Сложни типове данни

1. **Класове**
2. **Интерфейси**
3. **Масиви**

Паметта за сложните типове данни се отделя в хийпа (динамична памет в RAM).
Паметта за примитивните типове данни се отделя в стека.

Идентификаторът за сложните типове данни представлява референция (reference) към обект.
Обектът трябва да се създаде чрез new и референцията да се свърже с обекта.

```
String s;
s = new String("Сума");
```

```
String s = new String("Сума");
```

или

```
String s;
s = "Сума";
```

```
String s = "Сума";
```

Масив

Масив – група от данни от един и същи тип с общо име (<идентификатор>).

1. Декларация на масив

<тип> [ ] <идентификатор>
<тип> <идентификатор> [ ]

<тип> – **тип на елементите на масива**

```
int [ ] a;
```

2. Отделяне на памет и определяне на размера

<идентификатор> = new <тип>[<израз>];
<израз> – **положителен от тип int (не може да е long).**

Ако не е отделена памет за елементите на масива, масивът има стойност null – масив с нулев брой елементи.

```
a = new int[10];
```

3. Многомерен масив – масив, чиито елементи са масиви;

– **има произволен брой размерности.**

public int length
Поле length – размер на масив.

```
float[][] matrix = new float[3][3];
```

или

```
float[][] matrix = new float[3][];
for (int j = 0; j < matrix.length; j++)
    matrix[j] = new float[3];
```

4. Изчисляване на <израз> за създаване на масив при изпълнение на програмата

– **<изразите> се изчисляват отляво надясно; ако някой завърши неуспешно, изразите надясно от него не се изчисляват;**
– **ако стойността на някой <израз> е отрицателна, се прехвърля изключението NegativeArrayException;**
– **отделя се памет за нов масив;**
  – **ако няма достатъчно памет, <израз> завършва неуспешно и прехвърля изключението AutoMemoryError;**
  – **ако има само един <израз>, създава се едномерен масив с определената дължина и всеки елемент се инициализира със стойност по подразбиране (за примитивните типове е 0);**
  – **ако има n израза, изпълнява се последователност от вградени цикли и се създават масиви от масиви.**

Пример: Създаване на триъгълна матрица

```
float[][] triang = new float[100][];
for(int j = 0; j < triang.length; j++)
    triang[j] = new float[j+1];
```

5. Достъп до елемент на масив

<идентификатор>[<израз>]

<израз> – **индекс на елемент от масив, тип** int**; индексирането започва от** 0**.**

a[8] = 5;

6. Изчисление на <израз> за обръщение към масив

- **ако завърши неуспешно, целият израз завършва неуспешно и индексният** <израз> **не се изчислява;**
- **ако стойността му е** null**, прехвърля се** NullPointerException**;**
- **в противен случай се изчислява индексният** <израз>**;**
  - **ако неговата стойност е отрицателна или по-голяма от дължината на масива, прехвърля се изключението** IndexOutOfBoundsException**;**
  - **в противен случай се осъществява обръщение към елемент от масива.**

```
a[2] = 5;
matrix[2][3] = 3.5;
```

Изразът от лявата страна на [] **се изчислява напълно преди индексния израз в** []**.**

7. Инициализация на масив

<тип> [] ... <идентификатор> = {<списък от стойности>}

```
int[] a = {10, 20, 30, 40, 50};

int[][] b = {{1, 2, 3}, {4, 5, 6}};
```

Пример: Достъп до елемент на масив

```
class Array {
  public static void main(String[] args) {
    int[] a = {11, 12, 13, 14};
    int[] b = {0, 1, 2, 3};
    System.out.println(a[(a=b)[3]]);
  }
}
```

Резултат:

14

a[(a=b)[3]]

Изразът отляво (a**) се изчислява напълно преди израза** (a=b)[3]**, т.е.** a **се запомня, докато се изчислява** (a=b)[3]**.**

a[(a=b)[3]] = a[b[3]] = a[3] = 14

Пример: Сума от елементите на едномерен масив и максимален елемент

```
import java.io.*;
class Arrays {
  public static void main (String[] args) throws IOException {
    BufferedReader stdin;
    stdin = new BufferedReader (new InputStreamReader (System.in));
    int[] a = new int [100];
    int n, sum, max;
    System.out.print ("Въведи брой елементи: ");
    n = Integer.parseInt(stdin.readLine());
    for (int i = 0; i < n; i++) {
      System.out.print ("a[" + (i+1) + "]: ");
      a[i] = Integer.parseInt(stdin.readLine());
    }
    System.out.println ("Елементи на масива");
    for (int i = 0; i < n; i++)
      System.out.print (a[i] + " ");
    System.out.println();
```

Въвеждане на елементите на масив

Извеждане на елементите на масив

```
    sum = 0;
    for (int i = 0; i < n; i++)
        sum += a[i];
    System.out.println("Сума = " + sum);
    max = a[0];
    for (int i = 1; i < n; i++)
      if (a[i] > max)
        max = a[i];
    System.out.println("Максимален елемент = " + max);
  }
}
```

Сума от елементите на масив

Максимален елемент

Пример: Максимален елемент и неговото място (индекс)

```
max = a[0];
int index = 0;
for (int i = 0; i < n; i++)
  if (a[i] > max) {
    max = a[i];
    index = i;
  }
System.out.println("Максимален елемент = " + max);
System.out.println("Индекс = " + index);
```

Пример: Двумерен масив

```
int[][] table = new int[5][5];
for (int row = 0; row < table.length; row++)
      for (int column = 0; column < table[row].length; column++)
            table[row][column] = Integer.parseInt(stdin.readLine());
System.out.println ("Елементи на масива");
for (int row = 0; row < table.length; row++) {
      for (int column = 0; column < table[row].length; column++)
            System.out.print(table[row][column] + " ");
   System.out.println();
}
```

Масивът table може да се представи и чрез масив от масиви:

```
int[] a0 = new int[5];
int[] a1 = new int[5];
int[] a2 = new int[5];
int[] a3 = new int[5];
int[] a4 = new int[5];
int[][] table = {a0, a1, a2, a3, a4};
```

Сортиране на елементите на едномерен масив

Сортирането е процес на пренареждане на дадено множество от обекти в определен ред.

$a_0, a_1, a_2, \ldots, a_{n-1}$

Сортиран масив във възходящ ред:

$a_{k0} \le a_{k1} \le a_{k2} \le \ldots \le a_{k,n-1}$

Сортиране чрез пряка размяна (метод на мехурчето)

Ако $a_{j-1} > a_j$, a_j и a_{j-1} разменят местата си.

	i=1	i=2	i=3	i=4
10	3	3	3	3
4	10	4	4	4
5	4	10	5	5
3	5	5	10	8
8	8	8	8	10

Пример: Сортиране по метода на мехурчето

```
for (int i = 1; i < n; i++)
  for (int j = n - 1; j >= i; j--)
    if (a[j - 1] > a[j]) {                  // Елементите разменят местата си
      int temp = a[j - 1];
      a[j-1] = a[j];
      a[j] = temp;
    }
```