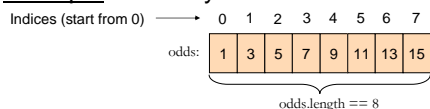


## Arrays

- **Array:** A fixed-length data structure for storing multiple values of the same type

- **Example:** An array of odd numbers:



The type of all elements is int

The value of the element at index 4 is 9: odds[4] == 9

2

## Software 1

### Recitation No. 2 Arrays and Control Structures

## Array Creation and Initialization

- What is the output of the following code:

```
int[] odds = new int[8];
for (int i=0 ; i < odds.length ; i++) {
    System.out.print(odds[i] + " ");
    odds[i] = 2*i+1;
    System.out.print(odds[i] + " ");
}
```

- Output:

0 1 0 3 0 5 0 7 0 9 0 11 0 13 0 15

Array creation: all elements get the default value for their type (0 for int).

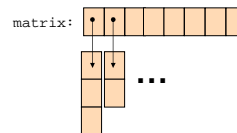
4

## Array Declaration

- An array is denoted by the [ ] notation

- Examples:

- int[] odds;
- int odds[]; // legal but discouraged
- String[] names;
- int[][] matrix; // an array of arrays



3

## Loop through Arrays

- By promoting the array's index:

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

- In Java 5.0:

```
for (String month: months) {
    System.out.println(month);
}
```

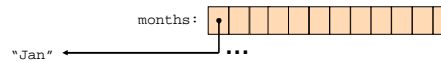
The variable month is assigned the next element in each iteration.

6

## Array Creation and Initialization

- Creating and initializing small arrays with *a-priori* known values:

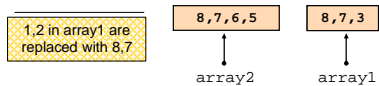
- int[] odds = {1,3,5,7,9,11,13,15};
- String[] months = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "July", "Aug", "Sep", "Oct", "Nov", "Dec"};



5

## Copying Arrays

- In the `java.lang.System` class:
  - public static void `arraycopy(Object src, int srcPos, Object dest, int destPos, int length)`
- `System.arraycopy(array2, 0, array1, 0, 2);`



- Details: <http://java.sun.com/j2se/1.5.0/docs/>

8

## Copying Arrays

- Assume:

```
int[] array1 = {1,2,3};
int[] array2 = {8,7,6,5};
```

- Naïve copy:

```
array1 = array2;
```

- How would we copy an array?

7

## Other Manipulations on Arrays

- The `java.util.Arrays` class has methods for sorting and searching, assigning arrays e.g.
  - public static void `sort(int[] a)`
  - public static int `binarySearch(int[] a, int key)`
  - public static void `fill(long[] a, long val)`
- More details in JDK 5.0 documentation <http://java.sun.com/j2se/1.5.0/docs/>

10

## Copying Arrays

- What is the output of the following code:

```
int[] odds = {1,3,5,7,9,11,13,15};
int[] newOdds = new int[8];
System.arraycopy(odds, 1, newOdds, 1, 7);
for (int odd: newOdds) {
    System.out.print(odd + " ");
}
```
- Output:  
0 3 5 7 9 11 13 15

9

## 2D Arrays

- `Table[i][j] = ij`

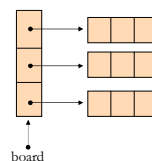
```
int[][] table = new int[10][10];
for (int i = 0 ; i < 10 ; i++) {
    table[i][0] = 1;
    for (int j = 1 ; j < 10 ; j++) {
        table[i][j] = table[i][j-1]*i;
    }
}
```
- Can we achieve that with the enhanced for loop? How? / Why?

12

## 2D Arrays

- There are no 2D arrays in Java but ...
- you can build array of arrays:

```
char[][] board = new char[3][];
for (int i = 0; i < 3 ; i++)
    board[i] = new char[3];
```



Or equivalently:  
`char[][] board = new char[3][3];`

11

## Switch Statement

```
public class Fibonacci {
    ...
    /** Returns the n-th Fibonacci element */
    public static int computeElement(int n) {
        switch(n) {
            case 0:
                return 1;
            case 1:
                return 1;
            default:
                return computeElement(n-1) + computeElement(n-2);
        }
    }
}
```

Assumption:  
 $n \geq 0$

can be placed  
outside the switch

14

## If-Else Statement

```
public class Fibonacci {
    ...
    /** Returns the n-th Fibonacci element */
    public static int computeElement(int n) {
        if (n==0) return 1;
        else if (n==1) return 1;
        else return computeElement(n-1) + computeElement(n-2);
    }
}
```

Assumption:  
 $n \geq 0$

Can be  
removed

13

## For Loop

- A loop instead of a recursion

```
static int computeElement(int n) {
    if (n == 0 || n == 1)
        return 1;

    int prev = 1;
    int prevPrev = 1;
    int curr;

    for (int i = 2; i < n; i++) {
        curr = prev + prevPrev;
        prevPrev = prev;
        prev = curr;
    }

    curr = prev + prevPrev;
    return curr;
}
```

Assumption:  
 $n \geq 0$

16

## Switch Statement

```
public class Fibonacci {
    ...
    /** Returns the n-th Fibonacci element */
    public static int computeElement(int n) {
        switch(n) {
            case 0:
                return 1;
            case 1:
                return 1;
            break;
            default:
                return computeElement(n-1) + computeElement(n-2);
        }
    }
}
```

Assumption:  
 $n \geq 0$

Compilation Error:  
Dead Code

15

## תשובה לתרגיל

```
static int computeElement(int n){
    if (n==0 || n==1)
        return 1;

    int prev = 1;
    int prevPrev = 1;

    for (int i = 2; i < n; i++) {
        prev = prev + prevPrev;
        prevPrev = prev - prevPrev;
    }

    return prev + prevPrev;
}
```

18

## נתונים במקום חישוב

- בתרגום רקורסיה ללולאה אנו משתמשים במשתני עזר לשמירת המצב  $prev$ ,  $prevPrev$  ו- $curr$
- הלולאה "זוכרת" את הנקודה שבה אנו נמצאים בתהליך החישוב
- **דין**: יעילות לעומת פשטות.
- עיקרון ה-KISS (keep it simple stupid)
- **תרגיל**: כתבו את השירות `computeElement` בעזרת  $prev$  ו- $prevPrev$  בלבד (ללא  $curr$ )

17

## מודולריות, שכפול קוד ועילות

- יש כאן חוסר יעילות מסוים:
  - לולאת ה-for חוזרת גם ב-main וגם ב-computeElement. לכאורה, במעבר אחד ניתן גם לחשב את האברים וגם להדפיס אותם
  - כמו כן כדי לחשב איבר בסדרה איננו משתמשים בתוצאות שכבר חישבנו (של אברים קודמים) ומתחילים כל חישוב מתחילתו

20

## For Loop

- Printing the first n elements:

```
public class Fibonacci {
    public static int computeElement(int n) {
        ...
    }

    public static void main(String[] args) {
        for(int i = 0; i < 10; i++)
            System.out.println(computeElement(i));
    }
}
```

It is better to use args[0]

19

## for vs. while

- The following two statements are almost equivalent:

```
for(int i = 0; i < n; i++)
    System.out.println(computeElement(i));
```

Variable i is not defined outside the for block

```
int i=0;
while (i < n) {
    System.out.println(computeElement(i));
    i++;
}
```

22

## מודולריות, שכפול קוד ועילות

- מתודה (פונקציה) צריכה לעשות דבר אחד בדיוק!
  - ערוב של חישוב והדפסה פוגע במודולריות (מדוע?)
  - היזהרו משכפול קוד!
    - קטע קוד דומה המופיע בשתי פונקציות שונות יגרום במוקדם או במאוחר לבאג בתוכנית (מדוע?)
  - את בעיית היעילות (הוספת מנגנון memoization) אפשר לפתור בעזרת מערכים (תרגיל)

21

## while vs. do while

- The following two statements are equivalent if and only if  $n > 0$  :

```
int i=0;
while (i < n) {
    System.out.println(computeElement(i));
    i++;
}
```

```
int i=0;
do {
    System.out.println(computeElement(i));
    i++;
} while (i < n);
```

works since  $n \geq 1$

23