Lecture 07

Single-Dimensional Arrays (after a bit of method practice)
Challenge: Area of hexagon method

- Write a program (that uses a method) to compute the area of a regular hexagon given the following formula

\[ A = \frac{3 \sqrt{3}}{2} a^2 \]
Single-Dimensional Arrays

- Fixed-size, sequential collection of elements (of the same type)
- Can be used to store *anything* - of the same type
- way better than declaring num1, num2, …, num500, …
Declaring arrays

- Syntax:
  
  ```
  elementType[] arrayRefVar;
  ```

- Example:
  
  ```
  int[] xPositions;
  ```
Using Arrays

• **Note:** Simply declaring an array does not actually create any space for it. It’s a placeholder in memory, but value is still *null*.

• In order to use it, you still need to create the object.
Creating an array object

• Syntax:

    arrayRefVar = new elementType[arraySize];

• Or combine with the declaration:

    elementType[] arrayRefVar = new elementType[arraySize];

    int[] xPositions = new int[3000];
Adding items to array

- **Syntax:**

  ```
  arrayRefVar[index] = value;
  ```

- **Example:**

  ```
  xPositions[0] = 0;
  xPositions[1] = 10;
  xPositions[2] = 20;
  xPositions[3] = 30;
  xPositions[4] = 40;
  xPositions[5] = 50;
  ```
Array length

- Length is fixed and must be declared when created.
- Can’t get bigger or smaller
- length can be accessed by `arrayVar.length`
- Default values are given when created. For numbers, it’s `0`, for booleans, its `false`
Accessing array items

- Index based, just like the strings we’ve seen
- Syntax:

  `xPositions[37]`
• Declare an array of 10 integers and assign each element in the array the numbers 1 - 10 in order
public class ArrayPractice {

    public static void main(String[] args) {
        int[] myNumbers = new int[10];

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

        for (int i = 0; i < myNumbers.length; i++){
            System.out.println(myNumbers[i]);
        }
    }
}
Shortcut declaration

double[] myNumbers = {1.9, 2.9, 3.4, 3.5};

Note: There’s no ‘new’ keyword. Can’t break up this line.
Practice

• Write a method that takes an array of doubles as a parameter and returns the highest value from the array
public static double highestValue(double[] myList){
    double highest = myList[0];

    for (int i = 0; i < myList.length; i++){
        if (myList[i] > highest){
            highest = myList[i];
        }
    }

    return highest;
}
Even shorter for loop

• Useful when you just want to loop through the whole array

• Syntax:

```java
for (elementType element: arrayRefVar) {
    // Process the element
}
```

• Example:

```java
for (double e: myList) {
    System.out.println(e);
}
```
Practice: Pick a card, any card

- Write a program that creates a 52 card deck, shuffles the deck, and then chooses 3 cards
public class DeckOfCards {

    public static void main(String[] args) {
        int[] deck = new int[52];
        char[] suits = {'♠', '♥', '♦', '♣'};
        String[] ranks = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};

        // Insert initial values for cards
        for (int i = 0; i < deck.length; i++) {
            deck[i] = i;
        }

        // Shuffle deck
        for (int i = 0; i < deck.length; i++) {
            int randomIndex = (int)(Math.random() * deck.length);

            int temp = deck[i];
            deck[i] = deck[randomIndex];
            deck[randomIndex] = temp;
        }

        // Pick 4 cards
        for (int i = 0; i < 3; i++) {
            char suit = suits[deck[i] / 13];
            String rank = ranks[deck[i] % 13];

            System.out.println(rank + " of " + suit);
        }
    }
}
To be continued...