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.

Manager:		Recorder:	
Presenter:		Reflector:	

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the difference of 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. It stores both the rank and suit as integers, but it does not perform any validation in the constructor. Consider projecting *Card.java* while you facilitate Model 2. Keep an eye on answers to the first question, in case teams get off track.

#12 and #14 are the most important questions. Make sure students have enough time to work through the algorithms. Consider reporting out after #13 to bring the class back in sync, since there is so much time allocated for Model 2.

Key questions: #4, #12, #14

Source files: *Card.java*, *Main.java*

Model 1 Hand of Cards

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

1. Declare the array

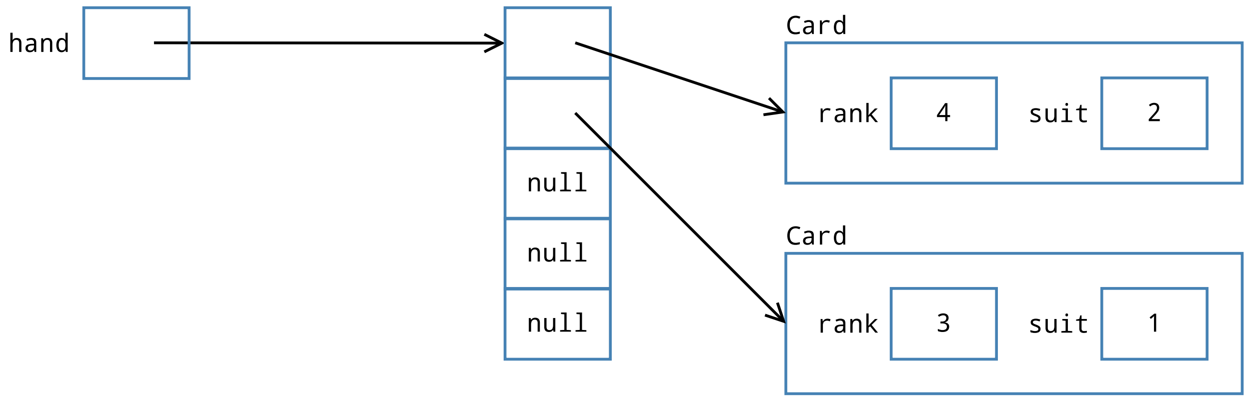
```
Card[] hand;
```

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 (or equivalent).

The `new` operator finds 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 code does. Be sure to explain how the random part works.

```
int index = (int) (Math.random() * hand.length);
hand[index] = 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? Explain why the if-statement is necessary.

```
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 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 usually named `i`.

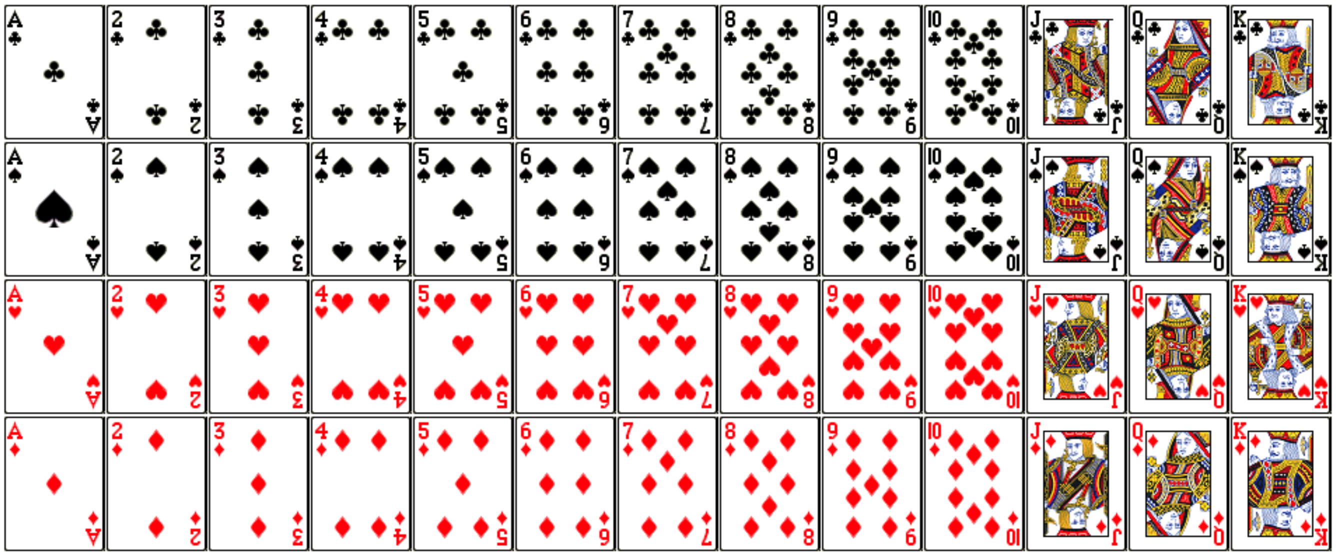
8. Rewrite the loop in #5 using an enhanced for loop. Use an appropriate variable name for the `Card` object (i.e., not `i`). For simplicity, you may omit the `System.out.println` line.

```
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 has one of ***13 ranks*** (1=Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11=Jack, 12=Queen, and 13=King) and one of ***4 suits*** (0=Clubs, 3=Spades, 2=Hearts, and 1=Diamonds). For example, `new Card(12, 2)` would construct the Queen of Hearts.

The following deck is represented by an array of `Card` objects. The array is one-dimensional, but the cards are shown in four rows (because of the paper margins).



Questions (25 min) **Start time:**

9. What is the index (in the array above) of the following cards?

a) Ace of Clubs 0

b) Jack of Clubs 10

c) 2 of Spades 14

d) Queen of Spades 24

e) 7 of Hearts 32

f) King of Diamonds 51

10. Write the following statements using one line of code each.

a) Declare and initialize a `Card` array named `deck` that can hold 52 cards.

```
Card[] deck = new Card[52];
```

b) Construct the Ace of Clubs, and assign it as the first element in `deck`.

```
deck[0] = new Card(1, 0);
```

c) Construct the King of Diamonds, and assign it as the last element in `deck`.

```
deck[51] = new Card(13, 1);    // or deck[deck.length - 1]
```

11. Describe how you could repeat code from the previous question to construct the entire deck of cards (without having to type 52 statements).

Use nested for loops to iterate each possible `rank` and `suit`, construct that card, and assign it to the next index in the `deck` array.

12. Discuss the following code as a team:

```java
int index = 0;
int[] suits = {0, 3, 2, 1};
for (int suit : suits) {
    for (int rank = 1; rank <= 13; rank++) {
        deck[index] = new Card(rank, suit);
        index++;
    }
}
```

a) What is the overall purpose of the code?

It creates a deck of cards in the order shown in Model 2.

b) Why is the `suits` array not just {0, 1, 2, 3}? (See Model 2.)

Because the picture shows the suits in a different order.

c) Why does the code use an enhanced for loop for `suit`?

The code iterates the suits out of order, as specified in the array.

d) Why does the code use a standard for loop for `rank`?

The code iterates the ranks in order; no array is needed for that.

e) What is the purpose of the `index` variable?

To keep track of where to store the next card (not based on rank and suit).

13. Write a 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.

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

14. 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!)

```java
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;
}
```

a) What is the overall purpose of the code?

This example sorts an array of cards.

b) What is the purpose of the if statement?

It avoids `NullPointerException` if `deck` is invalid.

c) Does this method modify the `deck` array? Justify your answer.

No; it creates and returns a new array named `sorted`.

d) How does the `sort` method know where to put each card?

It computes the `position` based on its rank and suit.

15. Identify 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