Boaz Kantor
Introduction to Computer Science,
Fall semester 2010-2011
IDC Herzliya

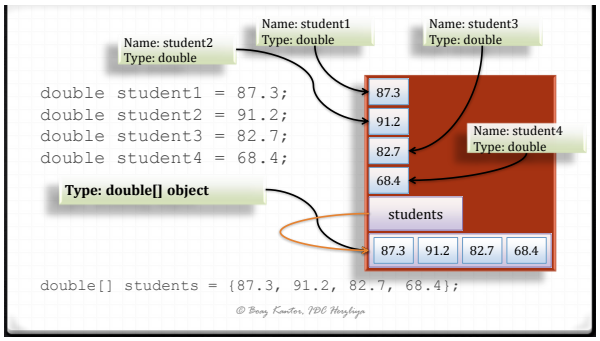
Arrays

"Whoa!" - ~ Neo, *The Matrix*


```
class Average {
  public static void main(String[] args) {
    double x = 4.2;
    double y = 5.9;
    System.out.println("The average of " + x + " and " + y + " is: " +
        average(x, y));
    double z = 2.1;
    System.out.println("The average of " + x + ", " + y + " and " + z +
        " is: " + average(x, y, z));
  }

  public static double average(double num1, double num2) {
    return (num1 + num2) / 2;
  }

  public static double average (double num1, double num2, double num3) {
    return (num1 + num2 + num3) / 3;
  }
}
```



```
public static double classAverage(double student1, double student2, double student3, double student4,
    double student5, double student6, double student7, double student8, double student9, double student10,
    double student11, double student12, double student13, double student14, double student15, double
    student16, double student17, double student18, double student19, double student20, double student21,
    double student22, double student23, double student24, double student25, double student26, double
    student27, double student28, double student29, double student30, double student31, double student32,
    double student33, double student34, double student35, double student36, double student37, double
    student38, double student39, double student40, double student41, double student42, double student43,
    double student44, double student45, double student46, double student47, double student48, double
    student49, double student50, double student51, double student52, double student53, double student54,
    double student55, double student56, double student57, double student58, double student59, double
    student60, double student61, double student62, double student63, double student64, double student65,
    double student66, double student67, double student68, double student69, double student70, double
    student71, double student72, double student73, double student74, double student75, double student76,
    double student77, double student78, double student79, double student80, double student81, double
    student82, double student83, double student84, double student85, double student86, double student87,
    double student88, double student89, double student90, double student91, double student92, double
    student93, double student94, double student95, double student96, double student97, double student98,
    double student99, double student100) {

    return (student1 + student2 + student3 + student4 + student5 + student6 + student7 + student8 +
    student9 + student10 + student11 + student12 + student13 + student14 + student15 + student16 +
    student17 + student18 + student19 + student20 + student21 + student22 + student23 + student24 +
    student25 + student26 + student27 + student28 + student29 + student30 + student31 + student32 +
    student33 + student34 + student35 + student36 + student37 + student38 + student39 + student40 +
    student41 + student42 + student43 + student44 + student45 + student46 + student47 + student48 +
    student49 + student50 + student51 + student52 + student53 + student54 + student55 + student56 +
    student57 + student58 + student59 + student60 + student61 + student62 + student63 + student64 +
    student65 + student66 + student67 + student68 + student69 + student70 + student71 + student72 +
    student73 + student74 + student75 + student76 + student77 + student78 + student79 + student80 +
    student81 + student82 + student83 + student84 + student85 + student86 + student87 + student88 +
    student89 + student90 + student91 + student92 + student93 + student94 + student95 + student96 +
    student97 + student98 + student99 + student100) / 100;
}
```

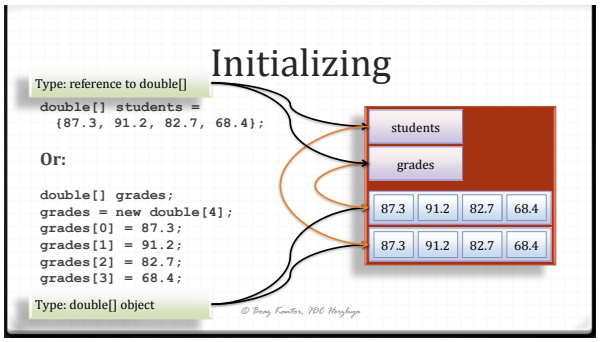


Syntax

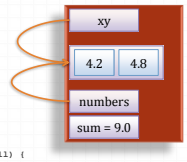
- Declaring:
  - `<Type>[] <name>;`
  - `double[] vector;`
- Initializing:
  - `<name> = new <Type>[<number_of_cells>];`
  - `vector = new double[63];`
  - Constant initialization:
    - `<Type>[] <name> = {<cell_value_1>, <cell_value_2>, …};`
    - `double[] vector = {12.0, 13.52, -1.444};`
- Accessing:
  - Array object:
    - `<name>.<member>;`
    - `vector.length;`
  - Array cell:
    - `<name1>[<cell_number>]`
    - `vector[2] = 14.8;`
    - `double element = vector[1];`

© Barg Kantor, PDC Herzliya

Initializing



```
class Average {
   public static void main(String[] args) {
      double[] xy = new double[2];
      xy[0] = 4.2;
      xy[1] = 4.8;
      System.out.println("The average of " + xy[0] + " and " + xy[1] + " is: " + average(xy));
      double[] xyz = {xy[0], xy[1], 2.1};
      System.out.println("The average of " + xyz[0] + ", "
            + xyz[1] + " and " + xyz[2] + " is: "
            + average(xyz));
   }

   public static double average(double[] numbers) {
      if (numbers == null) {
            throw new IllegalArgumentException();
      }
      if (numbers.length == 0) {
            return 0;
      }
      double sum = 0;
      for (int currCell = 0; currCell < numbers.length; ++currCell) {
            sum += numbers[currCell];
      }
      return sum / numbers.length;
   }
}
```



Foreach

- A nicer way of iteration. Instead of:
  double[] vec = new double[50];

```
for (int i = 0; i < vec.length; ++i) {
    System.out.println("i=" + i);
}
```

- Use this, it's the same, but looks better and less error-prone:

```
for (int i : vec) {
    System.out.println("i=" + i);
}
```


Class Arrays

- Arrays of classes
- Initializing the array does not initialize its cells!

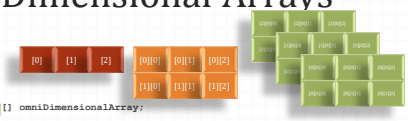
```
Turtle[] bale = new Turtle[3];
bale[0] = new Turtle();
bale[0].tailDown();
```

NullPointerException !!

Multi Dimensional Arrays

- Declaring:
  `<Type>[] <name>;`

```
double[] vector;
double[][] matrix;
double[][][] cube;
double[][][][][][][][][..][] omniDimensionalArray;
```




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

Where is the bug?
What 2 approaches can fix it?

Multi Dimensional Arrays – Exercise

- Define a class **`TravelLengths`** which provides information services about distances between cities.
- API:

```
public TravelLengths(String[] cityNames, int[][]distances);
public int getDistance(String from, String to);
```

- Sample use:

```
String[] cities = {"Tel Aviv", "Netanya", "Haifa"};
int[][] distances = {{0, 30, 85}, {30, 0, 55}, {85, 55, 0}};
TravelLengths t = new TravelLengths(cities, distances);
System.out.println("The distance from Haifa to Tel Aviv is " +
   t.getDistance("Haifa", "Tel Aviv") + " km");
```

	Tel Aviv	Netanya	Haifa
Tel Aviv	0	30	85
Netanya	30	0	55
Haifa	85	55	0

Multi Dimensional Arrays – Solution

```
public class TravelLengths {
   private String[] cityNames;
   private int[][] distances;
   public TravelLengths(String[] cityNames, int[][]distances) {
      this.cityNames = cityNames; // should we copy the array?
      this.distances = distances; // should we copy the array?
      // array copying is described next
   }
   public int getDistance(String from, String to) {
      // search for the desired cities
      int fromIndex = 0;
      int toIndex = 0;
      while (!cityNames[fromIndex++].equals(from));
      while (!cityNames[toIndex++].equals(to));
      return distances[fromIndex - 1][toIndex - 1];
   }
}
```

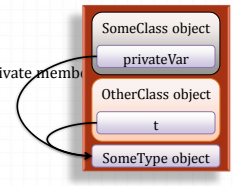
Copying Arrays – Encapsulation

- When we have a reference as a private member:

```
class SomeClass {
   private SomeType privateVar;
   public SomeClass(SomeType other) {
      privateVar = other;
   }
}
```

- The caller keeps a reference to our private memb

```
class OtherClass {
   SomeType t;
   private static void foo() {
      t = new SomeType("I have $10 in the bank");
      SomeClass c = new SomeClass(t);
      t.setBalance(1000000);
   }
}
```

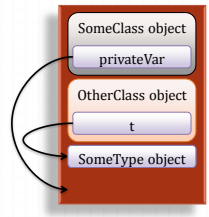


Copying Arrays – Encapsulation

- To solve this, we create our own copy:

```
class SomeClass {
   private SomeType privateVar;
   public SomeClass(SomeType other) {
      // using copy-constructor:
      privateVar = new SomeType(other);
      // using clone() method:
      privateVar = other.clone();
      // using assignments:
      privateVar = new SomeType();
      privateVar.setName(other.getName());
      privateVar.setBalance(other.getBalance());
   }
}
```

- How do we do that with arrays?



Copying Arrays

- Either:

```
class SomeClass {
   private int[] privateVar;
   public SomeClass(int[] other) {

      // using System.arrayCopy:
      privateVar = new int[other.length];
      System.arrayCopy(other, 0, this.privateVar, 0, other.length);

      // using clone() method:
      privateVar = other.clone();

      // using assignments:
      privateVar = new int[other.length];
      for (int currIndex = 0; currIndex < privateVar.length; ++currIndex) {
         privateVar[currIndex] = other[currIndex];
      }
   }
}
```

Every array has a clone() method!

- How do we do that with multi-dimensional arrays?

Copying Multi-Dimensional Arrays

```
public class TravelLengths {
    private String[] cityNames;
    private int[][] distances;

    // either of the following:
    public TravelLengths(String[] cityNames, int[][]distances) {
        this.cityNames = new String[cityNames.length];
        System.arraycopy(cityNames, 0, this.cityNames, 0, cityNames.length);
        this.distances = new int[distances.length][distances[0].length];
        for (int currDistance = 0; currDistance < distances.length; ++currDistance) {
            System.arraycopy(distances[currDistance], 0,
                this.distances[currDistance], 0, distances.length);
        }
    }

    Or:
    public TravelLengths(String[] cityNames, int[][]distances) {
        this.cityNames = cityNames.clone();
        this.distances = distances.clone();
    }
}
```


Multi Dimensional Arrays – Exercising

- How to solve:

1. Write the skeleton (called "a stub"):
   1. All the needed method definitions
   2. Documentation
   3. Trivial implementation:
      - `return 0;`
      - `return false;`
      - `throw new RuntimeException("Unimplemented!");  //< Recommended!`
2. Write a test Main method (called "a driver")
3. Decide on data structures:
   1. Add as instance variables
   2. Add getters/setters where needed
4. Write the solution as pseudo-code comments
5. Divide to methods where needed
6. Implement, test and fix. And test and fix and test (and fix, and test). Then fix.
7. Test.

Boaz Kantor
Introduction to Computer Science,
Fall semester 2010-2011
IDC Herzliya

Arrays

Next: styling and best practices

Styling rules

- General form:

```
Type[] name = new Type[number];
Type[] name = {initval1, initval2, .., initvaln};
```

- Arrays should be declared with the brackets next to the type:

```
int[] vec = new int[8];
```

(and not `int vec[] = new int[8];`)

Styling rules, foreach

- General form:

```
for (type var : collection) {
    // code
}
```

- Note the blank spaces.

Best practices

- Until the size is known, initialize arrays with null.

```
int[] vec = new int[5];
Turtle turtleMatrix = null;
// some code
turtleMatrix = new Turtle[numOfTurtles];
```

Best practices

- Prefer `foreach` over `for` whenever possible.
- Prefer generic collections over non-generics.