class Lecture5 {

    "Arrays"

}
Arrays

An array stores a large collection of data which is of the same type.

```java
... // assume the size variable exists above
T[] A = new T[size];
// this creates an array of T type, referenced by A
...
```

- $T$ can be any data type.
- This statement comprises two parts:
  - Declaring a reference
  - Creating an array
Variable Declaration for Arrays

- In the left-hand side, it is a declaration for an array variable, which does not allocate real space for the array.
- In reality, this variable occupies only a certain space for the reference to an array.\(^1\)
- If a reference variable does not refer to an array, the value of the variable is null.\(^2\)
- In this case, you cannot assign elements to this array variable unless the array object has already been created.

\(^1\)Recall the stack and the heap in the memory layout.
\(^2\)Moreover, this holds for any reference variable. For example, the Scanner type.
Creating A Real Array

- All arrays of Java are objects.
- As seen before, the new operator returns the memory address of that object.
  - Recall that the type of reference variables must be compatible to that of the array object.
- The variable size must be a positive integer for the number of elements.
- Note that the size of an array cannot be changed after the array is created.³

³Alternatively, you may try the class ArrayList, which is more useful in practice.
The array is allocated **contiguously** in the memory.

All arrays are **zero-based indexing**. (Why?)


---

Same in C, C++, python, Javascript, and more.
The elements of arrays are initialized once created.

- By default, every element is assigned as follows:
  - 0 for all numeric primitive data types
  - \u0000 for char type
  - false for boolean type
- An array can also be initialized by enumerating all the elements without using the new operator.
- For example,

```java
int[] A = {1, 2, 3};
```
Processing Arrays

When processing array elements, we often use for loops.

- Recall that arrays are objects.
- They have an attribute called length which records the size of the arrays.
  - For example, use A.length to get the size of A.
- Since the size of the array is known, it is natural to use a for loop to manipulate with the array.
Initialization of arrays by a Scanner object

```java
... // let x be an integer array with a certain size
for (int i = 0; i < A.length; ++i) {
    A[i] = input.nextInt();
}
...
```

Initialization of arrays by random numbers

```java
... for (int i = 0; i < A.length; ++i) {
    A[i] = (int) (Math.random() * 10);
}
...```
Display of array elements

```
...  for (int i = 0; i < A.length; ++i) {
      System.out.printf("%3d", A[i]);
  }
...
```

Sum of array elements

```
...  int sum = 0;
  for (int i = 0; i < A.length; ++i) {
      sum += A[i];
  }
...  
```
Extreme value problems of array elements

```
...  
  int max = A[0];  
  int min = A[0];  
  for (int i = 1; i < A.length; ++i) {
      if (max < A[i]) max = A[i];
      if (min > A[i]) min = A[i];
  }
...  
```

• How about the location of the extreme values?
Shuffling over array elements

```java
...  
for (int i = 0; i < A.length; ++i) {
    // choose j randomly
    int j = (int) (Math.random() * A.length);
    // swap
    int tmp = A[i];
    A[i] = A[j];
    A[j] = tmp;
}
...  
```

- How to swap values of two variables without `tmp`?
- However, this naive algorithm is biased.\(^5\)

\(^5\)See https://blog.codinghorror.com/the-danger-of-naivete/
Deck of Cards

Write a program which picks first 5 cards at random from a deck of 52 cards.

- 4 suits: Spade, Heart, Diamond, Club
- 13 ranks: 3, ..., 10, J, Q, K, A, 2
- Label 52 cards by 0, 1, ..., 51
- Shuffle the numbers
- Deal the first 5 cards
String[] suits = {"Spade", "Heart", "Diamond", "Club"};
String[] ranks = {
    "3", "4", "5", "6", "7",
    "8", "9", "10", "J", "Q", "K",
    "A", "2"};

int size = 52;
int[] deck = new int[size];
for (int i = 0; i < deck.length; i++)
    deck[i] = i;

// shuffle over deck; correct version
for (int i = 0; i < size - 1; i++) {
    int j = (int) (Math.random() * (size - i)) + i;
    int z = deck[i];
    deck[i] = deck[j];
    deck[j] = z;
}

for (int i = 0; i < 5; i++) {
    String suit = suits[deck[i] / 13];
    String rank = ranks[deck[i] % 13];
    System.out.printf("%8s%3s\n", suit, rank);
}
Cloning Arrays

• In practice, one might duplicate an array for some reason.
• One could attempt to use the assignment statement (\(=\)), for example,

```java
// assume A is an array
T[] A = {...};
T[] B = A; // shallow copy; you don't have a new array
```

• However, this is impossible to make two distinct arrays.
• Recall that the array variables are simply references to the arrays in the heap.
Moreover, all the reference variables share this property!

For example,
• Use a loop to copy individual elements one by one.

```java
... int[] A = {2, 1, 3, 5, 10};
int[] B = new int[A.length];
// deep copy
for (int i = 0; i < A.length; ++i) {
    B[i] = A[i];
}
...
```

• Alternatively, you may use the `arraycopy` method in the `System` class.

```java
... int[] A = {2, 1, 3, 5, 10};
int[] B = new int[A.length];
System.arraycopy(A, 0, B, 0, A.length);
... 
```
• A for-each loop is designed to iterate over a collection of objects, such as arrays and other data structures, in strictly sequential fashion, from start to finish.

• For example,

```java
... T[] A = {...}; // assume some T-type array
for (T element: A) {
    // body
}
...
```

• Note that the type `T` should be compatible to the element type of `A`.

---

6Beginning with JDK5. Now we have JDK9.
• Not only is the syntax streamlined, but it also prevents boundary errors.
Short Introduction to Data Structures

• A data structure is a particular way of organizing data in a program so that it can be used efficiently.

• Data structures can implement one or more particular abstract data types (ADT), which specify the operations that can be performed on a data structure and the computational complexity of those operations.

• In comparison, a data structure is a concrete implementation of the specification provided by some ADT.

• Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks.\(^7\)

---

\(^7\)See http://bigocheatsheet.com/.
Common Operations on Data

• A specific data structure is chosen in one problem.
• Then the operations are implemented accordingly.
• The **Arrays** class contains useful methods for common array operations such as *sorting* and *searching*.
• For example,

```java
import java.util.Arrays;

... 
int[] A = {5, 2, 8};
Arrays.sort(A); // sort the whole array

cchar[] B = {'A', 'r', 't', 'h', 'u', 'r'};
Arrays.sort(B, 1, 3); // sort the array partially
...
```
Selection Sort

...  

// selection sort
for (int i = 0; i < A.length; i++) {
    int k = i; // the position of min starting from i
    for (int j = i + 1; j < A.length; j++) {
        if (A[k] > A[j])
            k = j;
    }
    // swap(A[i], A[k])
    int tmp = A[k];
    A[k] = A[i];
    A[i] = tmp;
}

...  

• Time complexity: $O(n^2)$
• You can find more sorting algorithms.\(^8\)

\(^8\)See http://visualgo.net/.
Linear Search

Write a program which searches for the index associated with the key.

- For convenience, assume that there is no duplicate key.
- The linear search approach compares the key with each element in the array sequentially.
// assume A is an array
// linear search
for (int i = 0; i < A.length; i++) {
    if (A[i] == key) {
        System.out.printf("%3d", i);
    }
}

• Time complexity: $O(n)$
Alternative: Binary Search

- Time complexity: $O(\log n)$
- Overall time complexity (sorting + searching): still $O(\log n)$?
...
Beyond 1-Dimensional Arrays

• 2D or high-dimensional arrays are widely used.
  • For example, a colorful image is represented by three 2D arrays (R, G, B).
• We can create a 2D $T$-type array with 4 rows and 3 columns as follows:

```java
... int rowSize = 4; // row size
int colSize = 3; // column size
T[][] x = new T[rowSize][colSize];
...
```
Case (c) shows that we can create a 2D array by enumeration.
Reality

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

```java
... int[][] A = {{1, 2, 3}, {4, 5}, {6}};

    // conventional for loop
    for (int i = 0; i < A.length; i++) {
        for (int j = 0; j < A[i].length; j++)
            System.out.printf("%2d", A[i][j]);
        System.out.println();
    }

    // for–each loop
    for (int[] B : A) {
        for (int item : B)
            System.out.printf("%2d", item);
        System.out.println();
    }

    ...
```

---

9Thanks to a lively discussion on January 31, 2016.
Exercise: Matrix Multiplication

Write a program which determines $C = A \times B$ for the input matrices $A_{m \times n}$ and $B_{n \times q}$ for $m, n, q \in \mathbb{N}$.

- You may use the formula

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

where $a_{ik}, i = 1, 2, \ldots, m$ is a shorthand for $A$ and $b_{kj}, j = 1, 2, \ldots, q$ for $B$.

- Time complexity: $O(n^3)$ (Why?)