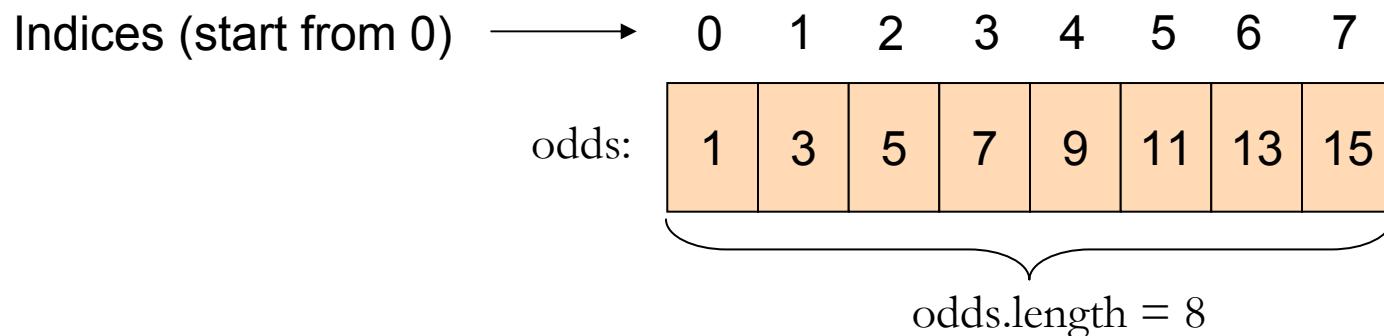


Software 1

Recitation No. 2
Arrays and Control Structures

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`

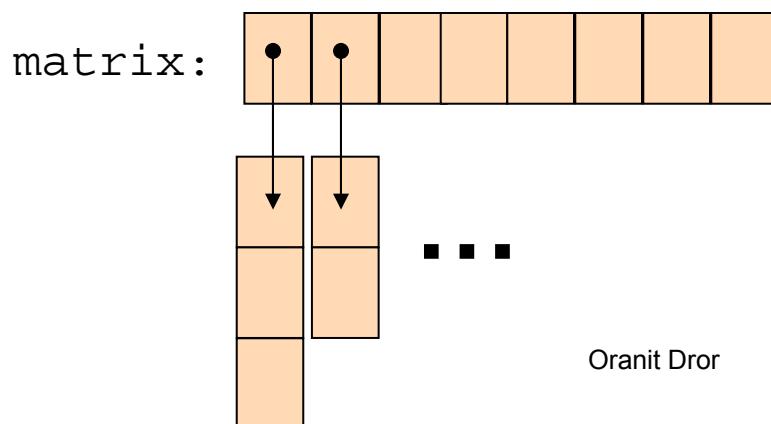
The value of the element at index 4 is 9: `odds[4] == 9`

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



Array Creation and Initialization

- What is the output of the following code:

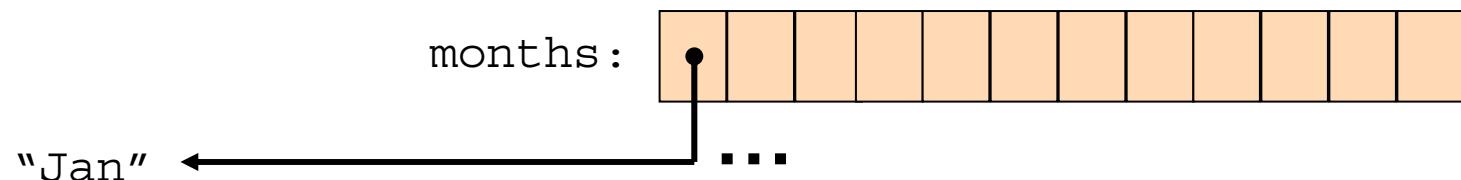
```
int [] odds = new int [8];  
for (int i=0 ; i < odds.length ; i++) {  
    System.out.print(odds[i] + " ");  
    odds[i] = 2*i+1;  
    System.out.print(odds[i] + " ");  
}
```

- Output: ?

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" };
```



Loop through Arrays

■ By promoting the array's index:

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

The variable month is assigned
the next element in each iteration

■ In Java 5.0:

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

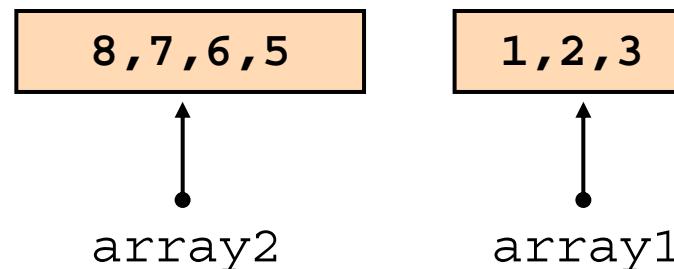
Copying Arrays

■ Assume:

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

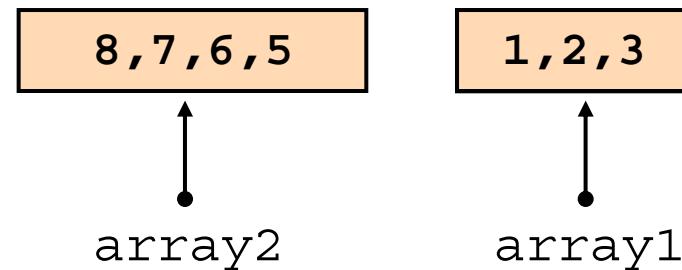
■ Naïve copy:

```
array1 = array2;
```



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/>

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:?

Other Manipulations on Arrays

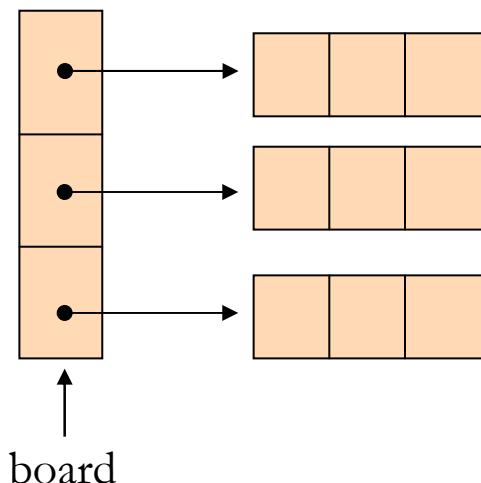
- The `java.util.Arrays` class has methods for sorting and searching, assigning arrays e.g.
 - `public static void sort(int[] a)`
 - `public static int binarySearch(int[] a, int key)`
 - `public static void fill(long[] a, long val)`
- More details in JDK 5.0 documentation

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

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] ;
```



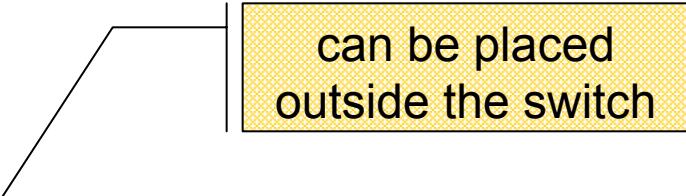
Or equivalently:
`char[][] board = new char[3][3];`

2D Arrays

```
int[][] table = new int[10][10];
for (int i = 0 ; i < 10 ; i++) {
    for (int j = 0 ; j < 10 j++) {
        table[i][j] = (i+1) * (j+1);
    }
}
```

Switch Statement

```
static int computeElement(int n) {  
    switch(n) {  
        case 0:  
            return 1;  
        case 1:  
            return 1;  
        default:  
            return computeElement(n-1) + computeElement(n-2);  
    }  
}
```



can be placed
outside the switch

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);  
        }  
    }  
}
```

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;  
}
```

נתונים במקום חישוב

- בתרגום רקורסיה לולאה אנו משתמשים במשתני עזר לשמרות המצביע prevPrev, curr, prev ו-
הlolאה "זכרת" את הנקודה שבה אנו נמצאים
בתחילת החישוב
- דיוון: יעילות לעומת פשוטות.
עיקנון ה-KISS (keep it short and simple)
- תרגיל: כתבו את השירות computeElement
בעזרת prevPrev-
curr בלבד (לא prev ו- curr)

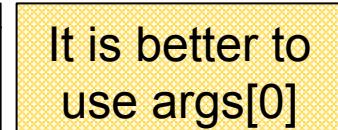
תשובה לתרגיל

```
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;  
}
```

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]

מודולריות, שכפול קוד ויעילות

- יש כאן חוסר יעילות מסוים:
- לולאת ה-for חוזרת גם ב- main וגם ב- computeElement. לכאורה, מעבר אחד ניתן גם לحساب את האברים וגם להדפס אותם
- כמו כן כדי לحساب איבר בסדרה איןנו משתמשים בהתוצאות שכבר חישבנו (של אברים קודמים) ומתחילה כל חישוב מתחילה

מודולריות, שכפול קוד ויעילות

- מתודה (פונקציה) צריכה לעשות דבר אחד בדיק!
- ערוב של חישוב והdfs פוגע במודולריות (מדוע?)
- היזרו משכפול קוד!
- קטע קוד דומה המופיע בשתי פונקציות שונות יגרום במקדם או מאוחר לבאג בתוכנית (מדוע?)
- את בעיית היעילות (הוסףת מנגן memoization) אפשר לפתור בעזרת מערכים

for vs. while

- The following two statements are equivalent:

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

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

while vs. do while

- The following two statements are equivalent:

```
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 $n \geq 1$