Memory Diagrams

When tracing code by hand, it's helpful to draw a picture to keep track of variables, methods, and objects. Memory diagrams represent the state of a program at a particular moment in time.

Manager: **Alex Wang**

Recorder:

Presenter:

Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Describe primitive values and references in a memory diagram.
- Draw memory diagrams that have variables, arrays and objects.
- Summarize differences between variables, arrays, and objects.

Process Skill Goals

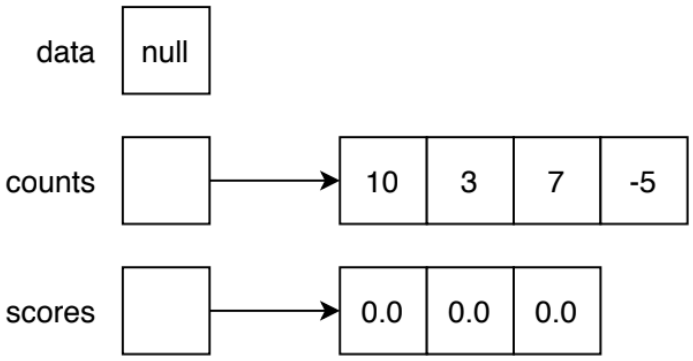
During the activity, students should make progress toward:

- Leveraging prior knowledge and experience of other students. (Teamwork)

Model 1 Arrays

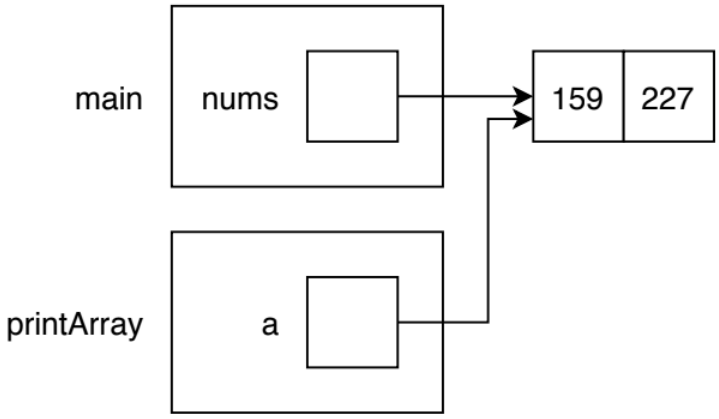
An array variable stores a ***reference*** to an array object. We draw references as arrows, because they "point" to other memory locations.

```
int[] data = null;

int[] counts = {10, 3, 7, -5};

double[] scores = new double[3];
```



When passing an array to a method, only the reference is copied:

```
public static void printArray(int[] a) {
        System.out.print("{" + a[0]);
        for (int i = 1; i < a.length; i++) {
                System.out.print(", "  + a[i]);
        }
        System.out.println("}");
}


public static void main(String[] args) {
        int[] nums = {159, 227};
        printArray(nums);
}
```



Questions (15 min) Start time:

1. What is the length of each array?

a) Counts? **4**

b) Scores? **3**

c) Nums? **2**

d) A? **2**

2. Looking at both diagrams above:

a) How many array variables were declared? **5**

b) How many array objects were created? **3**

3. Based on the top diagram, what is different about the variable named data?

It's not initialized (has the value "null").

4. Based on counts and scores, describe two ways that array objects can be created. How are these two ways different from each other?

Arrays can be created with the curly bracket syntax, which defines the values in the array at creation, or with the square bracket syntax, which does not define the values in the array.

5. If the printArray method were to modify the array contents, would that change be visible in the main method? Explain your reasoning.

Yes, because the array passed into printArray refers to the same memory location as the array in the main method.

6. Draw (or describe) a diagram of the following source code:

```
int[] data = {1, 2, 3};

int[] copy = data;
```

The item data in heap memory refers to an array in stack memory filled with the three ints 1, 2, and 3. The item copy in heap memory refers to the same array in stack memory as data.

7. (Optional) Paste the contents of *Arrays.java* into Java Visualizer. What differences do you notice between the diagram in Java Visualizer and those in Model 1?

n/a

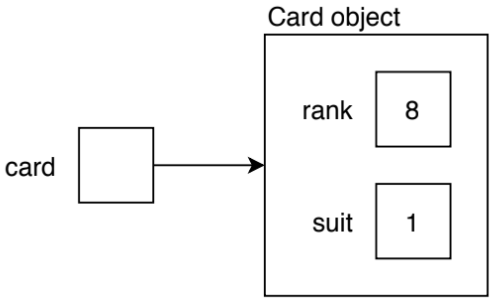
Model 2 Objects

Consider the definition for a playing card:

```
public class Card {
    private int rank;          // 1=Ace, ..., 11=Jack, 12=Queen, 13=King
    private int suit;          // 0=Clubs, 1=Diamonds, 2=Hearts, 3=Spades

    public Card(int rank, int suit) {
        this.rank = rank;
        this.suit = suit;
    }
}
```

Here is a memory diagram of a Card object:

```
Card card = new Card(8, 1);
```



Questions (15 min) Start time:

8. Which card (i.e., "the _____of _____") is represented in the diagram?

The 8 of diamonds

9. In one line of code, show how you would construct the "4 of Clubs".

Card myCard = new Card(4, 0);

10. What is the difference between lowercase card and uppercase Card? Explain in a few sentences how these concepts are illustrated in the diagram.

Lowercase card refers to the variable defined in card (the left side of the diagram), while uppercase Card refers to the class Card (an instance of which is on the right side of the diagram).

11. How are arrays and objects similar? How are arrays and objects different? Explain your answer in terms of how they are drawn in memory diagrams.

Arrays and objects are similar because they are both reference types. However, an array of objects will have two references while an object will only have one reference. (i.e., two arrows vs one arrow)

12. Draw (or describe) a diagram of the following source code:

```
Card card = null;
```

The item card in heap memory has no reference (is "null").

13. Draw (or describe) a diagram of the following source code:

```
Card card = new Card(5, 2);

Card copy = card;
```

The item card in heap memory has a reference to a Card object in stack memory with the instance variables rank and suit set to 5 and 2, respectively. The item copy in heap memory refers to that same Card object in stack memory.

14. (Optional) Paste the contents of *Card.java* into Java Visualizer. What differences do you notice between the diagram in Java Visualizer and those in Model 2?

n/a

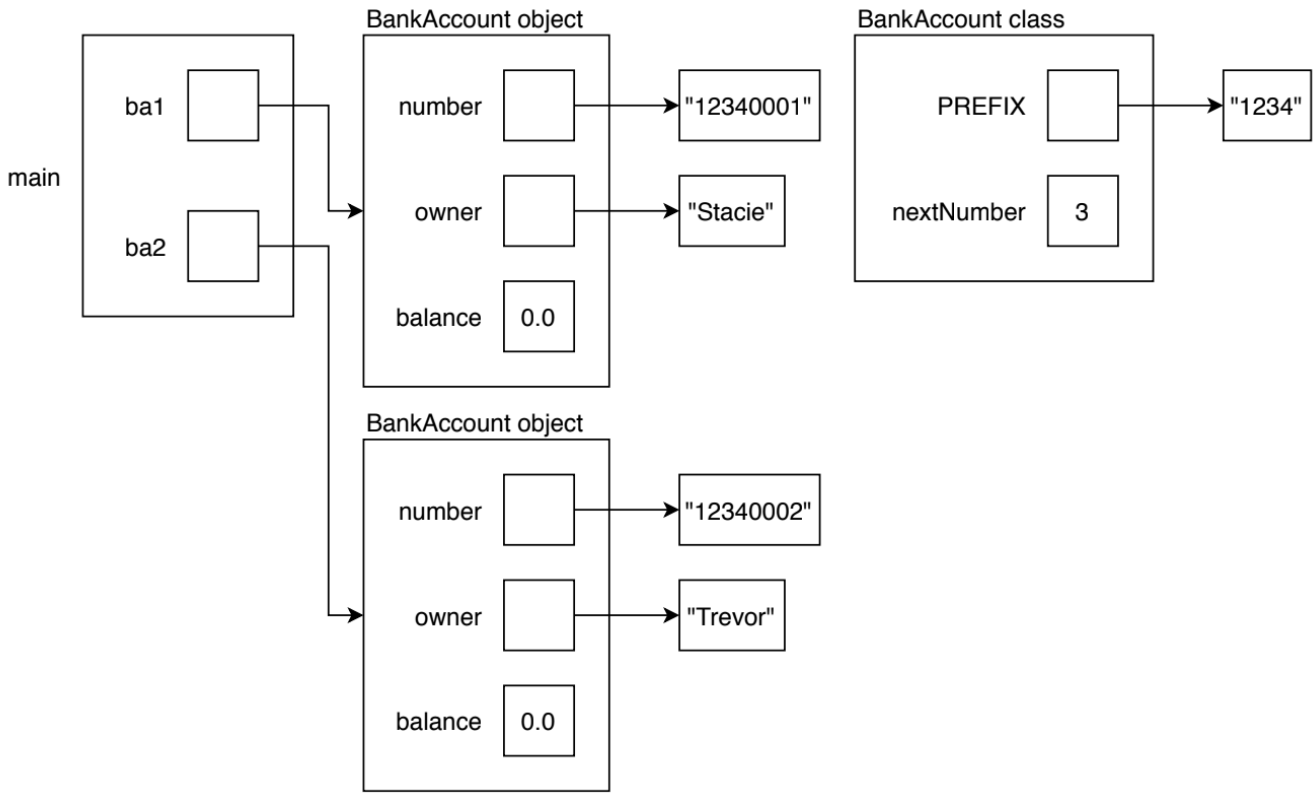
Model 3 Static Variables

Consider the definition for a bank account:

```
public class BankAccount {
    private static final String PREFIX = "1234";
    private static int nextNumber = 1;

    private String number;
    private String owner;
    private double balance;

    public BankAccount(String owner) {
        this.number = PREFIX + String.format("%04d", nextNumber);
        this.owner = owner;
        nextNumber++;
    }
}
```

Here is a memory diagram of two BankAccount objects:

```
public static void main(String[] args) {
    BankAccount ba1 = new BankAccount("Stacie");
    BankAccount ba2 = new BankAccount("Trevor");
}
```



Questions (15 min) Start time:

15. Based on the source code and memory diagram:

a) How many BankAccount variables were declared?

2

b) How many BankAccount objects were created?

2

16. How many instances of each variable are in memory?

a) PREFIX **1**

b) nextNumber **1**

c) Number **2**

d) Owner **2**

e) Balance **2**

17. What is the difference between static and non-static variables of a class? Explain your answer in terms of the diagram.

Static variables of a class belong to the class. The number of static variables is always one per class. Non static variables of a class belong to class instances. The number of non-static variables is equal to the number of class instances in memory.

18. Why are all the strings shown in separate boxes as opposed to being written inside of the variable boxes?

They are reference types, not primitives.

19. How would you modify the memory diagram if the following line were added at the end of the main method?

```
BankAccount ba3 = ba2;
```

Another item in the 'main' box would have an arrow pointing to the ba2 BankAccount object box.

20. (Optional) Paste the contents of *BankAccount.java* into Java Visualizer. What differences do you notice between the diagram in Java Visualizer and those in Model 3?

n/a