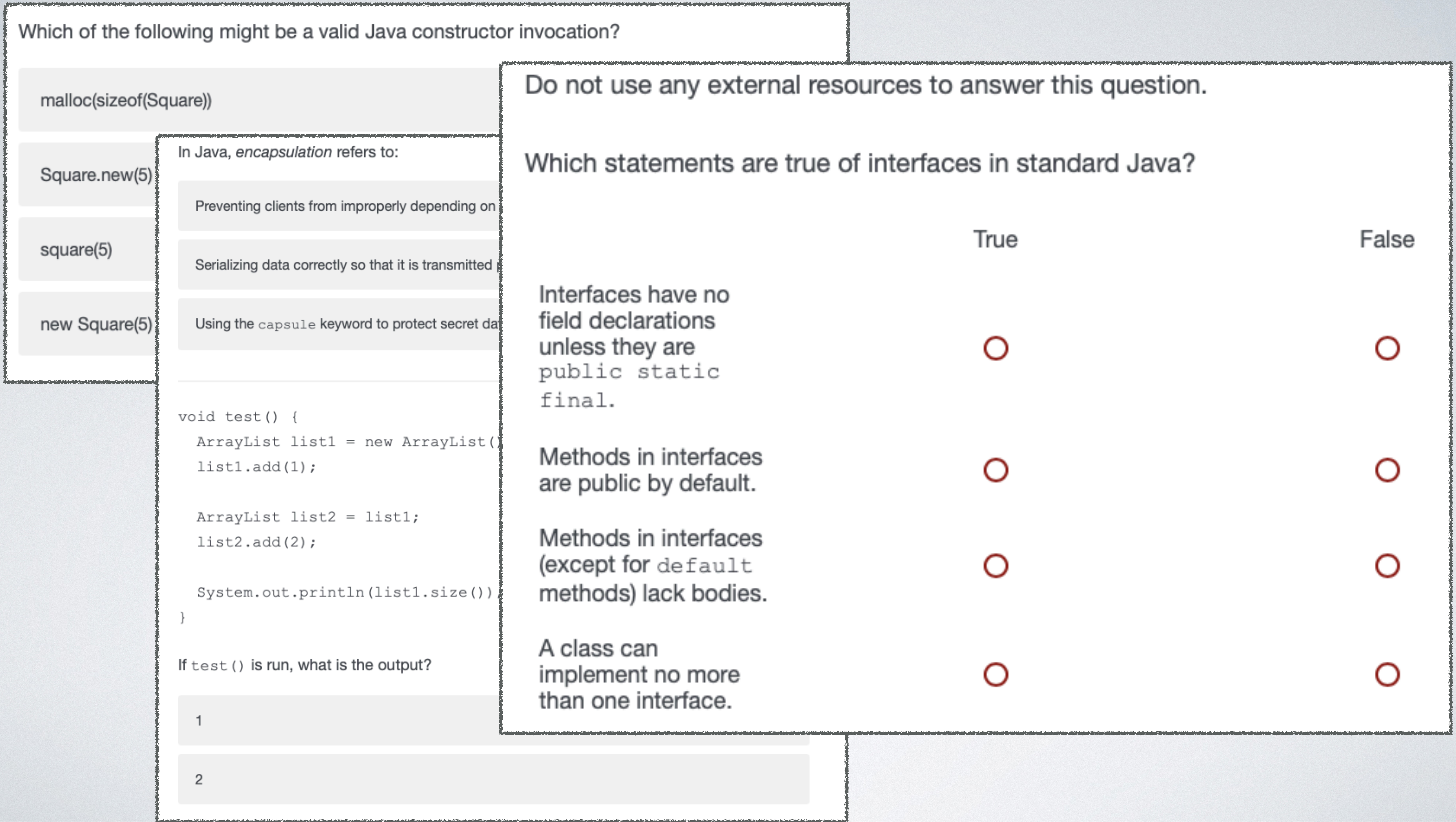
USER-CENTERED LANGUAGE DESIGN, PART 2: TASKS

Michael Coblenz



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?

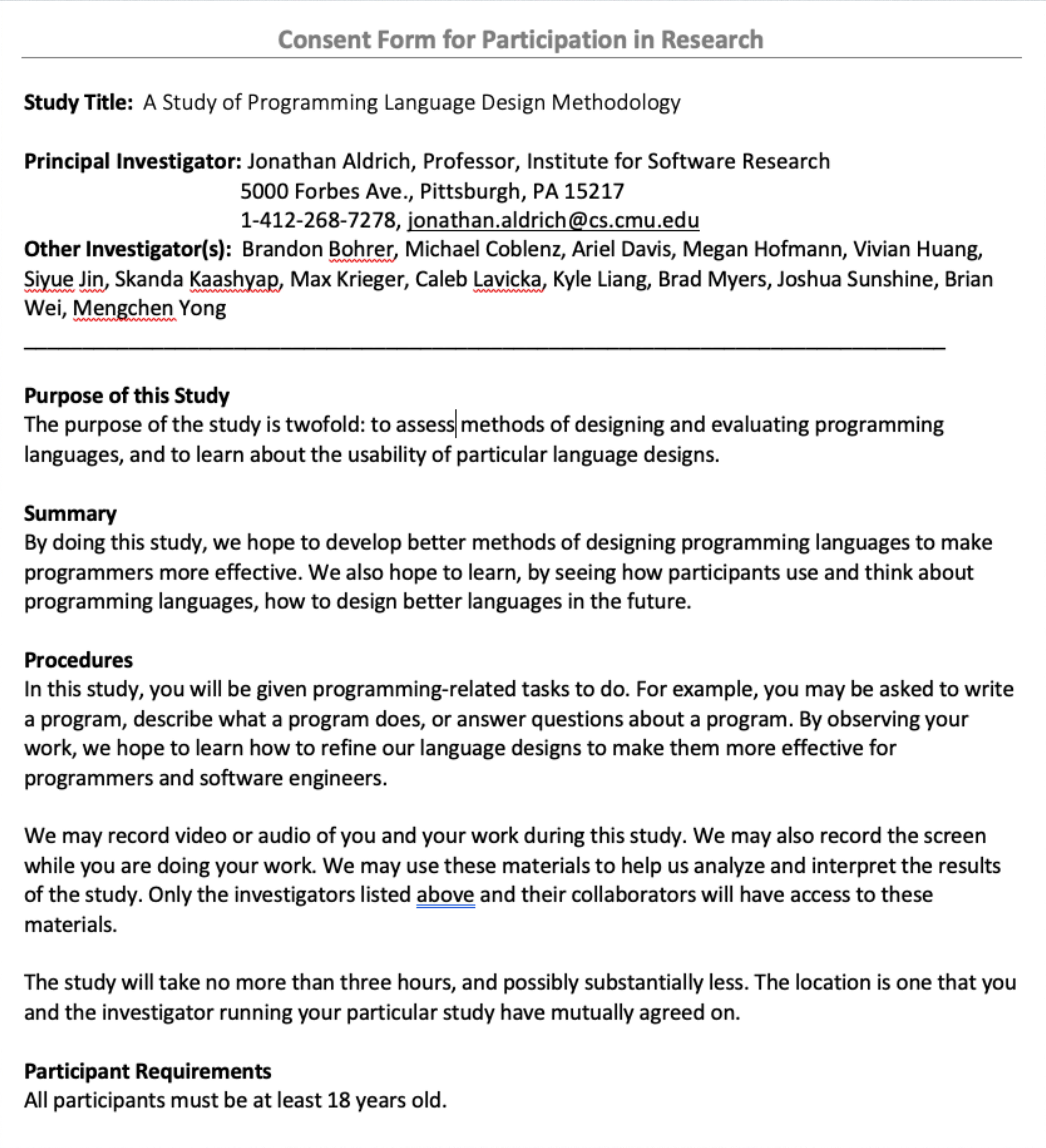
	True	False
Interfaces have no field declarations unless they are `public static final`.	○	○
Methods in interfaces are public by default.	○	○
Methods in interfaces (except for `default` methods) lack bodies.	○	○
A class can implement no more than one interface.	○	○

YOUR TURN

- Assume you have some tasks for the usability question from before.
- Who will you try to recruit?
- How will you:
  - Recruit?
  - Incentivize?
  - Screen?

INFORMED CONSENT

- Online consent is OK (with IRB approval)
- We proposed paper-based consent for this study



Consent Form for Participation in Research

Study Title: A Study of Programming Language Design Methodology

Principal Investigator: Jonathan Aldrich, Professor, Institute for Software Research
5000 Forbes Ave., Pittsburgh, PA 15217
1-412-268-7278, jonathan.aldrich@cs.cmu.edu

Other Investigator(s): Brandon Bohrer, Michael Coblenz, Ariel Davis, Megan Hofmann, Vivian Huang, Siyue Jin, Skanda Kaashyap, Max Krieger, Caleb Lavicka, Kyle Liang, Brad Myers, Joshua Sunshine, Brian Wei, Mengchen Yong

---

Purpose of this Study

The purpose of the study is twofold: to assess methods of designing and evaluating programming languages, and to learn about the usability of particular language designs.

Summary

By doing this study, we hope to develop better methods of designing programming languages to make programmers more effective. We also hope to learn, by seeing how participants use and think about programming languages, how to design better languages in the future.

Procedures

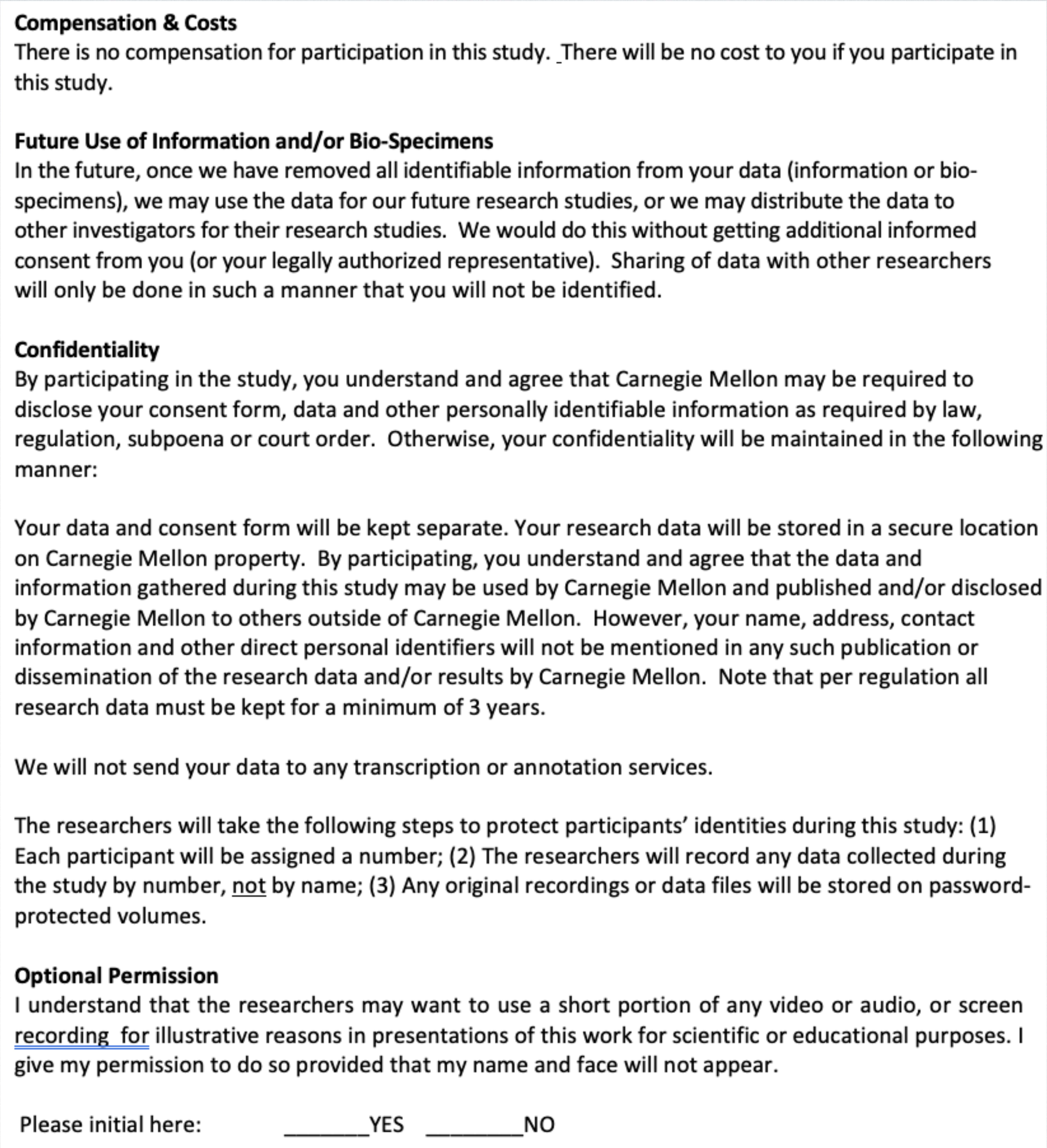
In this study, you will be given programming-related tasks to do. For example, you may be asked to write a program, describe what a program does, or answer questions about a program. By observing your work, we hope to learn how to refine our language designs to make them more effective for programmers and software engineers.

We may record video or audio of you and your work during this study. We may also record the screen while you are doing your work. We may use these materials to help us analyze and interpret the results of the study. Only the investigators listed above and their collaborators will have access to these materials.

The study will take no more than three hours, and possibly substantially less. The location is one that you and the investigator running your particular study have mutually agreed on.

Participant Requirements

All participants must be at least 18 years old.



Compensation & Costs
There is no compensation for participation in this study. There will be no cost to you if you participate in this study.

Future Use of Information and/or Bio-Specimens
In the future, once we have removed all identifiable information from your data (information or bio-specimens), we may use the data for our future research studies, or we may distribute the data to other investigators for their research studies. We would do this without getting additional informed consent from you (or your legally authorized representative). Sharing of data with other researchers will only be done in such a manner that you will not be identified.

Confidentiality
By participating in the study, you understand and agree that Carnegie Mellon may be required to disclose your consent form, data and other personally identifiable information as required by law, regulation, subpoena or court order. Otherwise, your confidentiality will be maintained in the following manner:

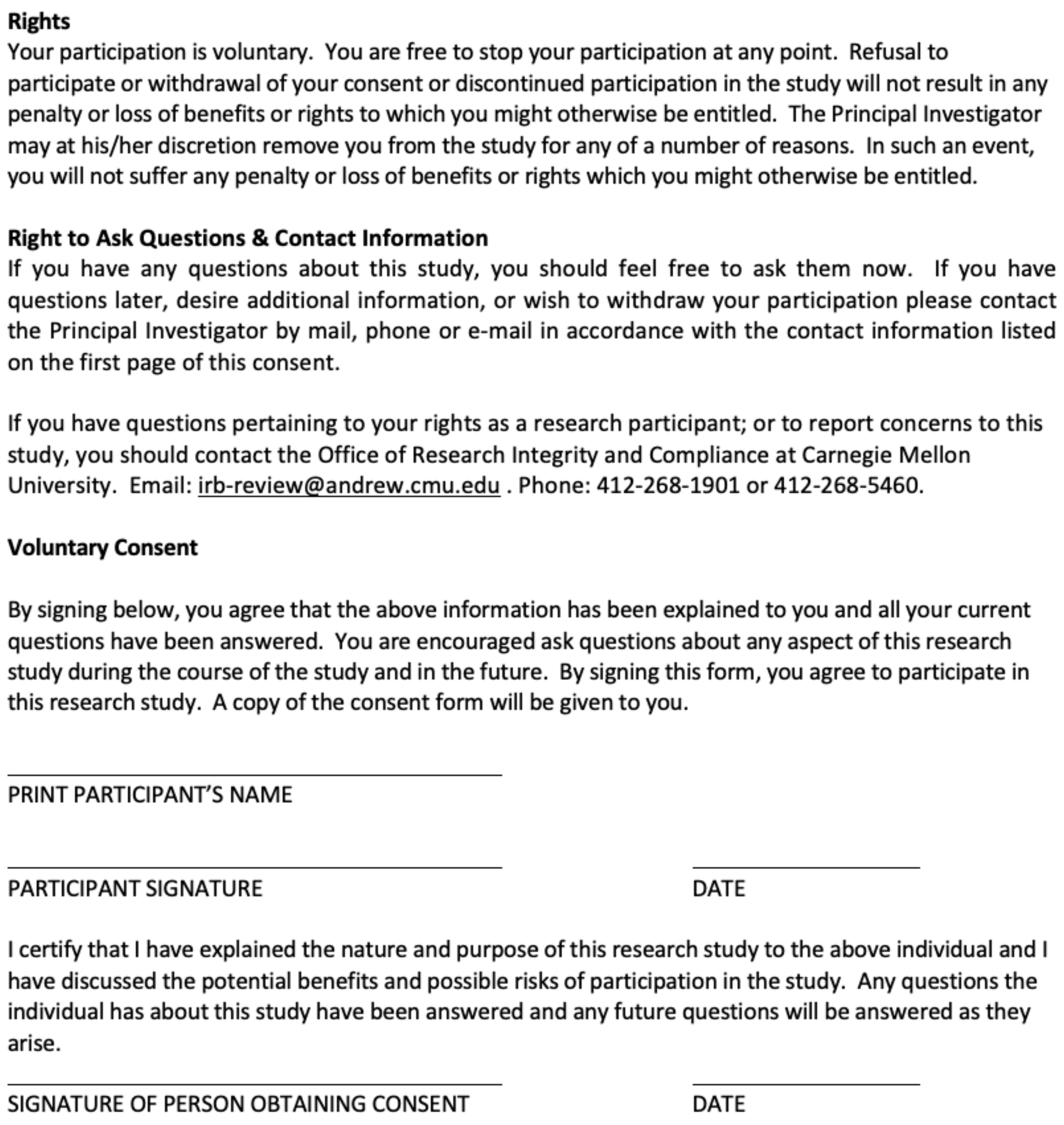
Your data and consent form will be kept separate. Your research data will be stored in a secure location on Carnegie Mellon property. By participating, you understand and agree that the data and information gathered during this study may be used by Carnegie Mellon and published and/or disclosed by Carnegie Mellon to others outside of Carnegie Mellon. However, your name, address, contact information and other direct personal identifiers will not be mentioned in any such publication or dissemination of the research data and/or results by Carnegie Mellon. Note that per regulation all research data must be kept for a minimum of 3 years.

We will not send your data to any transcription or annotation services.

The researchers will take the following steps to protect participants' identities during this study: (1) Each participant will be assigned a number; (2) The researchers will record any data collected during the study by number, not by name; (3) Any original recordings or data files will be stored on password-protected volumes.

Optional Permission
I understand that the researchers may want to use a short portion of any video or audio, or screen recording for illustrative reasons in presentations of this work for scientific or educational purposes. I give my permission to do so provided that my name and face will not appear.

Please initial here: _______YES ________NO



Rights

Your participation is voluntary. You are free to stop your participation at any point. Refusal to participate or withdrawal of your consent or discontinued participation in the study will not result in any penalty or loss of benefits or rights to which you might otherwise be entitled. The Principal Investigator may at his/her discretion remove you from the study for any of a number of reasons. In such an event, you will not suffer any penalty or loss of benefits or rights which you might otherwise be entitled.

Right to Ask Questions & Contact Information

If you have any questions about this study, you should feel free to ask them now. If you have questions later, desire additional information, or wish to withdraw your participation please contact the Principal Investigator by mail, phone or e-mail in accordance with the contact information listed on the first page of this consent.

If you have questions pertaining to your rights as a research participant; or to report concerns to this study, you should contact the Office of Research Integrity and Compliance at Carnegie Mellon University. Email: irb-review@andrew.cmu.edu . Phone: 412-268-1901 or 412-268-5460.

Voluntary Consent

By signing below, you agree that the above information has been explained to you and all your current questions have been answered. You are encouraged ask questions about any aspect of this research study during the course of the study and in the future. By signing this form, you agree to participate in this research study. A copy of the consent form will be given to you.

PRINT PARTICIPANT'S NAME

PARTICIPANT SIGNATURE DATE

I certify that I have explained the nature and purpose of this research study to the above individual and I have discussed the potential benefits and possible risks of participation in the study. Any questions the individual has about this study have been answered and any future questions will be answered as they arise.

SIGNATURE OF PERSON OBTAINING CONSENT DATE

DEMOGRAPHICS

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

Participant code_________

Pre-study questionnaire

1.How long have you been programming?______ years ______ months

2.Gender: __________

3.If you have any academic computer science background, what degrees have you completed? If you are partway through a degree, what degree and how far?

____

4.How much professional (paid) software development experience do you have?______ years ______ months

5.List any programming languages in which you are currently comfortable programming in DECREASING order of familiarity.

____

6.How much experience do you have programming in Java? Include time spent doing Java part-time, such as in a course.

______ years ______ months

7.Please rate your level of expertise in Java by circling one option:

beginner   intermediate   advanced

8.Please rate your level of expertise in Rust by circling one option:

none   beginner   intermediate   advanced

9.Please rate your level of expertise in each of the following blockchain programming languages/environments:

Solidity:

I don’t know what this is/none   beginner   intermediate   advanced

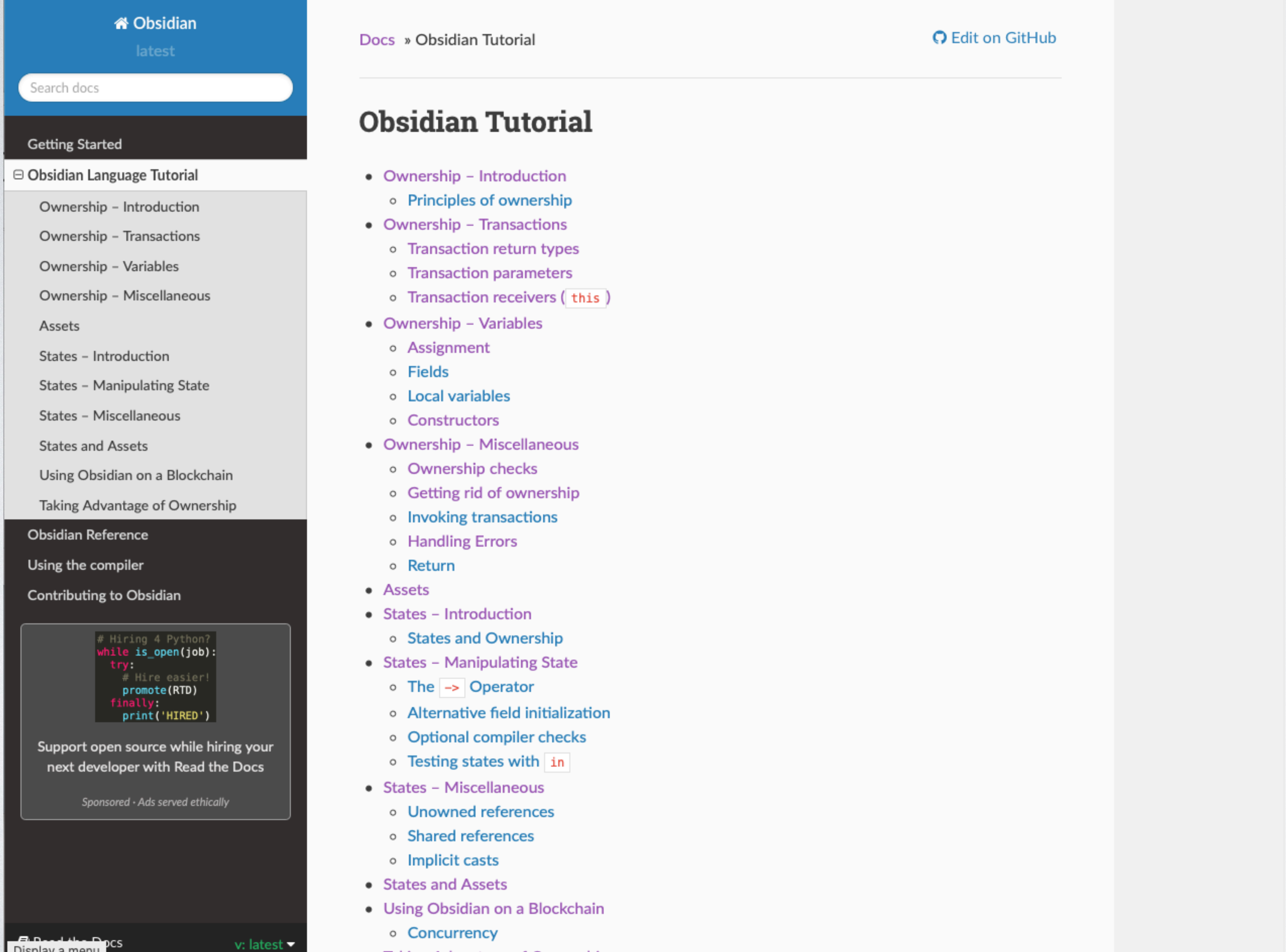
Hyperledger Fabric:

I don’t know what this is/none   beginner   intermediate   advanced

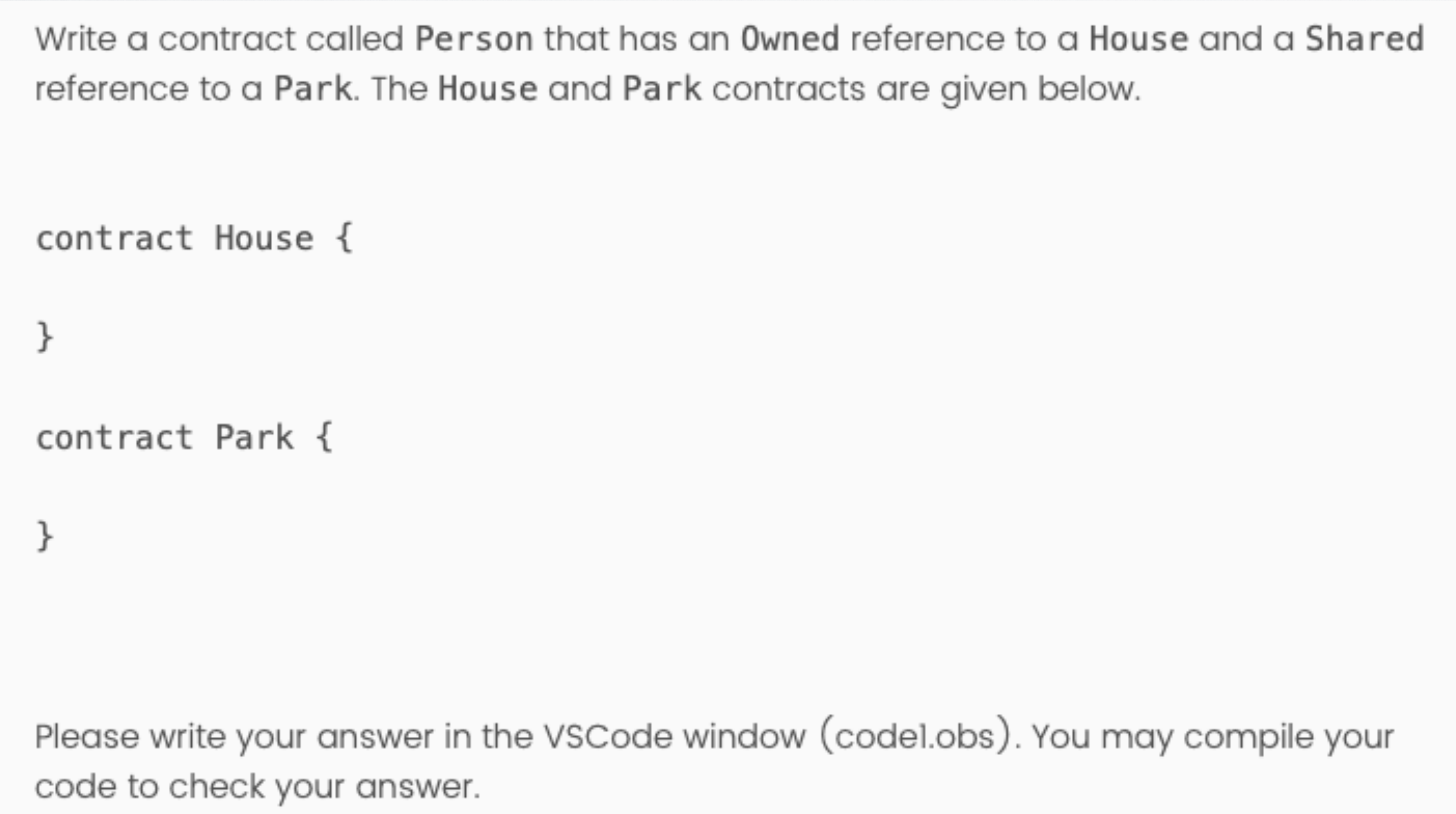
Other (specify which)_______________________________________

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 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.

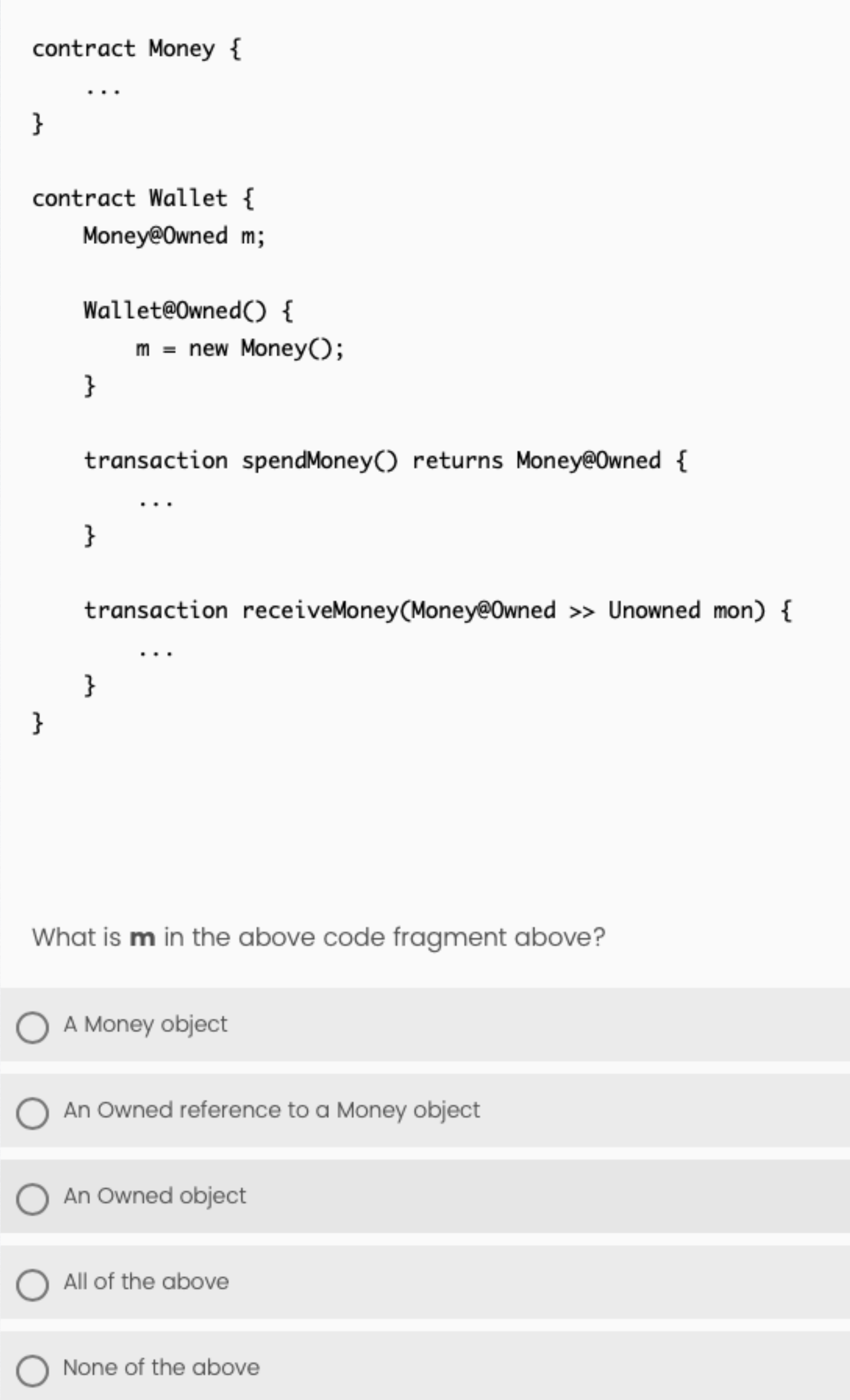
```
contract House {

}

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?

USABILITY STUDY TASKS

- Choose an *interesting* task
  - One that you think might be hard
  - One that is central to the usability of your design
- Can't test everything

TASK IDEAS

- Write a program according to this specification.
- Are there bugs in this code? If so, what are they?
- Fill in the missing code…
- What does this code do?
- Answer these questions about this code.

PARSONS PROBLEMS

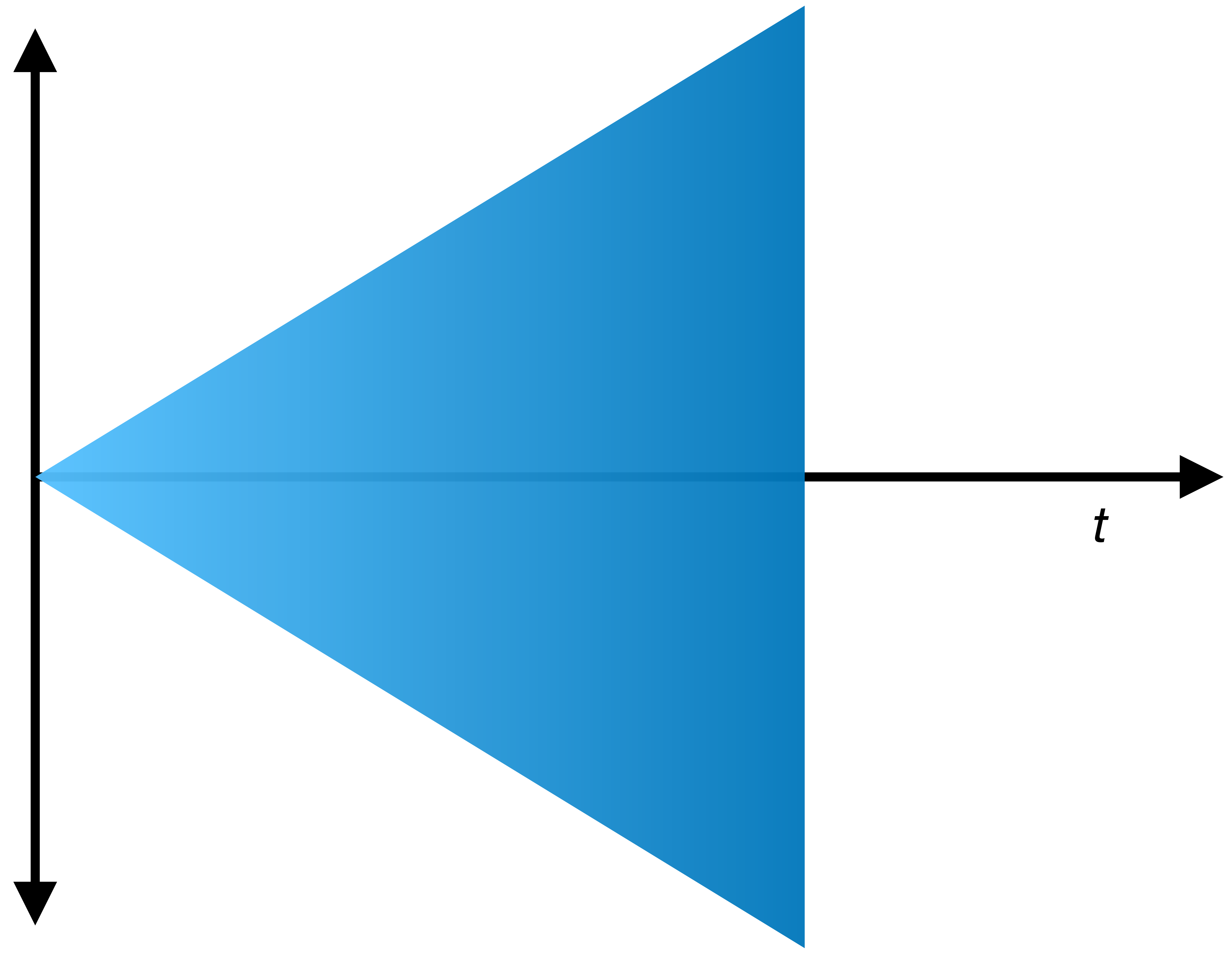
- Participant is given snippets of code, but they are out of order.
- Task: put them in the right order.
- "We find notable correlation between Parsons scores and code writing scores. We find low correlation between code writing and tracing and between Parsons and tracing." [1]

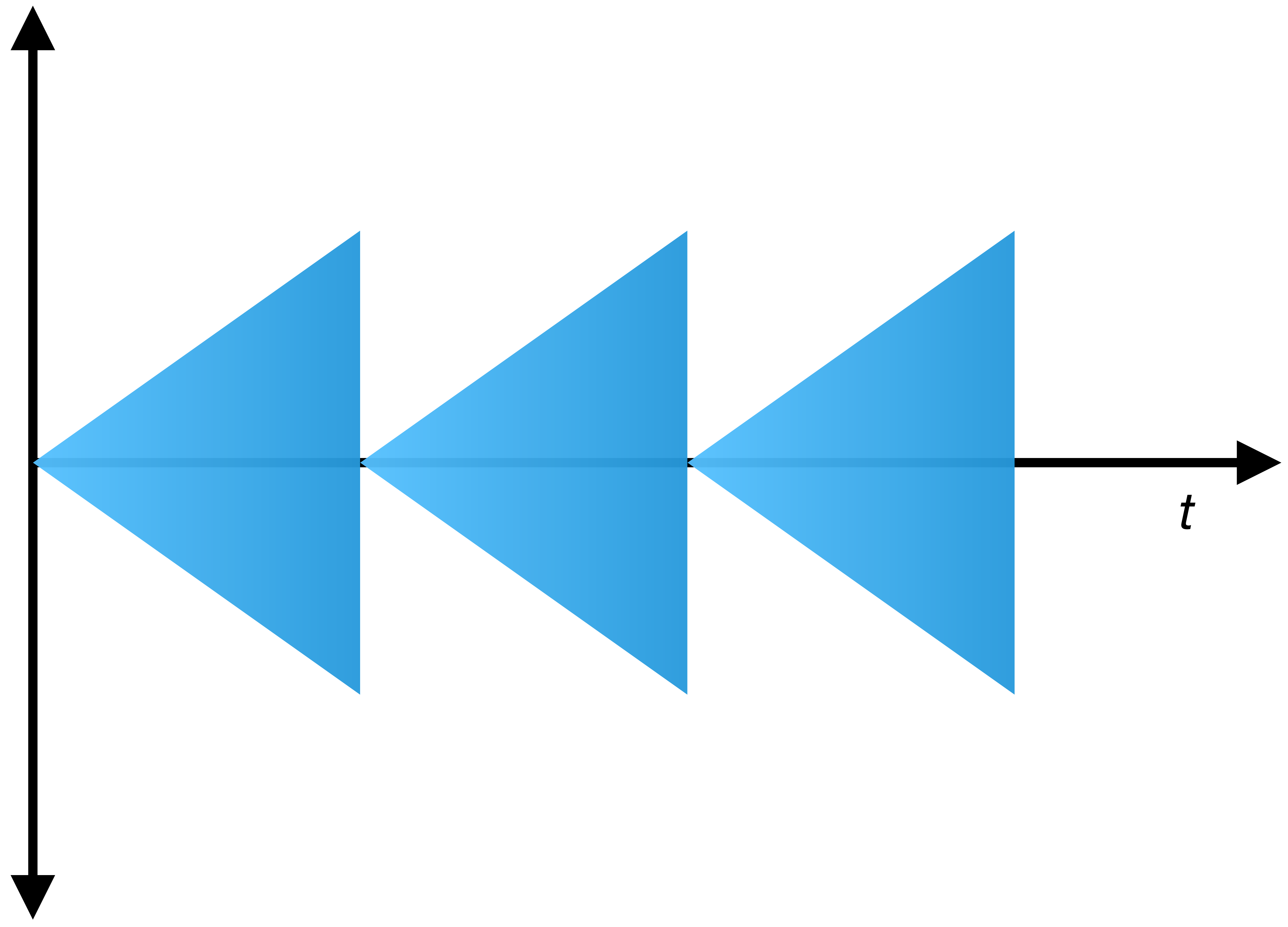
[1] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In Proceedings of the Fourth international Workshop on Computing Education Research (ICER ’08). Association for Computing Machinery, New York, NY, USA, 113–124. DOI:https://doi.org/10.1145/1404520.1404532

TASK DESIGN

- Must carefully restrict tasks!
- People will get stuck on irrelevant things
- Decide how much help to provide
- Ideally: scope task to focus on the variable of interest
- *Constrain the task as much as possible.*

DECOMPOSING TASKS





Monolithic task

Subtasks

DATA COLLECTION

- Think-aloud
- Audio recordings
- Videos
- Screen capture
- Eye tracking
- Post-study survey

- Take lots of notes!, including timestamps! You do not want to watch the videos.
- Include a clock on the screen.

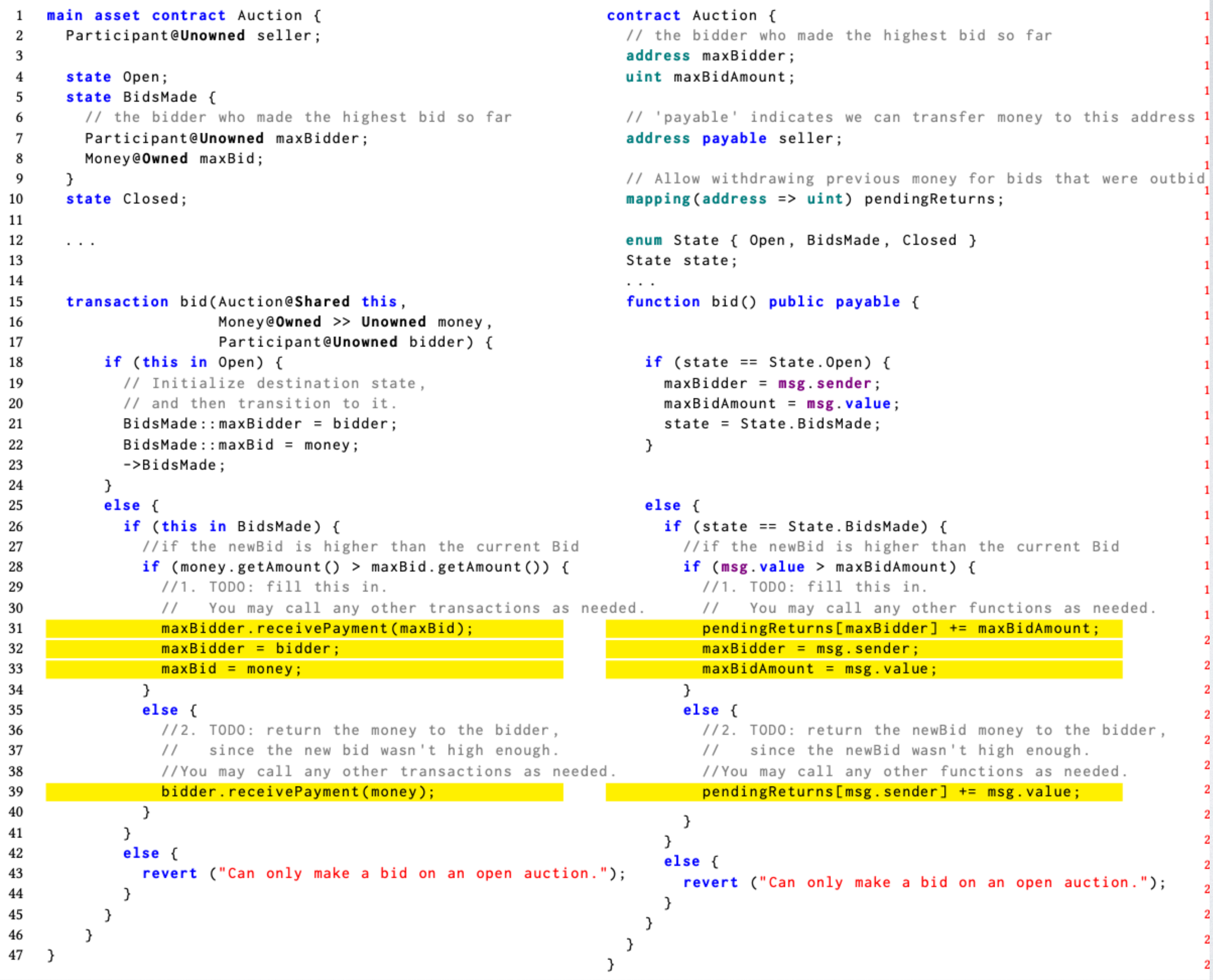
THINK-ALOUD

- Two varieties: concurrent and retrospective
- "Please keep talking."
- Can't use timing as a dependent variable due to effect of explanations.

TASK CONTEXTS

- Pencil/paper
- Text editor
- IDE

- Compiler?
- Debugger?
- Test cases?



```
main asset contract Auction {
  Participant@Unowned seller;

  state Open;
  state BidsMade {
    // the bidder who made the highest bid so far
    Participant@Unowned maxBidder;
    Money@Owned maxBid;
  }
  state Closed;

  . . .


  transaction bid(Auction@Shared this,
                  Money@Owned >> Unowned money,
                  Participant@Unowned bidder) {
      if (this in Open) {
        // Initialize destination state,
        // and then transition to it.
        BidsMade::maxBidder = bidder;
        BidsMade::maxBid = money;
        ->BidsMade;
      }
      else {
        if (this in BidsMade) {
          //if the newBid is higher than the current Bid
          if (money.getAmount() > maxBid.getAmount()) {
            //1. TODO: fill this in.
            //   You may call any other transactions as needed.
            maxBidder.receivePayment(maxBid);
            maxBidder = bidder;
            maxBid = money;
          }
          else {
            //2. TODO: return the money to the bidder,
            //   since the new bid wasn't high enough.
            //You may call any other transactions as needed.
            bidder.receivePayment(money);
          }
        }
        else {
          revert ("Can only make a bid on an open auction.");
        }
      }
    }
}
```

```
contract Auction {
  // the bidder who made the highest bid so far
  address maxBidder;
  uint maxBidAmount;

  // 'payable' indicates we can transfer money to this address
  address payable seller;

  // Allow withdrawing previous money for bids that were outbid
  mapping(address => uint) pendingReturns;

  enum State { Open, BidsMade, Closed }
  State state;
  . . .
  function bid() public payable {


    if (state == State.Open) {
      maxBidder = msg.sender;
      maxBidAmount = msg.value;
      state = State.BidsMade;
    }


    else {
      if (state == State.BidsMade) {
        //if the newBid is higher than the current Bid
        if (msg.value > maxBidAmount) {
          //1. TODO: fill this in.
          //   You may call any other functions as needed.
          pendingReturns[maxBidder] += maxBidAmount;
          maxBidder = msg.sender;
          maxBidAmount = msg.value;
        }
        else {
          //2. TODO: return the newBid money to the bidder,
          //   since the newBid wasn't high enough.
          //You may call any other functions as needed.
          pendingReturns[msg.sender] += msg.value;

        }
      }
      else {
        revert ("Can only make a bid on an open auction.");
      }
    }
  }
}
```