Arrays

Dr. H. B. Prajapati

Associate Professor
Department of Information Technology
Dharmsinh Desai University

9 July '20

Core Java Technology

Table of contents

Why are Arrays Important in Programing?

- Suppose we have daily expenses in Rs.
- We want to find out average daily expense
- We might think to create 30 variables (expense1, expense2, ..., expense30) and then assign daily expenses to them.
- We sum these 30 variables and calculate average by dividing the sum by 30.
- But this approach of programming has many limitations:
  - What if we do not have any expense on some days?
  - What if total days are less than 30 or more than 30?
  - What if we want to find out average daily expense of a quarter or of half year or of a whole year?
- Therefore, we need efficient and organized approach to deal with a collection of variables.

Why are Arrays Important in Programing?

- Array is a data structure that provides a collection of variables of same data type
  - Array allows access of each variable of the collection using the same name.
  - Array uses index, integer number, to refer each variable of the collection
- Thus, instead of creating 30 separate variables, we create a single variable expense, which is an array of double type (for expenses in Rs.).
- If we want to increase or decrease the number of days, it can easily be done by modifying the size of array in array declaration, at a single place.
- Java supports to decide the size of array at runtime.

Outline of Presentation

Outline of Presentation

Declaring an Array

- To use an array, first we need to declare it.
- The syntax to declare an array is as follows:

```
dataType[] arrayName; //preferred style
//OR
dataType arrayName[];
```

- The following code declares an array of double data type.

```
double[] expense; //preferred style
//OR
double expense[];
```

Creating Array

- Java array is an object, so declaration does not allocate any space in memory to hold elements.
- We cannot assign value to uninitialized/uncreated array.
- We can use new operator to create the array with the following syntax:

```
arrayName = new dataType[arraySize];
```

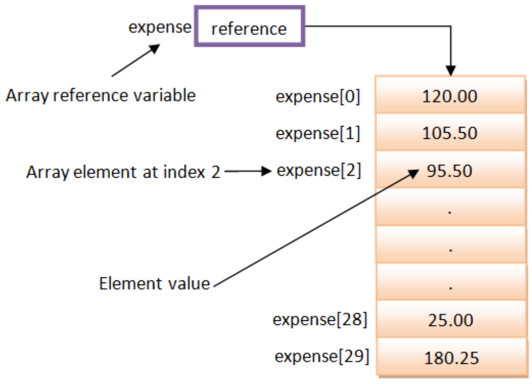
- We can combine declaration and creation in one statement:

```
dataType[] arrayName = new dataType[arraySize];
//OR
dataType arrayName[] = new dataType[arraySize];
```

- The following code creates an array of type double of size 30 elements:

```
double[] expense = new double[30];
```

Array Representation



Array Declaration vs Array Creation

- At the time of declaring an array, the size of array is not required.
- At the time of creating an array, the size of array is required.
- After the array is created, its size cannot be changed.
- But, we can create a new array of new size and can assign it to the same reference variable.

Outline of Presentation

Initializing an Array

- When an array is created, all the elements are assigned the default value depending upon the data type of the array.
  - 0 for the numeric primitive data type variables
  - '\u0000' for char variables
  - false for boolean variables
  - null for object variables
- Array is accessed using index
  - Array indices are from 0 to arraySize-1.
  - For example, for expense array, expense[29] will return the last element in the array.
  - We can also assign a new value to an array element. For example, expense[0]=120.00
  - When invalid index (other than 0 to arraySize-1) is specified, exception will occur. For example, expense[30] will throw exception.

Size of Array and its use in Initialization

- The size of an array is denoted by arrayObject.length.
- After an array is created, the length data field is assigned a value that indicates the number of elements in the created array. Imp. Note: length is a field, not a method–length().
- For example, we can initialize expense array in the following way:

```java
for(int i=0;i<expense.length;i++)
   expense[i] = 0;
```

Array Initialization at the time of Creation

- Java provides a shorthand notation to create an array object and initialize it at the same time:

```java
double[] expense =   {120.0, ..., 180.25};
```

- We put a list of values (literals) enclosed, separated by comma, in curly braces.
- We do not require to use new operator.

Initialization of Array at Runtime, Slide - I

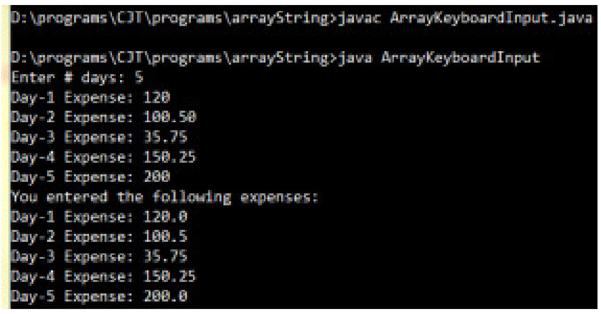
- We can create an array at runtime and can assign values at runtime using Scanner class.

```java
import java.util.*;
class ArrayKeyboardInput{
   public static void main(String[] args){
      Scanner input=new Scanner(System.in);
      double[] expense;
      int days;
      System.out.print("Enter # days: ");
      days = input.nextInt();
      expense = new double[days];
      for(int i=0;i<days;i++){
         System.out.print("Day-"+(i+1)+"
            Expense: ");
         expense[i] = input.nextDouble();
      }
```

Initialization of Array at Runtime, Slide - II

```java
      System.out.println("You entered the
         following expenses: ");
      for(int i=0;i<days;i++){
         System.out.println("Day-"+(i+1)+"
            Expense: "+expense[i]);
      }
   }
}
```

Initialization of Array at Runtime, Slide - III



```
D:\programs\CJT\programs\arrayString>javac ArrayKeyboardInput.java

D:\programs\CJT\programs\arrayString>java ArrayKeyboardInput
Enter # days: 5
Day-1 Expense: 120
Day-2 Expense: 100.50
Day-3 Expense: 35.75
Day-4 Expense: 150.25
Day-5 Expense: 200
You entered the following expenses:
Day-1 Expense: 120.0
Day-2 Expense: 100.5
Day-3 Expense: 35.75
Day-4 Expense: 150.25
Day-5 Expense: 200.0
```

Processing Array Elements

- For processing array elements, we can use for loop because
  - All the elements of the array are of the same data type
  - The size of the array is known.
- For example, if in our daily expense, we forgot to add expense of tea, Rs. 10, then it can easily be added into expense of each day using only two statements.

```java
for(int i=0;i<expense.length;i++)
   expense[i] += 10.0;
```

Outline of Presentation

Program: Assigning Grades to Students, Slide - I

- We read scores of students from the keyboard and get the best sore.
- We then assign grades based on the following scheme:
  - Grade A if score is >= best - 10;
  - Grade B if score is >= best - 20;
  - Grade C if score is >= best - 30;
  - Grade D if score is >= best - 40;
  - Grade F otherwise;
- We write a solution having the following steps:
  - The program prompts the user to enter the total number of students
  - Then, the program prompts the user to enter score for each student
  - Then, the program calculates and assigns grades
  - Finally, the program displays scores and grades.

Program: Assigning Grades to Students, Slide - II

```java
import java.util.*;
class AssigningGrade{
   public static void main(String[] args){
      int numberOfStudents;
      int[] score;
      char[] grade;
      int best = 0;
      Scanner input=new Scanner(System.in);
      System.out.print("Enter # students: ");
      numberOfStudents = input.nextInt();

      //Create arrays for score and grade
      score=new int[numberOfStudents];
      grade=new char[numberOfStudents];

      //Read scores and find the best score
      for(int i=0;i<score.length;i++){
```

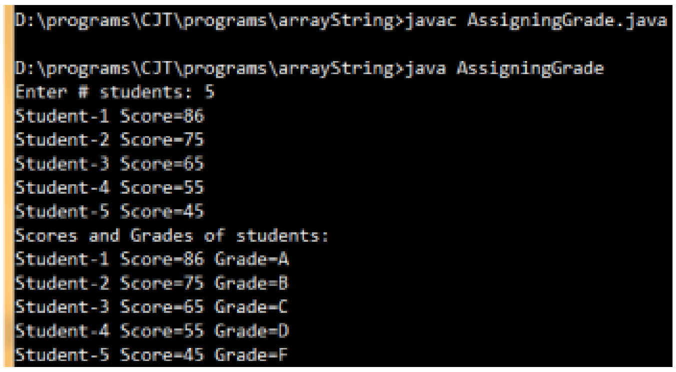
Program: Assigning Grades to Students, Slide - III

```java
         System.out.print("Student-"+(i+1)+"
            Score=");
         score[i]=input.nextInt();
         if(score[i]>best)
            best = score[i];
      }
      //Assign Grades
      for(int i=0;i<score.length;i++){
         if(score[i] >= best -10)
            grade[i] = 'A';
         else if(score[i] >= best -20)
            grade[i] = 'B';
         else if(score[i] >= best -30)
            grade[i] = 'C';
         else if(score[i] >= best -40)
            grade[i] = 'D';
         else
            grade[i] = 'F';
      }
```

Program: Assigning Grades to Students, Slide - IV

```java
      //Display Grades
      System.out.println("Scores and Grades of
         students:");
      for(int i=0;i<score.length;i++)
         System.out.println("Student-"+(i+1)+"
            Score="+
            score[i]+" Grade="+grade[i]);
   }
}
```

Program: Assigning Grades to Students, Slide - V



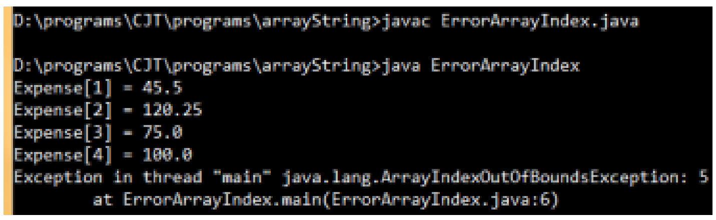
```
D:\programs\CJT\programs\arrayString>javac AssigningGrade.java

D:\programs\CJT\programs\arrayString>java AssigningGrade
Enter # students: 5
Student-1 Score=86
Student-2 Score=75
Student-3 Score=65
Student-4 Score=55
Student-5 Score=45
Scores and Grades of students:
Student-1 Score=86 Grade=A
Student-2 Score=75 Grade=B
Student-3 Score=65 Grade=C
Student-4 Score=55 Grade=D
Student-5 Score=45 Grade=F
```

Error While Processing Array, Slide - I

- While processing array, a common error is accessing an array out of its bounds.
- Valid indices are from 0 to arrayObject.length-1

```
class ErrorArrayIndex{
   public static void main(String[] args){
      double[] expense={120.0, 45.50, 120.25,
         75, 100.0};
      for(int i=1;i<=expense.length;i++){
         System.out.println("Expense["+i+"] =
            "+expense[i]);
      }
   }
}
```

Error While Processing Array, Slide - II



```
D:\programs\CJT\programs\arrayString>javac ErrorArrayIndex.java

D:\programs\CJT\programs\arrayString>java ErrorArrayIndex
Expense[1] = 45.5
Expense[2] = 120.25
Expense[3] = 75.0
Expense[4] = 100.0
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at ErrorArrayIndex.main(ErrorArrayIndex.java:6)
```

- Valid indices for expense array of size 5 is 0 to 4.
- But we are accessing the array for indices 1 to 5.
- Therefore, we get an exception called ArrayIndexOutOfBoundsException for index value 5.

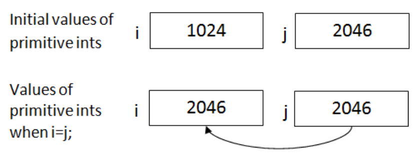
Outline of Presentation

Copying Arrays

- Sometimes, we need to duplicate an array or a part of an array.
- For ordinary variables of primitive types, we copy value by assigning a variable to another variable. That is, $i = j;$, where value of j is copied into i.
- We need to understand whether such assignment of array variables create copy of the array.

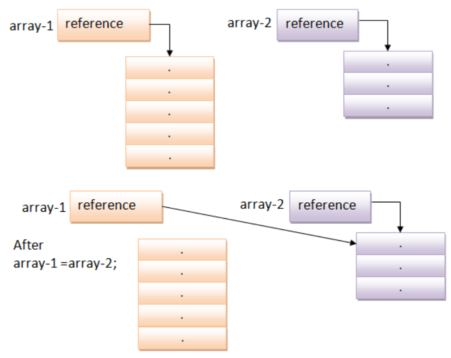
Copying Array versus Copying Primitive Variable

- The following shows what happens when a primitive variable is assigned to another primitive variable of same type.


Initial values of primitive ints
i
1024
j
2046
Values of primitive ints when i=j;
i
2046
j
2046

- For primitive variables, assignment works as copy.

Copying Array versus Copying Primitive Variable



- For array, even of primitive type, assignment does not work as copy of elements as an array is an object in Java.

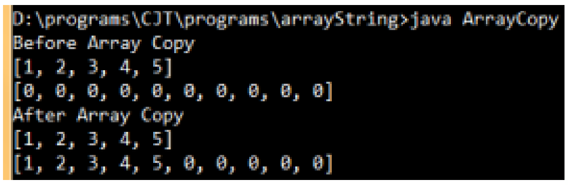
Program: Copy Arrray using Loop, Slide - I

```
class ArrayCopy{
   public static void main(String[] args){
      int[] sourceArray = {1,2,3,4,5};
      int[] targetArray = new
         int[sourceArray.length*2];

      System.out.println("Before Array Copy");
      printArray(sourceArray);
      printArray(targetArray);

      for(int i=0;i<sourceArray.length;i++){
         targetArray[i]=sourceArray[i];
      }
      System.out.println("After Array Copy");
      printArray(sourceArray);
      printArray(targetArray);
   }
   public static void printArray(int[] arr){
```

Program: Copy Arrray using Loop, Slide - II

```
      System.out.print("[");
      for(int i=0;i<arr.length;i++){
         System.out.print(arr[i]);
         if(i != arr.length-1)
            System.out.print(", ");
         else
            System.out.print("]");
      }
      System.out.println();
   }
}
```

Program: Copy Arrray using Loop, Slide - III



```
D:\programs\CJT\programs\arrayString>java ArrayCopy
Before Array Copy
[1, 2, 3, 4, 5]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
After Array Copy
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 0, 0, 0, 0, 0]
```

- In this program, we need to write a loop as per our requirement.
- Java provides a method arraycopy() in System class for copying a source array into destination array.

```
System.arraycopy(sourceArray, src_start,
   targetArray, target_start, noOfElements);
```

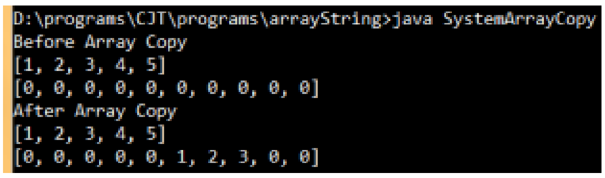
Program: Copy Arrray using System.arraycopy(), Slide - I

```
class SystemArrayCopy{
   public static void main(String[] args){
      int[] sourceArray = {1,2,3,4,5};
      int[] targetArray = new
         int[sourceArray.length*2];

      System.out.println("Before Array Copy");
      printArray(sourceArray);
      printArray(targetArray);

      System.arraycopy(sourceArray, 0,
         targetArray, 5, 3);
      System.out.println("After Array Copy");
      printArray(sourceArray);
      printArray(targetArray);
   }
   public static void printArray(int[] arr){
      System.out.print("[");
```

Program: Copy Arrray using System.arraycopy(), Slide - II

```
      for(int i=0;i<arr.length;i++){
         System.out.print(arr[i]);
         if(i != arr.length-1)
            System.out.print(", ");
         else
            System.out.print("]");
      }
      System.out.println();
   }
}
```

Program: Copy Arrray using System.arraycopy(), Slide - III



```
D:\programs\CJT\programs\arrayString>java SystemArrayCopy
Before Array Copy
[1, 2, 3, 4, 5]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
After Array Copy
[1, 2, 3, 4, 5]
[0, 0, 0, 0, 0, 1, 2, 3, 0, 0]
```

- While using arraycopy(), we need to allocate memory space for the target array.
- The target array must already be created using new operator.
- The arraycopy() method can copy any type of array elements (primitive or object)

Outline of Presentation

Passing Anonymous Array

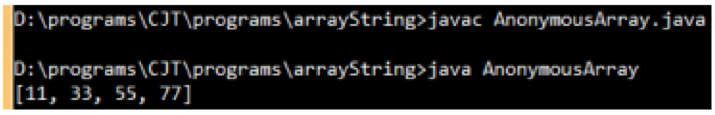
- We have passed a literal of primitive type to methods. E.g., sqrt(16); where 16 is a value we pass without creating a variable.
- For array, if we want to pass a list of values to some method, we need to create an anonymous array object.
- An anonymous object is an object without any name. That is, when it is created, it is not assigned to any reference variable.
- The syntax to create an anonymous array object is as follows:

```
new dataType[]{literal1, literal2, ...,
   literaln};
```

Program: Passing Anonymous Array, Slide - I

```
class AnonymousArray{
   public static void main(String[] args){
      printArray(new int[]{11, 33, 55, 77});
   }
   public static void printArray(int[] arr){
      System.out.print("[");
      for(int i=0;i<arr.length;i++){
         System.out.print(arr[i]);
         if(i != arr.length-1)
            System.out.print(", ");
         else
            System.out.print("]");
      }
      System.out.println();
   }
}
```

Program: Passing Anonymous Array, Slide - II



```
D:\programs\CJT\programs\arrayString>javac AnonymousArray.java

D:\programs\CJT\programs\arrayString>java AnonymousArray
[11, 33, 55, 77]
```

Anonymous Array Object in main() method

- Earlier, we have passed command-line arguments to our program.
- These command-line arguments are passed as an anonymous array of String objects to the main() method.
- We can call the main() method of one class from another class.

```
class Calc{
   public static void main(String[] args){
      //Implementation of command line
         calcualtor
   }
}

class UseCalc{
   public static void main(String[] args){
      Calc.main(new String[]{"10","+","30"});
   }
}
```

Outline of Presentation

Declare and Initialize Array of Objects

- There is a difference between creating elements of an array of primitive types and an array of object types.
- For an array of objects, we need one additional step of calling constructor for each element.

```java
//Declare a reference for an array of Student
Student[] students;
//Create an array of references
students = new Student[5];
//Create Student objects
for (int i=0;i<students.length;i++)
    students[i] = new Student();
```

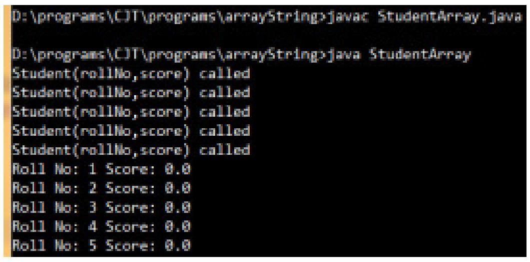
Program: Student Array, Slide - I

```java
class Student{
   private int rollNo;
   private double score;
   public Student(int rollNo, double score){
      System.out.println("Student(rollNo,score) called");
      this.rollNo=rollNo;
      this.score=score;
   }
   public String toString(){
      return "Roll No: "+rollNo+" Score: "+score;
   }
}
public class StudentArray{
   public static void main(String[] args){
      Student[] students;
      students = new Student[10];
```

Program: Student Array, Slide - II

```java
      for(int i=0;i<students.length/2;i++){
         students[i] = new Student(i+1, 0);
      }
      for(int i=0;i<students.length/2;i++){
         System.out.println(students[i]);
      }
   }
}
```

Program: Student Array, Slide - III



```
D:\programs\CJT\programs\arrayString>javac StudentArray.java

D:\programs\CJT\programs\arrayString>java StudentArray
Student(rollNo,score) called
Student(rollNo,score) called
Student(rollNo,score) called
Student(rollNo,score) called
Student(rollNo,score) called
Roll No: 1 Score: 0.0
Roll No: 2 Score: 0.0
Roll No: 3 Score: 0.0
Roll No: 4 Score: 0.0
Roll No: 5 Score: 0.0
```

- We can observe that we declare a reference of an array of Student–called array reference and created an array of 10 Student references.
- But, the constructor gets called only 5 times. That is because we created 5 Student objects using new operator.

Random Shuffling

- Random shuffling is useful in many applications, e.g. Games in computers or phones.
- How can we shuffle entities (e.g., Shuffling playing cards?)
- If we have a bundle of 30 answer books (i.e., answerBooks of type AnswerBook[ ]), we can shuffle them using the following code:

```java
int j;
AnswerBook temp;
for(int i=0;i<answerBooks.length;i++){
   j = (int)(Math.random()*answerBooks.length);
   temp = answerBooks[i];
   answerBooks[i] = answerBooks[j];
   answerBooks[j] = temp;
}
```

Program: Random Shuffling of Playing Cards, Slide - I

- Suppose we want to randomly shuffle a deck of 52 playing cards
  - The card numbers are: A(Ace), 2, 3, 4, 5, 6, 7, 8, 9, 10, J (Jack-young prince), Q(Queen), K (King)
  - The suites are: Spade (♠), Heart (♡), Diamond (♢), Club (♣)
- We use the following encoding for suites of cards:
  - S for Spade (♠)
  - H for Heart (♡)
  - D for Diamond (♢)
  - C for Club (♣)
- Note: In GUI based application, we can use Unicode characters for these card suites.

Program: Random Shuffling of Playing Cards, Slide - II

```java
class Card{
   private int cardNo;
   private char suite;
   public Card(char suite, int cardNo){
      this.cardNo=cardNo;
      this.suite=suite;
   }
   public String toString(){
      return ""+suite+" "+getCardFace();
   }
   private String getCardFace(){
      String face="";
      if(cardNo>=2 && cardNo<=10)
         face = ""+cardNo;
      else if(cardNo==1)
         face = ""+"A";
      else if(cardNo==11)
         face = ""+"J";
```

Program: Random Shuffling of Playing Cards, Slide - III

```java
      else if(cardNo==12)
         face = ""+"Q";
      else if(cardNo==13)
         face = ""+"K";
      else
         face=" Error";
      return face;
   }
}
public class DeckOfCards{
   public static void main(String[] args){
      Card[] deck=null;
      deck = getCardDeck();
      System.out.println("Cards before
         shuffling: ");
      printCardDeck(deck);
      shuffleCards(deck);
      System.out.println("Cards after
         shuffling: ");
```

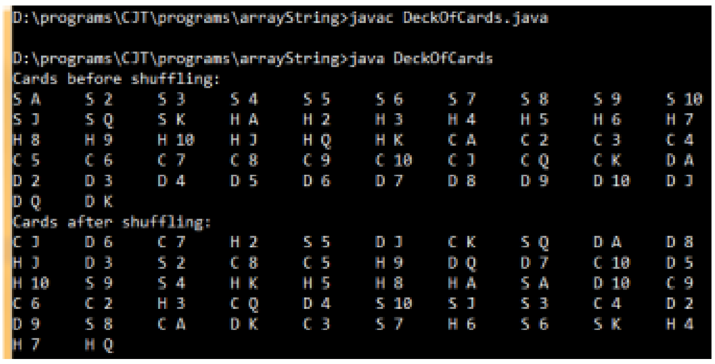
Program: Random Shuffling of Playing Cards, Slide - IV

```java
      printCardDeck(deck);
   }
   public static Card[] getCardDeck(){
      Card[] deck = new Card[52];
      for(int i=0;i<52;i++){
         char suite=' ';
         switch(i/13){
            case 0: suite = 'S'; break;
            case 1: suite = 'H'; break;
            case 2: suite = 'C'; break;
            case 3: suite = 'D'; break;
         }
         deck[i] = new Card(suite,(i%13)+1);
      }
      return deck;
   }
   public static void printCardDeck(Card[]
      deck){
      for(int i=0;i<deck.length;i++){
```

Program: Random Shuffling of Playing Cards, Slide - V

```java
         System.out.print(deck[i]+"\t");
      }
      System.out.println();
   }
   public static void shuffleCards(Card[]
      deck){
      int j;
      Card temp;
      for(int i=0;i<deck.length;i++){
         j = (int)(Math.random()*deck.length);
         temp = deck[i];
         deck[i] = deck[j];
         deck[j] = temp;
      }
   }
}
```

Program: Random Shuffling of Playing Cards, Slide - VI



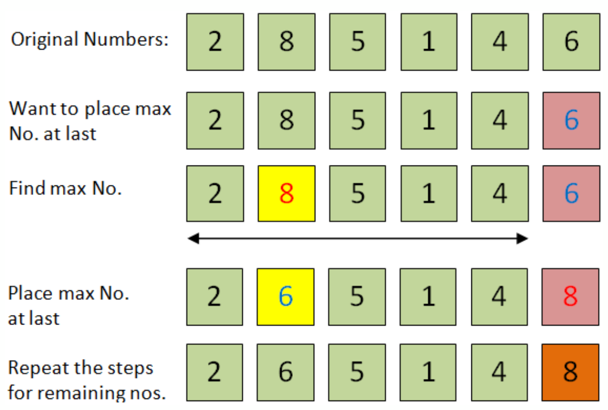
```
D:\programs\CJT\programs\arrayString>javac DeckOfCards.java

D:\programs\CJT\programs\arrayString>java DeckOfCards
Cards before shuffling:
S A     S 2     S 3     S 4     S 5     S 6     S 7     S 8     S 9     S 10
S J     S Q     S K     H A     H 2     H 3     H 4     H 5     H 6     H 7
H 8     H 9     H 10    H J     H Q     H K     C A     C 2     C 3     C 4
C 5     C 6     C 7     C 8     C 9     C 10    C J     C Q     C K     D A
D 2     D 3     D 4     D 5     D 6     D 7     D 8     D 9     D 10    D J
D Q     D K
Cards after shuffling:
C J     D 6     C 7     H 2     S 5     D J     C K     S Q     D A     D 8
H J     D 3     S 2     C 8     C 5     H 9     D Q     D 7     C 10    D 5
H 10    S 9     S 4     H K     H 5     H 8     H A     S A     D 10    C 9
C 6     C 2     H 3     C Q     D 4     S 10    S J     S 3     C 4     D 2
D 9     S 8     C A     D K     C 3     S 7     H 6     S 6     S K     H 4
H 7     H Q
```

Outline of Presentation

Sorting Arrays

- Sorting is a common task in computer programming. E.g., sorting grades.
- Sorting can be done using many algorithms.
- We study selection sort.
- We study how the selection sort arranges numbers in an ascending order:
  - Selection sort finds the largest number in the list and places at last.
  - Then, from the remaining numbers (i.e., excluding the last), the algorithm finds the largest number and places it last (i.e., second last)
  - These two steps (find max and place at last) are repeated until the remaining list contains a single number.

Selection Sort



Selection Sort Algorithm

```
for(int i=array.length-1;i>=1;i--){
   select the largest number in array[1..i];
   swap the largest with array[i], if
      necessary;
   //Apply next iteration on array[1..i-1]
}
```

Program: Selection Sort, Slide - I

```
class SelectionSort{
   public static void main(String[] args){
      double[] spi = {7.5, 8.25, 6.75, 8.5,
         7.0, 7.75};
      System.out.println("SPI of students
         before sorting ");
      printArray(spi);
      sortWithSelectionSort(spi);
      System.out.println("SPI of students
         after sorting ");
      printArray(spi);
   }
   public static void printArray(double[] arr){
      for(int i=0;i<arr.length;i++)
         System.out.println(arr[i]);
   }
   public static void
      sortWithSelectionSort(double[] arr){
```

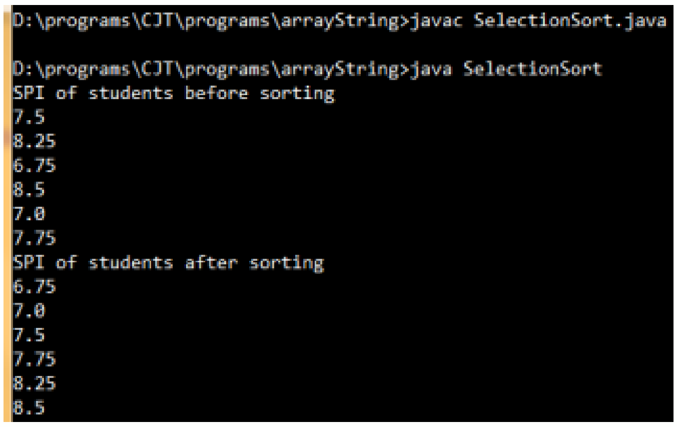
Program: Selection Sort, Slide - II

```
      double currentMax;
      int currentMaxIndex;
      for(int i=arr.length-1;i>=1;i--){
         //Find the maximum from arr[0..i]
         currentMax = arr[i];
         currentMaxIndex = i;
         for(int j=i-1; j>=0;j--){
            if(currentMax < arr[j]){
               currentMax = arr[j];
               currentMaxIndex = j;
            }
         }
         //swap arr[i] with
            arr[currentMaxIndex] if required
         if(currentMaxIndex != i){
            arr[currentMaxIndex] = arr[i];
            arr[i] = currentMax;
         }

```

Program: Selection Sort, Slide - III

```
      }
   }
}
```

Program: Selection Sort, Slide - IV



```
D:\programs\CJT\programs\arrayString>javac SelectionSort.java

D:\programs\CJT\programs\arrayString>java SelectionSort
SPI of students before sorting
7.5
8.25
6.75
8.5
7.0
7.75
SPI of students after sorting
6.75
7.0
7.5
7.75
8.25
8.5
```

Outline of Presentation

Searching Arrays

- Searching is the process of looking for a particular element in the array.
- For example, a particular name is included in an array of names.
- Searching is a common task in computer programming.
- We learn to implement linear search and binary search algorithms using Java.

Linear Search Approach

- An item that we want to search in the array is called key.
- The linear search method starts searching the key from the first element and continues the search
  - until the key matches an element in the array
  - or the array list is exhausted without a match found.
- If a match is found, the algorithm returns the index of the element in the array that matches the given key.
- The algorithm can be represented as follows:

```
for(int i=0;i<list.length;i++){
   if(key==list[i])
      return i;
}
return -1;
```

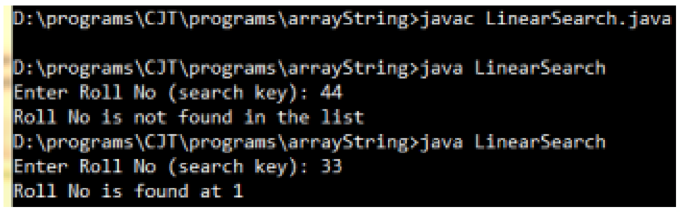
Program: Linear Search, Slide - I

```
import java.util.*;
class LinearSearch{
   public static void main(String[] args){
      Scanner input=new Scanner(System.in);
      int[] absentNos = {11, 33, 55, 77, 99,
         111};
      int key;
      int index;
      System.out.print("Enter Roll No (search
         key): ");
      key = input.nextInt();
      index = linearSearch(absentNos, key);
      if(index != -1)
         System.out.print("Roll No is found at
            "+index);
      else
         System.out.print("Roll No is not
            found in the list");
```

Program: Linear Search, Slide - II

```
   }
   public static int linearSearch(int[] list,
      int key){
      for(int i=0;i<list.length;i++){
         if(key==list[i])
            return i;
      }
      return -1;
   }
}
```

Program: Linear Search, Slide - III



```
D:\programs\CJT\programs\arrayString>javac LinearSearch.java

D:\programs\CJT\programs\arrayString>java LinearSearch
Enter Roll No (search key): 44
Roll No is not found in the list
D:\programs\CJT\programs\arrayString>java LinearSearch
Enter Roll No (search key): 33
Roll No is found at 1
```

- The element in the array can be present at any location.
- On average the algorithm needs to compare half of the elements in an array.
- The execution time of the linear search increases linearly as the number of elements of the array increases.
- Linear search is inefficient for a large array.

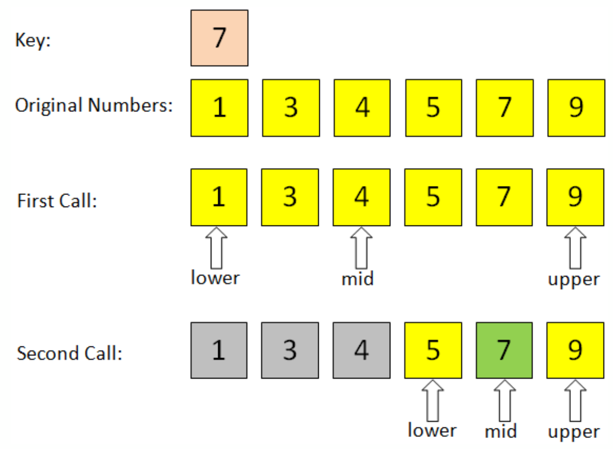
Binary Search Approach

- Binary search is another approach to search an array.
- For binary search, the array elements need to be ordered, ascending or descending.
- We study the algorithm for binary search for which array elements are available in ascending order.

Binary Search Algorithm

- The binary search algorithm first compares the key with the middle element of the array.
- Three cases are possible:
  1. If the key is matching with the middle element, then the search ends and the index of the middle element is returned.
  2. If the key is lower than the middle element, then further search is made in the first half of the array.
  3. If the key is higher than the middle element, then further search is made in the second half of the array

Steps of Binary Search Algorithm



Binary Search Algorithm

```java
private static int binarySearch(int[] list,
    int key, int lower, int upper){
    if(lower>upper)
    // list exhausted w/o match
        return -1;
    int mid = (lower+upper)/2;
    //match at the middle
    if(key==list[mid])
        return mid;
    else if (key<list[mid])
    //search in the first half
        return binarySearch(list, key, lower,
            mid-1);
    else if(key>list[mid])
    //search in the second half
        return binarySearch(list, key, mid+1,
            upper);
}
```

Program: Binary Search, Sllide - I

```java
import java.util.*;
class BinarySearch{
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        int[] absentNos = {11, 33, 55, 77, 99,
            111};
        int key;
        int index;
        System.out.print("Enter Roll No (search
            key): ");
        key = input.nextInt();
        index = binarySearch(absentNos, key);
        if(index != -1)
            System.out.print("Roll No is found at
                "+index);
        else
            System.out.print("Roll No is not
                found in the list");
```

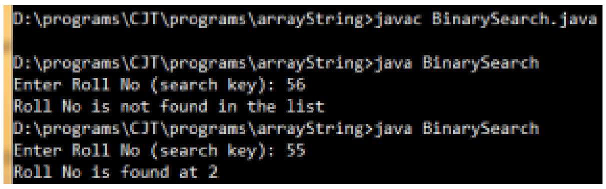
Program: Binary Search, Sllide - II

```
   }
   public static int binarySearch(int[] list,
      int key){
      int lower=0;
      int upper=list.length-1;
      return binarySearch(list, key, lower,
         upper);
   }
   private static int binarySearch(int[] list,
      int key, int lower, int upper){
      if(lower>upper)
         return -1;
      int mid = (lower+upper)/2;
      if(key==list[mid])
         return mid;
      else if (key<list[mid])
         return binarySearch(list, key, lower,
            mid-1);
      else if(key>list[mid])
```

Program: Binary Search, Sllide - III

```
         return binarySearch(list, key, mid+1,
            upper);
      return -1;
   }
}
```

Program: Binary Search, Sllide - IV



```
D:\programs\CJT\programs\arrayString>javac BinarySearch.java

D:\programs\CJT\programs\arrayString>java BinarySearch
Enter Roll No (search key): 56
Roll No is not found in the list
D:\programs\CJT\programs\arrayString>java BinarySearch
Enter Roll No (search key): 55
Roll No is found at 2
```

- The binary search eliminates half of the elements after each comparison.
- For simplicity, let us assume n is power of 2.
- After the first comparison, n/2 elements are left for further search.
- After the second comparison, (n/2)/2 elements are left for further search.
- Thus, after kth comparison, $n/2^k$ elements are left for further search.

Program: Binary Search, Sllide - V

- When $k = log_2 n$, only one element is left in the array.
- Therefore, in the worst case, we need $log_2 n + 1$ comparisons for finding a key in the sorted array.
- For an array of 1024 (2^{10}) elements:
  - Binary search requires 11 comparisons in the worst case
  - Linear search requires 1024 comparisons in the worst case.

Outline of Presentation

Multi-dimensional Array

- One-dimensional (1D) array models a linear collection of elements.
- To represent a table or matrix, we can use two-dimensional (2D) array.
- A 2D array in Java is declared as an array of arrays of objects.
- Similar way, we can represent a 3D array or multi-dimensional array.

Declaring a 2D Array

- The syntax to declare a 2D array is as follows:

```
dataType[][] arrayName; //preferred style
//OR
dataType arrayName[][];
```

- The following code declares a 2D array of int data type.

```
int[][] matrix; //preferred style
//OR
int matrix[][];
```

Creating a 2D Array

- Similar to 1D array, we use new operator to create the array with the following syntax:

```
arrayName = new dataType[rowSize][columnSize];
```

- We can combine declaration and creation in one statement:

```
dataType[][] arrayName = new
   dataType[rowSize][columnSize];
//OR
dataType arrayName[][] = new
   dataType[rowSize][columnSize];
```

- The following code creates an array of type int of size 3X4 elements:

```
int[][] matrix = new int[3][4];
```

Initializing 2D array

- If an array is created using new operator, we can assign value to each element by accessing an array element using the following syntax:

```
matrix[1][2] = 5;
```

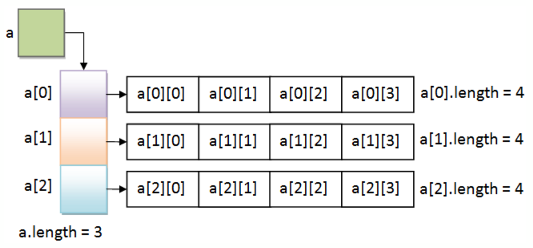
- Java provides a shorthand notation to create an array object and initialize it at the same time:

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

- When we assign literals as values of array elements, we do not need to use new operator.
- We can see that a 2D array is an array of 1D arrays.
- In the array, matrix, there are three rows and four columns.

Lengths of 2D Array

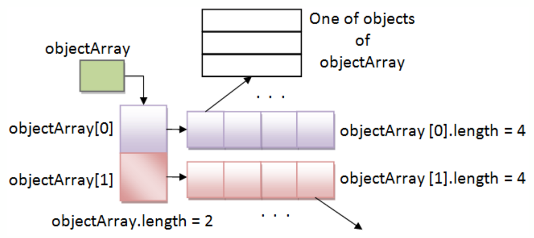
- A 2D array is actually an array in which each element is a one-dimensional array.
- A 2D array of primitive type in Java can be represented as below:


a
a[0]
a[1]
a[2]
a[0][0] a[0][1] a[0][2] a[0][3]
a[1][0] a[1][1] a[1][2] a[1][3]
a[2][0] a[2][1] a[2][2] a[2][3]
a[0].length = 4
a[1].length = 4
a[2].length = 4
a.length = 3

- The number of rows can be obtained using length property of array a and the number of columns can be obtained using length property on a row (e.g., a[0].length gives length of row 0)

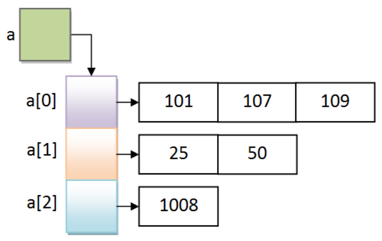
Representation of 2D Array of objects

- A 2D array of objects is represented in different way.
- The following shows an array of objects of size 2 X 4.



Ragged Array

- As each row in two-dimensional array is itself an array, Each row can have different length.
- An array with different lengths in each row is called ragged array or zigzag array.



```
int[][] a = {
 {101, 107, 109},
 {25, 50},
 {1008}
};
```

Program: Matrix Addition, Slide - I

```
import java.util.Scanner;
class MatrixAddition{
   public static void main(String[] args){
      int[][] m1, m2;
      int nRows, nColumns;

      Scanner input=new Scanner(System.in);
      System.out.print("Enter the number of
         rows : ");
      nRows = input.nextInt();
      System.out.print("Enter the number of
         columns : ");
      nColumns = input.nextInt();

      m1 = new int[nRows][nColumns];
      m2 = new int[nRows][nColumns];
      System.out.println("== Enter Matrix-1
         ==");
```

Program: Matrix Addition, Slide - II

```
      inputMatrix(m1);
      System.out.println("== Enter Matrix-2
         ==");
      inputMatrix(m2);

      int[][] result = addMatrix(m1,m2);
      System.out.println("======= Matrix-1
         =======");
      printMatrix(m1);
      System.out.println("======= Matrix-2
         =======");
      printMatrix(m2);
      System.out.println("= Matrix-1 +
         Matrix-2 =");
      printMatrix(result);
   }
   public static void inputMatrix(int[][]
      matrix){
      Scanner input=new Scanner(System.in);
```

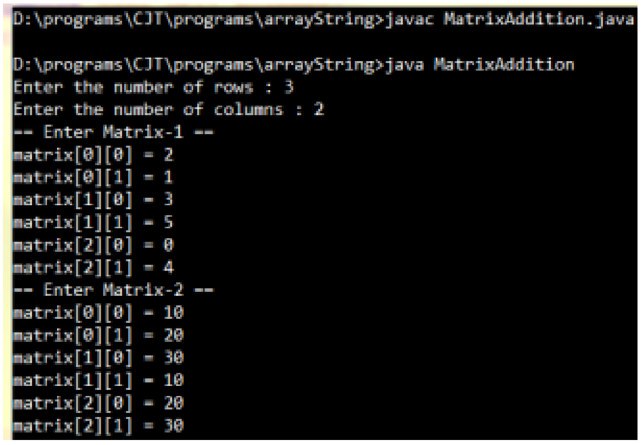
Program: Matrix Addition, Slide - III

```
      for(int i=0;i<matrix.length;i++){
         for(int j=0;j<matrix[0].length;j++){
            System.out.print("matrix["+ i
               +"][" + j + "] = ");
            matrix[i][j]=input.nextInt();
         }
      }
   }
   public static int[][] addMatrix(int[][] m1,
      int[][] m2){
      int[][] result = new
         int[m1.length][m1[0].length];
      for(int i=0;i<m1.length;i++)
         for(int j=0;j<m1[0].length;j++)
            result[i][j] = m1[i][j]+m2[i][j];
      return result;
   }
   public static void printMatrix(int[][]
      matrix){
```

Program: Matrix Addition, Slide - IV

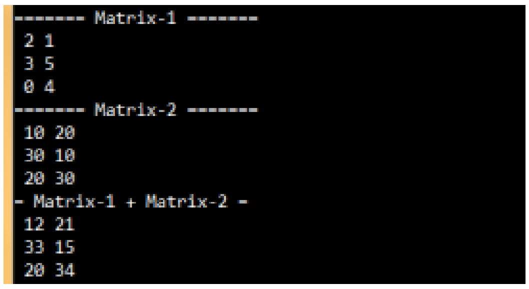
```
      for(int i=0;i<matrix.length;i++){
         for(int j=0;j<matrix[0].length;j++)
            System.out.print(" "+matrix[i][j]);
         System.out.println();
      }
   }
}
```

Program: Matrix Addition, Slide - V



```
D:\programs\CJT\programs\arrayString>javac MatrixAddition.java

D:\programs\CJT\programs\arrayString>java MatrixAddition
Enter the number of rows : 3
Enter the number of columns : 2
-- Enter Matrix-1 --
matrix[0][0] = 2
matrix[0][1] = 1
matrix[1][0] = 3
matrix[1][1] = 5
matrix[2][0] = 0
matrix[2][1] = 4
-- Enter Matrix-2 --
matrix[0][0] = 10
matrix[0][1] = 20
matrix[1][0] = 30
matrix[1][1] = 10
matrix[2][0] = 20
matrix[2][1] = 30
```

Program: Matrix Addition, Slide - VI



```
------- Matrix-1 -------
 2 1
 3 5
 0 4
------- Matrix-2 -------
 10 20
 30 10
 20 30
- Matrix-1 + Matrix-2 -
 12 21
 33 15
 20 34
```

Summary of key terms

- Importance of Arrays
- 1D Array: Declaring, Creating, Initializing, and Processing
- Error while array access
- Copying arrays
- Anonymous array, its uses
- Array of objects, Random Shuffling
- Sorting array-selection sort
- Searching arrays-linear search and binary search
- Multi-dimensional arrays

References

- An Introduction to Java Programming, Y. Daniel Liang, PHI
- An Introduction to Java Programming, Y. Daniel Liang, Eigth Edition, Prentice Hall