Activity 12: Arrays of Objects

With arrays and objects, you can represent pretty much any type of data. It's not only possible to have arrays of objects, but also objects of arrays, objects of objects, arrays of arrays, arrays of objects of arrays, and so forth.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the differences when instantiating an array and an object.
- Rewrite a for loop (over an array) using an enhanced for loop.
- Use enhanced for loops to construct and search arrays of objects.

Process Skill Goals

During the activity, students should make progress toward:

- Developing algorithms for constructing and searching arrays. (Problem Solving)

Facilitation Notes

The first two objectives apply to Model 1. Students should understand the differences in syntax (i.e., using square brackets vs parentheses) as well as what happens behind the scenes (i.e., `new` invokes a constructor for objects but not arrays).

Keep an eye on what students write down for #4. If their answer is superficial, challenge them to be more specific. Report out on #8, and ask students why `i` is not an appropriate variable name for this loop (`i` typically stands for index or iteration).

This activity makes use of a simple Card class. Be sure to take a look at the source code before facilitating. It stores both the rank and suit as integers, and it does not perform any validation. #9 in Model 2 asks students to implement a simple default value replacement. This question can optionally be extended by having students call a setter method instead.

#11 and #15 are the most important questions. Make sure students have enough time to develop the algorithms, and if possible, write the complete source code. Consider reporting out after #11 to bring the class back in sync, since there is so much time allocated for Model 2.

Model 1 Hand of Cards

Creating an array of objects is typically a 3-step process:

1. Declare the array

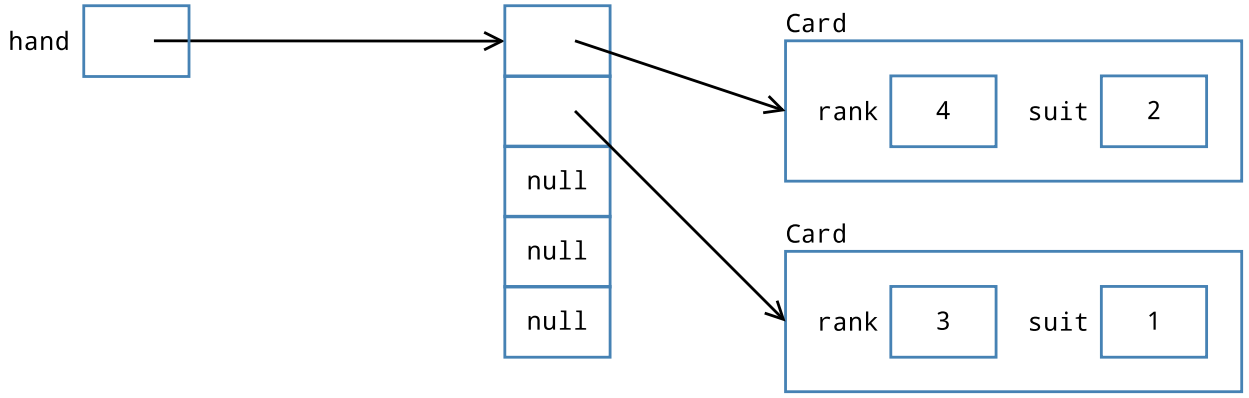
```
Card[] hand;
```

2. Instantiate the array

```
hand = new Card[5];
```

3. Instantiate each object

```
hand[0] = new Card(4, 2);
hand[1] = new Card(3, 1);
```



Questions (20 min)

Start time: _____

1. What is the type of the local variable `hand`? What is the value of `hand` *before* step 2? What is the value of `hand` *after* step 2?

The variable `hand` is an array of `Card` objects. Before step 2, it's uninitialized (i.e., you can't read its value). After step 2, its value is the memory location of the first array element.

2. When you create an array (e.g., `new Card[5]`) what is the initial value of each element?

The initial values are automatically set to null (for reference types). For arrays of integers, it's 0; for doubles, it's 0.0; for booleans, it's `false`; for characters, it's the unicode character `'\u0000'`.

3. When you construct a new object (e.g., `new Card(4, 2)`) what are the initial values of its attributes (e.g., `this.rank`)?

It depends on the constructor. For the `Card` class, the attributes `rank` and `suit` are initialized from the parameters. If there is no constructor, Java automatically initializes attributes to zero.

The `new` operator requests a memory location to store an array or object. Java automatically determines how much memory is needed and initializes the contents of the corresponding memory cells to zero. That's why array elements and object attributes have default values, whereas local variables (not allocated with `new`) must be initialized before they are used.

4. Describe in your own words what the following statement does. Be sure to explain how the random part works.

```
hand[(int) (Math.random() * hand.length)] = null;
```

`Math.random()` returns a value in the range [0, 1). Multiplying that value by `hand.length` and then casting it to an integer gives a value in the range 0..5 inclusive. So this statement randomly sets one of the `Card` references to `null`.

5. What is the result of running the loop below? What is the purpose of the nested if-statement?

```
for (int i = 0; i < hand.length; i++) {
    if (hand[i] != null) {
        int suit = hand[i].getSuit();
        System.out.println("The suit of #" + i + " is " + Card.SUITS[suit]);
    }
}
```

The loop prints the suits of all cards in the hand. Because some of the cards are `null`, the if-statement prevents `NullPointerException`.

6. The *enhanced for loop* allows you to iterate the elements of an array. Another name for this structure is the "for each" loop. Rewrite the following example using a standard for loop.

```
String[] days = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
for (String day : days) {
    System.out.println(day + " is a great day!");
}
```

```
for (int i = 0; i < days.length; i++) {
    System.out.println(days[i] + " is a great day!");
}
```

7. In contrast to enhanced for loops, what does a standard for loop typically iterate? Why would it be misleading to name the enhanced for loop variable `i` instead of `day`?

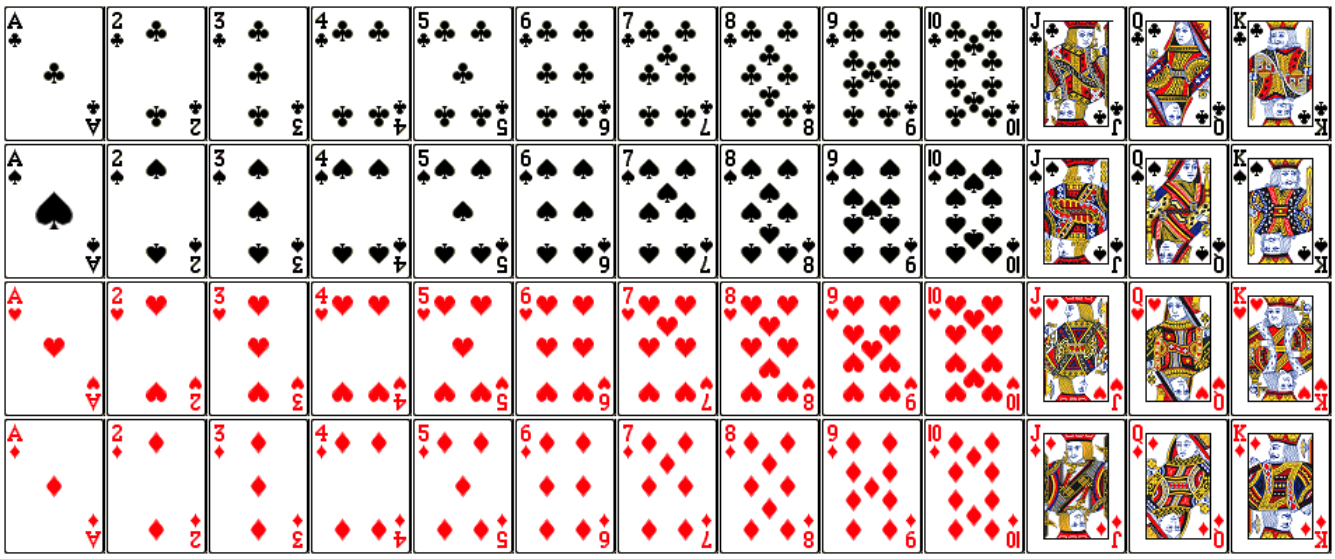
Standard for loops typically iterate indexes; that's why the variable is almost always named `i`.

8. Rewrite the loop in #5 using an enhanced for loop. Use an appropriate variable name for the `Card` object (i.e., not `i`). For simplicity, you may omit the `System.out.println` line.

```
for (Card card : hand) {
    if (card != null) {
        int suit = card.getSuit();
    }
}
```

Model 2 Deck of Cards

There are 52 cards in a standard deck. Each card belongs to one of four suits (Clubs, Spades, Hearts, and Diamonds) and one of 13 ranks (Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, and King). The array below is one-dimensional, but the cards are displayed on four lines for convenience.



Questions (25 min) Start time: _____

9. Implement the following constructor. The class has two attributes: `rank` and `suit`. Make sure the arguments are within range, and if not, replace them with the minimum value.

```
/**
 * Constructs a face card given its rank and suit.
 *
 * @param rank face value (1 = ace, 11 = jack, 12 = queen, 13 = king)
 * @param suit category (0 = clubs, 1 = diamonds, 2 = hearts, 3 = spades)
 */
public Card(int rank, int suit) {

    if (rank < 1 || rank > 13) {
        this.rank = 1;
    } else {
        this.rank = rank;
    }

    if (suit < 0 || suit > 3} {
        this.suit = 0;
    } else {
        this.suit = suit;
    }
}
```

10. In one line of code, declare an array of `int`s named `suits` and initialize its contents to the four possible suits as shown in Model 2.

```
int[] suits = {0, 3, 2, 1};
```

11. Write several lines of code to declare and create an array of 52 `Card` objects. Use nested `for` loops to construct `Card` objects in the order of Model 2. Make use of your `suits` array from the previous question. Discuss with your team how you will keep track of the array index.

```
Card[] cards = new Card[52];
int index = 0;
for (int suit : suits) {
    for (int rank = 1; rank <= 13; rank++) {
        cards[index] = new Card(rank, suit);
        index++;
    }
}
```

12. If you did not use an enhanced for loop in the previous question, go back and simplify your answer. Explain why an enhanced for loop is appropriate for one of the variables (suit or rank) but not the other.

Because the suits are out of order, we rely on the `suits` array to determine how to create the cards. It's easier to pull values out of an array with an enhanced for loop. The rank values simply iterate from 1 to 13 in order, so a standard for loop is appropriate for that task.

13. Describe what the following code does and how it works. (Note: You've come a long way this semester, to be able to understand this example!)

```
public static Card[] sort(Card[] deck) {
    if (deck == null) {
        System.err.println("Missing deck!");
        return null;
    }
    Card[] sorted = new Card[deck.length];
    for (Card card : deck) {
        int index = card.position();       // returns suit * 13 + rank - 1
        sorted[index] = card;
    }
    return sorted;
}
```

This example sorts an array of cards. It first validates the arguments, then it creates a new array of cards and assigns each `Card` reference according to its known position in the deck.

14. Identify examples of the following Java language features in the previous question.

a) variables deck, sorted, card, index

b) decisions if (deck == null)

c) loops for (Card card : deck)

d) methods sort, println, position

e) arrays deck, sorted

f) objects "Missing deck!", card

15. Write a static method named `inDeck` that takes a `Card[]` representing a deck of cards and a `Card` object representing a single card, and that returns `true` if the card is somewhere in the deck of cards. Be sure to use the equals method of the `Card` object to make comparisons.

```
public static boolean inDeck(Card[] deck, Card card) {
    for (Card c : deck) {
        if (c.equals(card)) {
            return true;
        }
    }
    return false;
}
```