Initialization

Adam Krechowicz

Initialization

Initialization

In java every element must be properly initialized before use

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public static void main(String[] args) {
6          int i;
7          System.out.println(i);
8      }
9  }
```

Initialization

Initialization

In java every element must be properly initialized before use

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public static void main(String[] args) {
6          int i;
7          System.out.println(i);
8      }
9  }
```

variable i might not have been initialized!

Proper way to initialize primitive types

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public static void main(String[] args) {
6          int i = 1;
7          System.out.println(i);
8      }
9  }
```

Object initialization

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      int i;
6
7      public void test(){
8
9      }
10
11     public static void main(String[] args) {
12         test();
13         System.out.println(i);
14     }
15 }
```

Object initialization

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      int i;
6
7      public void test(){
8
9      }
10
11     public static void main(String[] args) {
12         test();
13         System.out.println(i);
14     }
15 }
```

non-static variable i cannot be referenced from a static context cannot find method test()

Object initialization

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public void test(){
6
7      }
8
9      public static void main(String[] args) {
10         InitializationNeeds initializationNeeds;
11         initializationNeeds.test();
12     }
13 }
```

Object initialization

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public void test(){
6
7      }
8
9      public static void main(String[] args) {
10         InitializationNeeds initializationNeeds;
11         initializationNeeds.test();
12     }
13 }
```

variable initializationNeeds might not have been initialized

Object initialization

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public void test(){
6
7      }
8
9      public static void main(String[] args) {
10         InitializationNeeds initializationNeeds;
11         initializationNeeds = null;
12         initializationNeeds.test();
13     }
14 }
```

Object initialization

```
1  package com.wyklad.init;
2
3  public class InitializationNeeds {
4
5      public void test(){
6
7      }
8
9      public static void main(String[] args) {
10         InitializationNeeds initializationNeeds;
11         initializationNeeds = null;
12         initializationNeeds.test();
13     }
14 }
```

Exception in thread "main"java.lang.NullPointerException

Constructors

```
1  package com.wyklad.init;
2
3  public class Constructor {
4
5      int pole;
6
7      void metoda(){}
8
9      public Constructor() {
10         super();
11     }
12
13     public static void main(String[] args){
14         Constructor c = new Constructor();
15         c.pole = 3;
16         c.metoda();
17     }
18
19 }
```

Constructors with parameters

```
1  package com.wyklad.init;
2
3  public class ConstructorParameters {
4
5      private int i;
6
7      public ConstructorParameters() {
8          super();
9      }
10
11     public ConstructorParameters(int i){
12         this.i = i;
13     }
14
15     public static void main(String[] args) {
16         ConstructorParameters cp = new ConstructorParameters();
17         ConstructorParameters cp1 = new ConstructorParameters(1);
18     }
19 }
```

this

```
1  package com.wyklad.init;
2
3  public class This {
4
5      int i = 5;
6
7      public This(int i) {
8          System.out.println(i);
9      }
10
11     public static void main(String[] args) {
12         This t = new This(6);
13     }
14 }
```

Default constructor

```
1  package com.wyklad.init;
2
3  public class DefaultConstructor {
4
5      public static void main(String[] args) {
6          DefaultConstructor defaultConstructor = new DefaultConstructor();
7      }
8  }
```

Default constructor

```
1  package com.wyklad.init;
2
3  public  class  DefaultConstructor {
4
5      public DefaultConstructor(int i){
6
7      }
8
9      public  static  void main(String[] args) {
10         DefaultConstructor defaultConstructor = new DefaultConstructor();
11         DefaultConstructor defaultConstructor1 = new DefaultConstructor(1);
12     }
13 }
```

Default constructor

```
1  package com.wyklad.init;
2
3  public  class  DefaultConstructor {
4
5      public DefaultConstructor(int i){
6
7      }
8
9      public  static  void main(String[] args) {
10         DefaultConstructor defaultConstructor = new DefaultConstructor();
11         DefaultConstructor defaultConstructor1 = new DefaultConstructor(1);
12     }
13 }
```

cannot find constructor DefaultConstructor()

Constructors and inheritance

```
1  package com.wyklad.init;
2
3  class Bazowa{
4      public Bazowa(){
5          System.out.println("Konstruktor bazowy");
6      }
7  }
8
9  public class ExtendsInit extends Bazowa {
10     public ExtendsInit() {
11         super();
12         System.out.println("Konstruktor");
13     }
14
15     public static void main(String[] args){
16         ExtendsInit e = new ExtendsInit();
17     }
18 }
```

super

```
1  package com.wyklad.init;
2
3  public  class  ExtendsInit extends Bazowa {
4
5      public ExtendsInit() {
6          System.out.println("Konstruktor");
7          super();
8      }
9
10     public  static  void main(String[] args){
11         ExtendsInit e = new ExtendsInit();
12     }
13 }
```

super

```
1  package com.wyklad.init;
2
3  public  class  ExtendsInit extends Bazowa {
4
5      public ExtendsInit() {
6          System.out.println("Konstruktor");
7          super();
8      }
9
10     public  static  void main(String[] args){
11         ExtendsInit e = new ExtendsInit();
12     }
13 }
```

call to super must be first statement in constructor

Elementy statyczne

```
1  package com.wyklad.init;
2
3  public class Static {
4
5      static int pole;
6
7      static void metoda(){}
8
9      public Static() {
10         super();
11     }
12
13     public static void main(String[] args) {
14         pole = 3;
15         metoda();
16     }
17 }
```

Static elements

```
1  package com.wyklad.init;
2
3  public class Static {
4
5      static int pole;
6
7      static void metoda(){}
8
9      public static void main(String[] args) {
10         Static s = new Static();
11         s.pole = 3;
12         s.metoda();
13
14         Static s1 = new Static();
15         s.pole = 5;
16         s1.pole = 6;
17         System.out.println(s.pole);
18         System.out.println(s1.pole);
19     }
20 }
```

Static elements and this

```
1  package com.wyklad.init;
2
3  public class StaticThis {
4
5      int i;
6
7      void metoda(){
8          this.i = 5;
9      }
10
11     static void metoda1(){
12         this.i = 6;
13     }
14
15 }
```

Static elements and this

```
1  package com.wyklad.init;
2
3  public class StaticThis {
4
5      int i;
6
7      void metoda(){
8          this.i = 5;
9      }
10
11     static void metoda1(){
12         this.i = 6;
13     }
14
15 }
```

non-static variable this cannot be referenced from a static context

Objects counting

```
1  package com.wyklad.init;
2
3  public class ObjectCount {
4
5      public static int i = 0;
6
7      public ObjectCount() {
8          super();
9          System.out.println(++i);
10     }
11
12     public static void main(String[] args) {
13         ObjectCount objectCount = new ObjectCount();
14         ObjectCount a,b,c;
15         a = new ObjectCount();
16         b = new ObjectCount();
17         c = new ObjectCount();
18     }
19 }
```

Initialization block

```
1  package com.wyklad.init;
2
3  public class Initialization {
4
5      {
6          System.out.println(" Initialization ");
7          i = 2;
8      }
9
10     public static void main(String[] args) {
11         Initialization initialization = new Initialization();
12         System.out.println(initialization.i);
13     }
14 }
```

Initialization block

```
1  package com.wyklad.init;
2
3  public class Initialization {
4
5      {
6          System.out.println(" Initialization ");
7          i = 2;
8      }
9
10     int i = 1;
11
12     {
13         i = 5;
14     }
15
16     public static void main(String[] args) {
17         Initialization initialization = new Initialization();
18         System.out.println(initialization.i);
19     }
20 }
```

Initialization block

```
1  package com.wyklad.init;
2
3  public class Initialization {
4
5      {
6          System.out.println(" Initialization ");
7          i = 2;
8      }
9
10     public Initialization() {
11         super();
12         System.out.println("Constructor");
13     }
14
15     public static void main(String[] args) {
16         Initialization initialization = new Initialization();
17     }
18 }
```

Static initialization block

```
1  package com.wyklad.init;
2
3  public class StaticInitialization {
4
5    static int i;
6
7      static {
8          System.out.println("Static initialization ");
9          i = 1;
10     }
11
12
13     public static void main(String[] args) {
14         StaticInitialization staticInitialization = new StaticInitialization↩
       ();
15     }
16 }
```

Static initialization block

```
1  package com.wyklad.init;
2
3  public class StaticInitialization {
4
5      static {
6          i = 2;
7      }
8
9      static int i = 4;
10
11     static {
12         System.out.println("Static initialization ");
13         i = 1;
14     }
15
16     public static void main(String[] args) {
17         StaticInitialization staticInitialization = new StaticInitialization↩
       ();
18     }
19 }
```

Static initialization block

```
1  package com.wyklad.init;
2
3  public class StaticInitialization {
4
5      static {
6          System.out.println("Static  initialization ");
7      }
8
9      {
10         System.out.println(" Initialization ");
11     }
12
13     public StaticInitialization() {
14         System.out.println("Constructor");
15     }
16
17     public static void main(String[] args) {
18         StaticInitialization si = new StaticInitialization();
19     }
20 }
```

Static block initialization

```
1  package com.wyklad.init;
2
3  public class StaticInitialization {
4
5      static int i;
6
7      static {
8          System.out.println("Static initialization ");
9          i = 1;
10     }
11
12     public static void main(String[] args) {
13         System.out.println("Program start");
14         StaticInitialization staticInitialization = new StaticInitialization↩
       ();
15         StaticInitialization staticInitialization1 = new StaticInitialization↩
       ();
16     }
17 }
```

Initialization and inheritance

```
1  package com.wyklad.init;
2
3  class Bazowa{
4
5      static {
6          System.out.println("Base static   initialization  ");
7      }
8
9      {
10         System.out.println("Base  initialization ");
11     }
12
13     public Bazowa(){
14         System.out.println("Base constructor");
15     }
16
17 }
```

Initialization and inheritance

```
1  package com.wyklad.init;
2
3  public class ExtendsInit extends Bazowa {
4
5      static {
6          System.out.println("Static initialization ");
7      }
8
9      {
10         System.out.println(" Initialization ");
11     }
12
13     public ExtendsInit() {
14         System.out.println("Constructor");
15     }
16
17     public static void main(String[] args){
18         ExtendsInit e = new ExtendsInit();
19     }
20 }
```

Cloning

```
1  public class Cloning implements Cloneable {
2      int wartosc;
3      String nazwa;
4      public Cloning(int wartosc, String nazwa){
5          this.wartosc = wartosc;
6          this.nazwa = nazwa;
7      }
8      public String toString(){
9          return nazwa + " " + wartosc;
10     }
11     public static void main(String[] args){
12         try {
13             Cloning c1 = new Cloning(123, "hello");
14             Cloning c2 = (Cloning)c1.clone();
15             System.out.println(c1);
16             System.out.println(c2);
17         } catch (CloneNotSupportedException ex) {
18             Logger.getLogger(Cloning.class.getName()).log(Level.SEVERE, null, ex);
19         }
20     }
21 }
```

Factory

```
1  class Element{}
2
3  public class Factory {
4
5      public Element createElement(){
6          return new Element();
7      }
8
9      public static void main(String[] args){
10         Factory f = new Factory();
11         Element e = f.createElement();
12         Element e1 = f.createElement();
13     }
14 }
```

Finalization

Finalization

What is happening with unused objects?

Finalization

Garbage Collector

- GC is an element that is used to delete unused objects
- It works only when there is lack of memory
- It may not work during the whole program execution
- Removes unused objects – there no reference to object
- It may interfere with normal program execution

finalize method

```
1  package com.wyklad.init;
2
3  public class Finalization {
4
5      public Finalization() {
6          super();
7      }
8
9      protected void finalize(){
10         System.out.println("Object is deleted");
11     }
12
13     public static void main(String[] args){
14         Finalization f = new Finalization();
15     }
16 }
```

Metoda finalize

```
1  package com.wyklad.init;
2
3  public class Finalization {
4
5      public Finalization() {
6          super();
7      }
8
9      protected void finalize(){
10         System.out.println("Object is deleted");
11     }
12
13     public static void main(String[] args){
14         Finalization f;
15         for (int i = 0; i < 1000000; i++)
16             f = new Finalization();
17     }
18 }
```

Manual GC

```
1  package com.wyklad.init;
2
3  public class GarbageCollector {
4
5      public void finalize(){
6          System.out.println("Object is deleted");
7      }
8
9      public static void main(String[] args) {
10         GarbageCollector garbageCollector = new GarbageCollector();
11         System.gc();
12         garbageCollector = null;
13         System.gc();
14         garbageCollector = new GarbageCollector();
15         System.gc();
16         garbageCollector = new GarbageCollector();
17         System.gc();
18     }
19 }
```

Enumerations

```
1  enum Wyliczenie {
2      FISRT, SECOND, THIRD
3  }
4
5  public class Enumerations{
6
7      public static void main(String[] args){
8          Wyliczenie w = Wyliczenie.PIERWSZY;
9          switch (w){
10             case FIRST:
11                 System.out.println("It is  first ");
12                 break;
13             case SECOND:
14                 System.out.println("It is second");
15                 break;
16             case THIRD:
17                 System.out.println("It is  third ");
18                 break;
19         }
20     }
21
```

Arrays

```
1  package com.wyklad.init;
2
3  public class Array {
4
5      public static void main(String[] args) {
6          int [] tab;
7
8          tab[0] = 2;
9          tab[1] = 5;
10         tab[2] = 3;
11         tab[3] = 8;
12         tab[4] = 6;
13         System.out.println(tab1.length);
14         System.out.println(tab);
15         for (int i = 0; i < 5; i++)
16             System.out.println(tab[i]);
17     }
18 }
```

Arrays

```
1  package com.wyklad.init;
2
3  public class Array {
4
5      public static void main(String[] args) {
6          int [] tab;
7
8          tab[0] = 2;
9          tab[1] = 5;
10         tab[2] = 3;
11         tab[3] = 8;
12         tab[4] = 6;
13         System.out.println(tab1.length);
14         System.out.println(tab);
15         for (int i = 0; i < 5; i++)
16             System.out.println(tab[i]);
17     }
18 }
```

variable tab might not have been initialized

Array initialization

```
1  package com.wyklad.init;
2
3  public class Array {
4
5      public static void main(String[] args) {
6          int [] tab1 = new int[5];
7          int [] tab2 = {1,2,3,4,5};
8
9          tab1[0] = 2;
10         tab1[1] = 5;
11         tab1[2] = 3;
12         tab1[3] = 8;
13         tab1[4] = 6;
14         System.out.println(tab1.length);
15         for (int i = 0; i < 5; i++)
16             System.out.println(tab2[i]);
17     }
18 }
```

Range of array

```
1  package com.wyklad.init;
2
3  public class Array {
4
5      public static void main(String[] args) {
6          int [] tab1 = new int[5];
7
8          tab1[5] = 9;
9      }
10 }
```

Range of array

```
1  package com.wyklad.init;
2
3  public class Array {
4
5      public static void main(String[] args) {
6          int [] tab1 = new int[5];
7
8          tab1[5] = 9;
9      }
10 }
```

Exception in thread "main"java.lang.ArrayIndexOutOfBoundsException: 5

Arrays of objects

```
1  package com.wyklad.init;
2
3  public class ObjectArray {
4
5      public void metoda(){
6
7      }
8
9      public static void main(String[] args) {
10         ObjectArray[] objectArray = new ObjectArray[5];
11         for (int i = 0; i < 5; i++){
12             objectArray[i].metoda();
13         }
14     }
15 }
```

Arrays of objects

```
1  package com.wyklad.init;
2
3  public class ObjectArray {
4
5      public void metoda(){
6
7      }
8
9      public static void main(String[] args) {
10         ObjectArray[] objectArray = new ObjectArray[5];
11         for (int i = 0; i < 5; i++){
12             objectArray[i].metoda();
13         }
14     }
15 }
```

Exception in thread "main"java.lang.NullPointerException

Arrays of objects

```
1  package com.wyklad.init;
2
3  public class ObjectArray {
4
5      public void metoda(){
6
7      }
8
9      public static void main(String[] args) {
10         ObjectArray[] objectArray = new ObjectArray[5];
11         for (int i = 0; i < 5; i++){
12             objectArray[i] = new ObjectArray();
13             objectArray[i].metoda();
14         }
15     }
16 }
```

Arrays of objects

```
1  package com.wyklad.init;
2
3  public class ObjectArray {
4
5      public void metoda(){
6
7      }
8
9      public static void main(String[] args) {
10         ObjectArray[] objectArray = {new ObjectArray(), new ObjectArray(),
11                                      new ObjectArray(), new ObjectArray(),
12                                       new ObjectArray()};
13         for (int i = 0; i < 5; i++){
14             objectArray[i].metoda();
15         }
16     }
17 }
```

Array copying

```
1  package com.wyklad.init;
2
3  public class ArrayCopy {
4
5      public static void main(String[] args) {
6
7          char[] tablica1 = {'a', 'b', 'c', 'd'};
8          char[] tablica2 = new char[4];
9
10         System.arraycopy(tablica1, 0, tablica2, 0, 4);
11         System.out.println(tablica2);
12     }
13 }
```

Array copy

```
1  package com.wyklad.init;
2
3  public class ArrayCopy {
4      public ArrayCopy() {
5          super();
6      }
7
8      public static void main(String[] args) {
9
10         char [] tablica1 = {'a', 'b', 'c', 'd'};
11         char [] tablica2 = new char[4];
12
13         System.arraycopy(tablica1, 0, tablica2, 0, 15);
14         System.arraycopy(tablica1, 0, tablica2, 4, 4);
15     }
16 }
```

Exception in thread "main"java.lang.ArrayIndexOutOfBoundsException

Arrays class

java.util.Arrays

Supports the operations on arrays

- copying
- searching of elements
- filling with some values
- sorting
- array comparing

Searching

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class ArraysClass {
6
7      public static void main(String[] args) {
8          ArraysClass arraysClass = new ArraysClass();
9          int [] tab = {1,2,3,4,5,6,7};
10         System.out.println(Arrays.binarySearch(tab, 1));
11         System.out.println(Arrays.binarySearch(tab,2,4,1));
12         System.out.println(Arrays.toString(tab));
13     }
14 }
```

Searching

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class ArraysClass {
6
7      public static void main(String[] args) {
8          ArraysClass arraysClass = new ArraysClass();
9          int [] tab = {1,2,3,4,5,6,7};
10         System.out.println(Arrays.binarySearch(tab, 1));
11         System.out.println(Arrays.binarySearch(tab,2,4,1));
12         System.out.println(Arrays.toString(tab));
13     }
14 }
```

```
0
-3
[1, 2, 3, 4, 5, 6, 7]
```

Arrays copy

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class ArraysClass {
6
7      public static void main(String[] args) {
8          char[] tab1 = {'a', 'b', 'c', 'd', 'e'};
9          char[] tab2 = Arrays.copyOf(tab1, tab1.length);
10         System.out.println(new String(tab2));
11         char[] tab3 = Arrays.copyOf(tab1, tab1.length-2);
12         System.out.println(new String(tab3));
13         char[] tab4 = Arrays.copyOf(tab1, tab1.length+2);
14         System.out.println(new String(tab4));
15
16         char[] tab5 = Arrays.copyOfRange(tab1, 1, 3);
17         System.out.println(new String(tab5));
18     }
19 }
```

Array filling

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class ArraysClass {
6
7      public static void main(String[] args) {
8          char [] tab6;
9          Arrays.fill(tab6, 'a');
10     }
11 }
```

variable tab6 might not have been initialized

Arrays filling

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class ArraysClass {
6
7      public static void main(String[] args) {
8          char [] tab6;
9          tab6 = new char[6];
10         Arrays.fill(tab6, 'a');
11         System.out.println(new String(tab6));
12         Arrays.fill(tab6, 1, 3, 'b');
13         System.out.println(new String(tab6));
14     }
15 }
```

Array sorting

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class Sort{
6
7      public static void main(String[] args) {
8          int [] tablica = {4,6,3,8,2,7,4};
9          Arrays.sort(tablica);
10         for (int i = 0; i < tablica.length; i++){
11             System.out.print(tablica[i]);
12         }
13         System.out.println();
14 }
```

Array sorting

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class Sort{
6
7      public static void main(String[] args) {
8          int [] tablica3 = {4,6,3,8,2,7,4};
9          Arrays.sort(tablica3, 3, 5);
10         for (int i = 0; i < tablica3.length; i++){
11             System.out.print(tablica3[i]);
12         }
13         System.out.println();
14 }
```

Sorting of array of objects

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class Sort{
6      int i;
7      public Sort(int i) {
8          this.i = i;
9      }
10     public static void main(String[] args) {
11         Sort[] tablica2 = {new Sort(4), new Sort(1)};
12         Arrays.sort(tablica2);
13     }
14 }
```

Exception in thread "main"java.lang.ClassCastException:
com.wyklad.init.Sort cannot be cast to java.lang.Comparable

Sorting of arrays of objects

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class Sort implements Comparable {
6      int i;
7      public Sort(int i) {
8          this.i = i;
9      }
10     public static void main(String[] args) {
11         Sort[] tablica2 = {new Sort(4), new Sort(1)};
12         Arrays.sort(tablica2);
13     }
14     public int compareTo(Object o) {
15         return this.i-((Sort)o).i;
16     }
17 }
```

Searching in arrays of objects

```
1  public class SearchObject implements Comparable {
2      int i;
3      public SearchObject(int i) {
4          this.i = i;
5      }
6      public static void main(String[] args) {
7          SearchObject a = new SearchObject(5);
8          SearchObject b = new SearchObject(2);
9          SearchObject c = new SearchObject(7);
10         SearchObject d = new SearchObject(1);
11         SearchObject[] tablica = {a,b,c,d};
12         System.out.println(Arrays.binarySearch(tablica, c));
13     }
14     public int compareTo(Object o) {
15         return this.i-((SearchObject)o).i;
16     }
17 }
```

Arrays comparing

```
1  package com.wyklad.init;
2
3  import java.util.Arrays;
4
5  public class Deep {
6
7      public static void main(String[] args) {
8
9          int [] tab1 = {1,2,3,4};
10         int [] tab2 = {1,2,3,4};
11         int [] tab3 = {4,3,2,1};
12         System.out.println(Arrays.equals(tab1, tab2));
13         System.out.println(Arrays.equals(tab2, tab3));
14 }
```

Comparing of arrays of objects

```
1  public class Deep {
2      @Override
3      public boolean equals(Object o){
4          System.out.println("checking");
5          return true;
6      }
7      public static void main(String[] args) {
8          Deep[] deep = new Deep[3];
9          Deep[] deep1 = new Deep[3];
10         for (int i = 0; i < 3; i++){
11             deep[i] = new Deep();
12             deep1[i] = new Deep();
13         }
14         System.out.println(Arrays.equals(deep, deep1));
15         System.out.println(Arrays.deepEquals(deep, deep1));
16     }
17 }
```

checking checking checking true checking checking checking true

Comparing of arrays of objects

```
1  public class Deep {
2
3      public static void main(String[] args) {
4          Deep[] deep = new Deep[3];
5          Deep[] deep1 = new Deep[3];
6          for (int i = 0; i < 3; i++){
7              deep[i] = new Deep();
8              deep1[i] = new Deep();
9          }
10         System.out.println(Arrays.equals(deep, deep1));
11         System.out.println(Arrays.deepEquals(deep, deep1));
12     }
13 }
```

false false

Deep comparing

```
1  public class Deep {
2     @Override
3      public boolean equals(Object o){
4          System.out.println("checking");
5          return true;
6      }
7      public static void main(String[] args) {
8          Deep[] deep = new Deep[3];
9          Deep[] deep1 = new Deep[3];
10         for (int i = 0; i < 3; i++){
11             deep[i] = new Deep();
12             deep1[i] = new Deep();
13         }
14         Deep [][] d1 = {deep, deep};
15         Deep [][] d2 = {deep1, deep1};
16
17         System.out.println(Arrays.equals(d1, d2));
18         System.out.println(Arrays.deepEquals(d1, d2));
19     }
20 }
```

Deep comparing

false
checking
checking
checking
checking
checking
checking
true

Deep comparing

```
1  public class Deep {
2      @Override
3       public boolean equals(Object o){
4           System.out.println("checking");
5           return true;
6       }
7       public static void main(String[] args) {
8           Deep obiekt = new Deep();
9
10          Deep[] tablica1 = {obiekt};
11          Deep[] tablica2 = {obiekt};
12
13          Deep [][] tablica3 = {tablica1, tablica1};
14          Deep [][] tablica4 = {tablica2, tablica2};
15
16          System.out.println(Arrays.equals(tablica3, tablica4));
17          System.out.println(Arrays.deepEquals(tablica3, tablica4));
18      }
19 }
```

false true

Deep comparing

```
1  public class Deep {
2     @Override
3      public boolean equals(Object o){
4          System.out.println("checking");
5          return true;
6      }
7      public static void main(String[] args) {
8          Deep[] pusta = {null, null , null };
9          Deep[] pusta1 = {null, null , null };
10
11         Deep [][] pusta2 = {pusta, pusta};
12         Deep [][] pusta3 = {pusta1, pusta1};
13
14         System.out.println(Arrays.equals(pusta2, pusta3));
15         System.out.println(Arrays.deepEquals(pusta2, pusta3));
16     }
17 }
```

false true

THE END!

Additional reading:

- https://docs.oracle.com/javase/tutorial/java/javaOO/objectcreation.htm
- https://docs.oracle.com/javase/tutorial/java/javaOO/initial.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/constructors.html
- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
- Thinking in Java chapters by Bruce Eckel:
  - Initialization & Cleanup
  - Arrays