Arrays

Topics

- One dimensional arrays.
- Array processing.
- “Multi-dimensional” arrays.
**Introduction**

- An array is a named collection of contiguous storage locations holding data of the same type.
- Arrays elements are referenced by position within a structure rather than by name.
- Example: 26 Buttons, A to Z:

```java
JButton a = new JButton("A");
JButton b = new JButton("B");
JButton c = new JButton("C");
;
JButton z = new JButton("Z");

JButton[] buttons = new JButton[26];
for (char ch = 'A'; ch <= 'Z'; ch++) {
    buttons[ch - 'A'] = new JButton("" + ch);
}
```

Without Array

With Array

**An Integer Array**

```java
int[] a = new int[10];

a[2] = 5;


a[6]++
```

Arrays are allocated with the new operator.

Arrays are reference types

Individual elements of the array are accessed using a index (or “subscript”) in square brackets.

Arrays use zero based indexing.
Array Declaration and Initialization

```java
int[] a = new int[5];
a[0] = 1;
a[1] = 2;
a[2] = 3;
a[3] = 4;
a[4] = 5;

int[] a = new int[5];
for (int i = 0; i < a.length; i++) {
    a[i] = i + 1;
}

int[] a = { 1, 2, 3, 4, 5 };
```

Arrays have a `length` attribute

Or use a loop

Or use an initializer list.

Default Values

- Java fields are automatically initialized with default values if no initializer is specified.
  - zero (for numerical types, including char)
  - false (for booleans)
  - null (for reference types)
- Local variables do not have default values.
  - Exception: Elements of arrays are given default values, even when the array is a local variable. The array itself does not have a default value unless it is a field.
- Code is more readable when fields and array elements are explicitly initialized even when it is not necessary.

```java
private int[] a1;
int[] a2 = new int[5];
```
Array Processing with Loops

- Example: Print out all values in an array of integers:

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

- Example: Sum of array values:

```java
public int sum(int[] a) {
    int sum = 0;
    for (int i = 0; i < a.length; i++)
        sum += a[i];
    return sum;
}
```

The for-each Loop (enhanced for loop)

- Works with arrays and collection classes, e.g. ArrayList
- Example: Print array values:

```java
public void printArray(int[] a) {
    for (int n: a)
        System.out.println(n);
}
```

- Example: Sum of array values:

```java
public int sum(int[] a) {
    int sum = 0;
    for (int n: a)
        sum += n;
    return sum;
}
```
Example: Reverse

➢ Reverse the order of the elements in an array:

```java
public void reverse(int[] a) {
    int i = 0;
    int j = a.length - 1;
    while (i < j) {
        // swap
        int tmp = a[i];
        a[i++] = a[j];
        a[j--] = tmp;
    }
}
```

int[] a = {3, 5, 1, 4, 2, 7, 8, 6};

Reverse (2)

➢ Does the reverse method always work?
   - What if the number of elements is odd?
   - What if there is only one element?
   - What if there are zero elements?
Example: Lottery Quick Pick

- Problem: pick 5 random numbers from 1 to 56, without picking any number more than once.

```java
public int[] quickPick() {
    int[] pick = { 0, 0, 0, 0, 0 };  // or new int[5]
    for (int i = 0; i < 5; i++) {
        Random r = new Random();
        while(pick[i] == 0) {
            int n = r.nextInt(56) + 1;
            boolean duplicate = false;
            for (int j = 0; j < i; j++) {
                if (pick[j] == n) {
                    duplicate = true;
                    break;
                }
            }
            if (!duplicate) {
                pick[i] = n;
            }
        }
    return pick;
}
```

Example: Lottery Quick Pick (2)

```java
public int[] quickPick() {
    int[] pick = new int[5];
    int[] numbers = new int[56];
    for (int i = 0; i < 56; i++) {
        numbers[i] = i + 1;
    }
    for (int i = 0; i < 5; i++) {
        Random r = new Random();
        int index = r.nextInt(56 - i);
        pick[i] = numbers[index];
        numbers[index] = numbers[55 - i];
        numbers[55 - i] = pick[i];
    }
    return pick;
}
```

This step is not needed!
Example: A Deck Playing Card Class

- Suppose we want to program a card game.
- A deck of cards may be represented as an array of Card objects.
- We will create a Card class and a Deck class.
- A Card has suit and rank fields, a constructor, and a toString method that returns a string like “Queen of Hearts” or “5 of Clubs”.
- The Deck class should have a constructor and a shuffle method.
- For testing purposes we will also write a method to print out all the cards in the order they appear in the Deck.

```java
public class PlayingCard {
    private String suit;
    private int rank;
    public PlayingCard(String suit, int rank) {
        this.suit = suit;
        this.rank = rank;
    }
    public String toString() {
        String rankString;
        switch(rank) {
            case 1:
                rankString = "Ace";
                break;
            case 11:
                rankString = "Jack";
                break;
            case 12:
                rankString = "Queen";
                break;
            case 13:
                rankString = "King";
                break;
            default:
                rankString = Integer.toString(rank);
                break;
        }
        return rankString + " of " + suit;
    }
}
```
public class Deck {
    private PlayingCard[] cards;
    public Deck() {
        cards = new PlayingCards[52];
        String[] suits = {“Hearts”, “Diamonds”, “Clubs”, “Spades”};
        int index = 0;
        for (String s: suits) {
            for (int rank = 1; rank <= 13; rank++) {
                cards[index++] = new PlayingCard(s, rank);
            }
        }
    }
    public void printCards() {
        for (PlayingCard card: cards) {
            System.out.println(card);
        }
    }
    public void shuffle() {
        Random r = new Random();
        for (int i = 0; i < 52; i++) {
            // Randomly pick a card (that hasn’t already been picked)
            // and place it at position i
            int index = r.nextInt(52-i) + i;
            PlayingCard temp = cards[i];
            cards[i] = cards[index];
            cards[index] = temp;
        }
    }
}

Common Errors with Uninitialized References

String hello;
char c = hello.charAt(0);
int[] a;
a[0] = 3;
String[] strings;
strings[0] = “hello”;
String[] names = new String[5];
String name = names[0];
int n = name.length();
Multidimensional Arrays

- It is often useful to use a 2-dimensional array or “table” of data.
  - For example, when programming a board game like chess or checkers.
- We would like to be able to create tables with cells that can be referenced by row and column:

```
0   1   2
0
1   5
2
3
```

Multidimensional Arrays (2)

- In Java, we use an array of arrays:
  ```java
  int[][] table = new int[4][3];
  ```
- Table is an array with 4 elements
- Each element is an array of 3 int’s
  - e.g. the first element of the first array is `table[0][0]`
Multidimensional Arrays (3)

- The arrays can be of different sizes:

```java
int[][] pyramid = {{1}, {2,3}, {4,5,6}, {7,8,9,10}};
```

![Diagram of pyramid array]