Top 10 most viewed questions on Stack Overflow

* State: October 2019

Armin Kerscher

mail@javahochzwei.de

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java 'for each' loop work?”
11. Add-on: “Is Java 'pass-by-reference' or 'pass-by-value'?”

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’?”

- There are two different ways to convert a `String` to an `int`:
  - By invoking the method `Integer.parseInt`:

    ```
    public static int parseInt(String s)throws NumberFormatException
    ```
  - By invoking the method `Integer.valueOf`:

    ```
    public static Integer valueOf(String s)throws NumberFormatException
    ```
- Both methods can process either negative or positive numbers.
- They throw the unchecked `NumberFormatException`.
  Depending on what software you are developing, it could be usefull to catch it and precess it.

- Example of converting a `String`:

```java
try {
    int number1 = Integer.parseInt("-16");
} catch (NumberFormatException e) {
    // Exception handling
}

try {
    Integer number2 = Integer.valueOf("-16");
} catch (NumberFormatException e) {
    // Exception handling
}

// java.lang.NumberFormatException: For input string: "--16"
try {
    Integer.valueOf("--16");
} catch (NumberFormatException e) {
    e.printStackTrace();
}
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’?”

- There are several approaches for creating arrays in Java.
  1. Creation of an empty but defined size array.
     Also possible in multidimensional. The size of elements of each dimension is fix.
  2. Creation of an array by defining its content.
     The size of the array will be computed by the given content.
     Also possible in multidimensional, but the size of elements may differ.
  3. Creation of an initialized array by invoking the `toArray`-method of a `java.util.Stream` (only possible in one-dimensional).

- Example (1/2)

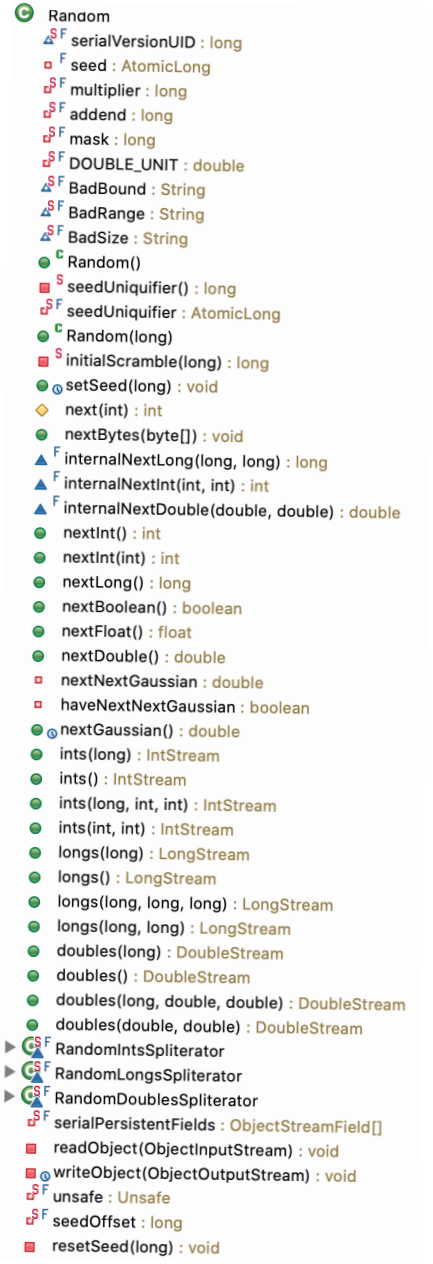
```java
// Creation:
int[] intArray1 = new int[5];
Integer[] intArray1_2 = new Integer[5];
int[] intArray2 = new int[]{1, 2, 3};
int[] intArray3 = {1, 2, 3};

int[][] intArray4 = new int[2][2];
Integer[][] intArray4_2 = new Integer[2][2];
int[][] intArray5 = new int[][]{{1, 2}, {3, 4}};
int[][] intArray5_2 = new int[][]{{1}, {1, 2}, {1, 2, 3}};
int[][] intArray6 = {{1, 2}, {3, 4}};
int[][][] intArray6_2 = {{{0}}, {{1}, {1, 2}, {1, 2, 3}}};

// Different notations:
int[] notationAsUsual = {1};
int notationAlsoFine[] = {1};

// Editing:
int number = 0;
for (int i = 0; i < intArray4.length; i++) {
    for (int j = 0; j < intArray4[i].length; j++) {
        intArray4[i][j] = number;
        number += 10;
    }
}
```

- Example (2/2):

```java
// Reading:
for (int i = 0; i < intArray4.length; i++) {
    for (int j = 0; j < intArray4[i].length; j++) {
        System.out.println(String.format("intArray4[%d][%d]: %d", i, j, intArray4[i][j]));
    }
}
// Output:
// intArray4[0][0]: 0
// intArray4[0][1]: 10
// intArray4[1][0]: 20
// intArray4[1][1]: 30


// Creation via Stream (since Java 8):
int[] intArray7 = IntStream.range(1, 5).toArray();
for (int i = 0; i < intArray7.length; i++) {
    System.out.println(String.format("intArray7[%d]: %d", i, intArray7[i]));
}
// Output:
// intArray7[0]: 1
// intArray7[1]: 2
// intArray7[2]: 3
// intArray7[3]: 4

// Different sizes within the same dimension
for (int i = 0; i < intArray5_2.length; i++) {
    System.out.println(String.format("intArray5_2[%d].length: %d", i, intArray5_2[i].length));
}
// Output:
// intArray5_2[0].length: 1
// intArray5_2[1].length: 2
// intArray5_2[2].length: 3
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’?”

- The central class for generating random numbers is `java.util.Random`.
- It could create random numbers of different types, for example `int`, `double`, `long` and so on.
- But as the name "Random" suggests, the class is not limited in generating numbers. You can also generate random `boolean` for example.
- How to generate random numbers in a specific range?
  - Integer: Use the method `nextInt(bound)` (with `[0, bound[`).
  - Floating-point number: `nextDouble()*bound` (with `[0, bound]`)



- Beispiel random numbers:

```java
Random random = new Random();

// Random ints
for (int i = 1; i <= 50; i += 10) {
    System.out.println(String.format("Random int in range [0, %d[: %d", i, random.nextInt(i)));
}
// Example for an output:
// Random int in range [0, 1[: 0
// Random int in range [0, 11[: 4
// Random int in range [0, 21[: 20
// Random int in range [0, 31[: 7
// Random int in range [0, 41[: 2

// Random doubles
for (int i = 1; i <= 50; i += 10) {
    System.out.println(String.format("Random double in range [0, %d[: %f", i, random.nextDouble() * i));
}
// Output:
// Random double in range [0, 1[: 0,454714
// Random double in range [0, 11[: 9,845851
// Random double in range [0, 21[: 2,244591
// Random double in range [0, 31[: 22,801551
// Random double in range [0, 41[: 20,328762
```

- Another usual approach is to invoke `java.util.Math#random()`:

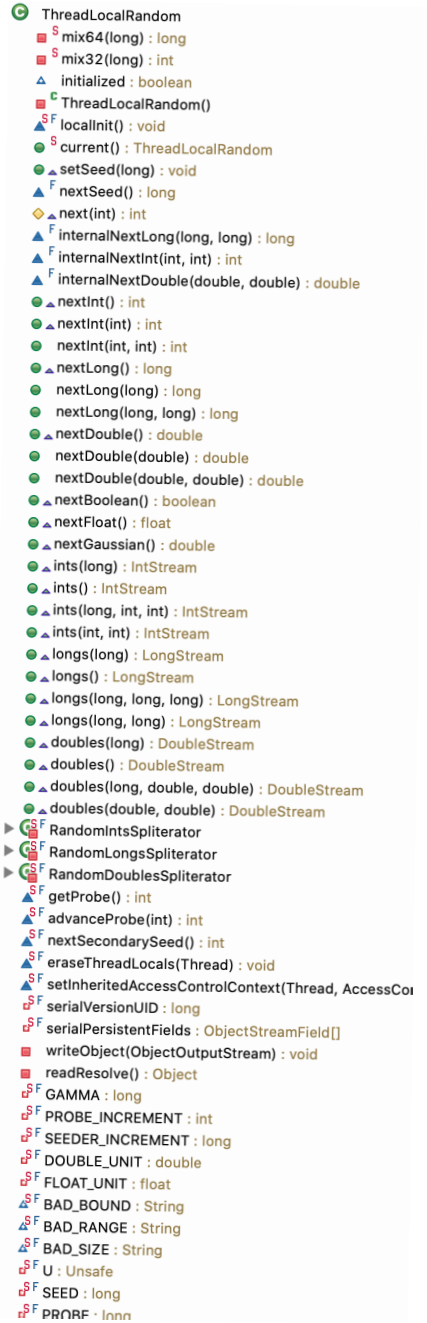
```java
public static double random()
```

Internally, it is also working with an instance of `Random` class, calling the `nextDouble()`-method.

- In multi-threaded applications, where multiple threads should generate random numbers, it is not advisable to use the same `Random` instance.
- Despite instances of this class are threadsafe.
- But you could face performance degredation if you do so nevertheless.
- It is written in the API of `Random`:

```
* <p>Instances of {@code java.util.Random} are threadsafe.
* However, the concurrent use of the same {@code java.util.Random}
* instance across threads may encounter contention and consequent
* poor performance. Consider instead using
* {@link java.util.concurrent.ThreadLocalRandom} in multithreaded
* designs.
```

- The class `ThreadLocalRandom` extends the class `Random` and is need to be used in these cases.



- Example for multi-threaded generation of random numbers:

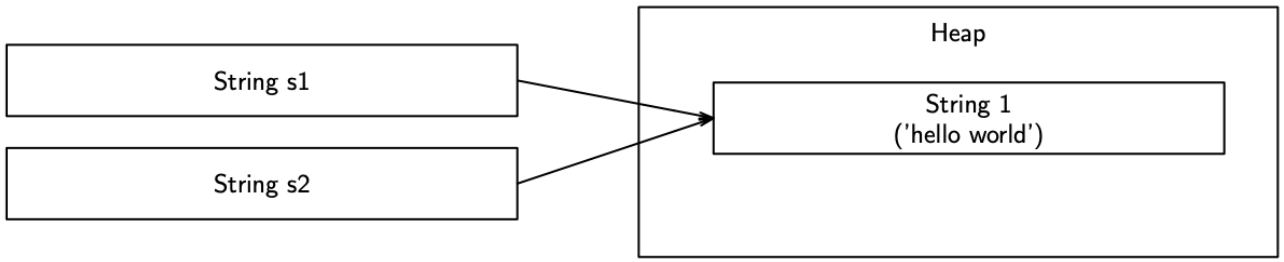
```
1  Random random = new Random();
2
3  ExecutorService newFixedThreadPool = Executors.newFixedThreadPool(100);
4
5  ArrayList<Callable<Integer>> callables = new ArrayList<>();
6  for (int i = 0; i < 1_000_000; i++) {
7      callables.add(() -> random.nextInt());
8  }
9
10 LocalTime now = LocalTime.now();
11 try {
12     newFixedThreadPool.invokeAll(callables);
13 } catch (InterruptedException e) { e.printStackTrace(); }
14
15 System.out.println(ChronoUnit.MILLIS.between(now, LocalTime.now())); // e.g. 1042
16
17
18 newFixedThreadPool = Executors.newFixedThreadPool(100);
19 callables = new ArrayList<>();
20
21 for (int i = 0; i < 1_000_000; i++) {
22     callables.add(() -> ThreadLocalRandom.current().nextInt());
23 }
24
25 now = LocalTime.now();
26 try {
27     newFixedThreadPool.invokeAll(callables);
28 } catch (InterruptedException e) { e.printStackTrace(); }
29
30 System.out.println(ChronoUnit.MILLIS.between(now, LocalTime.now())); // e.g. 619
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’?”

- There are two different approaches to compare two `Strings` (or in general `Objects`) with each other:
  - By using the operator: ==
  - By using the `equals()`-method of class `Object`.
- The ==-operator compares two references with each other.
- The `equals()`-method checks if the “content” of two objects is the same.

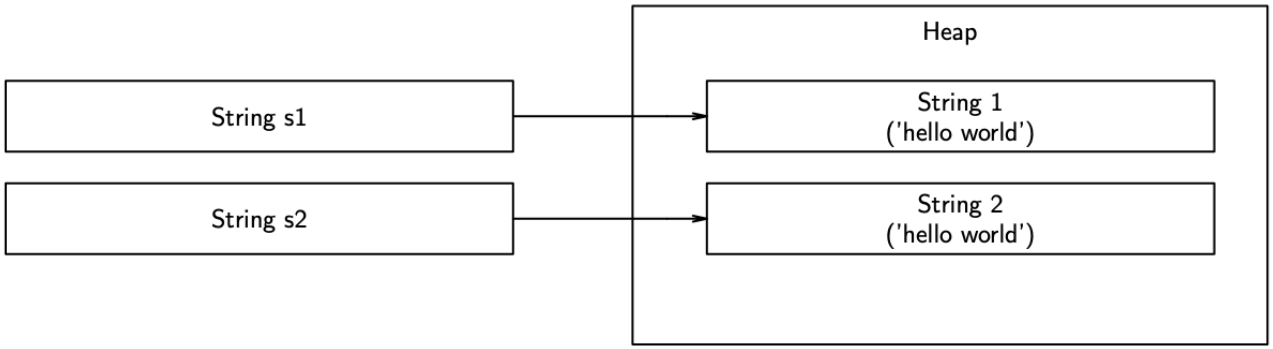
- With the help of the ==-operator you can compare two references with eachother, if they are pointing on the same `Object` on the heap.
- Example (1/2)

```java
public static void main(String[] args) {

    String s1 = new String("hello world");
    String s2 = s1;

    boolean sameReference = s1 == s2; // true

    System.out.println("s1 == s2? " + sameReference);
```



- Example (2/2)

```
1
2       s2 = new String("hello world");
3
4       sameReference = s1 == s2; // false
5
6       System.out.println("s1 == s2? " + sameReference);
7   }
```



- To compare the “content” of two objects, you can use the `equals`-method.
  This comparison is independently to the references.
- For example:

```java
public static void main(String[] args) {

    String s1 = new String("hello world");
    String s2 = s1;

    boolean equals = s1.equals(s2); // true

    System.out.println("s1.equals(s2)? " + equals);


    s2 = new String("hello world");

    equals = s1.equals(s2); // true

    System.out.println("s1.equals(s2)? " + equals);
}
```

- Beware of `NullPointerExceptions`!
- Example:

```
1  String s3 = null;
2  String s4 = null;
3
4  System.out.println("s3 == s4? " + (s3 == s4)); // true
5
6  System.out.println("s3.equals(s4)? " + s3.equals(s4)); // NullPointerException
7
8  System.out.println(Objects.equals(s3, s4)); // true
```

- Since Java version 7 you can use the class `Objects` and you will always be *NPE-safe*:

```
public static boolean equals(Object a, Object b) {
    return (a == b) || (a != null && a.equals(b));
}
```

- A little trap: The so called *String pool*.
- Example:

```
1 String s5 = "hello world";
2 String s6 = "hello world";
3 String s7 = new String("hello world");
4
5 System.out.println("s5 == s6? " + (s5 == s6)); // true
6 System.out.println("s6 == s7? " + (s6 == s7)); // false
```

- The expression in line 5 is true because `String s5` and `String s6` are pointing to the same `String` instance within in the *String pool*.
- The statement in line 3 explicitly creates an own object of the type `String` that is stored in the heap.

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’?”

- To split a `String` into different parts one can use the method `java.lang.String#split`:

```
public String[] split(String regex)
```

- The `String` is splitted based on a regular expression.
- The resulting parts are returned as a `String` array.

- Example for splitting a `String`:

```java
// source of regex: https://stackoverflow.com/questions/8204680/java-regex-email
Pattern validMailAdressRegex =
    Pattern.compile("^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$", Pattern.CASE_INSENSITIVE);

String s = "hans.mustermann@gmail.com";

if (validMailAdressRegex.matcher(s).find()) {

    for (String part : s.split("[@]")) {
        System.out.println("-1-> " + part);
    }

    for (String part : s.split("[.@]")) {
        System.out.println("-2-> " + part);
    }

    for (String part : Pattern.compile("[@]").split(s)) {
        System.out.println("-3-> " + part);
    }
} else {
    throw new IllegalArgumentException("Given string is not a valid mail adress");
}
// Output:
// -1-> hans.mustermann
// -1-> gmail.com
// -2-> hans
// -2-> mustermann
// -2-> gmail
// -2-> com
// -3-> hans.mustermann
// -3-> gmail.com
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’?”

- A `java.util.Map` is an amount of key-value-pairs.
- These pairs are represented by the interface `java.util.Map.Entry`.
- There are several ways to iterate though a `Map`:
  - Iterate through all entries:

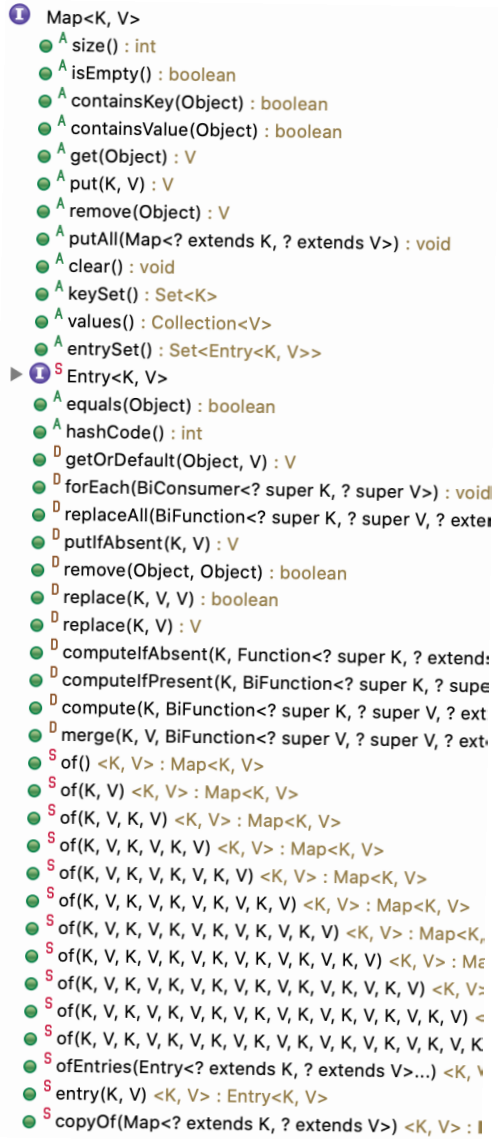
    ```
    Set<Map.Entry<K, V>> entrySet()
    ```

  - Iterate through all keys:

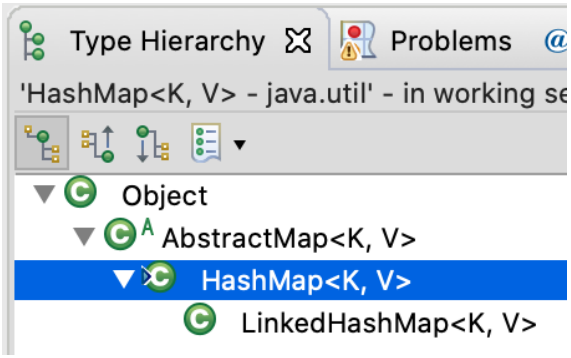
    ```
    Set<K> keySet()
    ```

  - Iterate through all values:

    ```
    Collection<V> values()
    ```



- The class `java.util.HashMap` is maybe the most used implementation of `Map`.
- It extends `java.util.AbstractMap`, which already implements the interfaces `Map` and `Map.Entry`:



- Example for iterating through a `Map`:

```java
private static void iterateThroughMap(Map<?, ?> map) {
    for(Map.Entry<?, ?> entry : map.entrySet()) {
        System.out.println(String.format("Key=%s, value=%s", entry.getKey(), entry.getValue()));
    }
}
public static void main(String[] args) {
    Map<Integer, String> testMap = new HashMap<>();
    testMap.put(0, "zero");
    testMap.put(1, "one");
    testMap.put(10, "ten");
    testMap.put(100, "hundred");
    iterateThroughMap(testMap);

    System.out.println("------");

    Map<String, Object> testMap2 = new TreeMap<>();
    testMap2.put("one", new String("one"));
    testMap2.put("two", Integer.valueOf(2));
    testMap2.put("five", Integer.parseInt("5"));
    iterateThroughMap(testMap2);

    // Output:
    // Key=0, value=zero
    // Key=1, value=one
    // Key=100, value=hundred
    // Key=10, value=ten
    // ------
    // Key=five, value=5
    // Key=one, value=one
    // Key=two, value=2
}
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java 'for each' loop work?”
11. Add-on: “Is Java 'pass-by-reference' or 'pass-by-value'?”

- There are various opportunities to create lists (implementations of the interface `java.util.List`):
  1. Invoking one ot the methods `List.of()`, `List.of(E)`, `List.of(E, E)`, etc.:

     ```
     static <E> List<E> of()
     ```

  2. Invoking the method `Arrays.asList(...)`:

     ```
     public static <T> List<T> asList(T... a)
     ```

  3. u.v.m.
- These methods don’t return `ArrayLists`, but special implementations of the `List` interface.

1) Excerpt of the API of `List#of`:

```
/**
 * Returns an unmodifiable list containing zero elements.
 *
 * See <a href="#unmodifiable">Unmodifiable Lists</a> for details.
 *
 * @param <E> the {@code List}'s element type
 * @return an empty {@code List}
 *
 * @since 9
 */
static <E> List<E> of() {
    return ImmutableCollections.emptyList();
}
```

q.v.: oracle.com/.../List.html#unmodifiable

2) Excerpt of the API of `Arrays#asList`:

```
/**
 * Returns a fixed-size list backed by the specified array.  (Changes to
 * the returned list "write through" to the array.)  This method acts
 * as bridge between array-based and collection-based APIs, in
 * combination with {@link Collection#toArray}.  The returned list is
 * serializable and implements {@link RandomAccess}.
 *
 * <p>This method also provides a convenient way to create a fixed-size
 * list initialized to contain several elements:
 * <pre>
 *     List&lt;String&gt; stooges = Arrays.asList("Larry", "Moe", "Curly");
 * </pre>
 *
 * @param <T> the class of the objects in the array
 * @param a the array by which the list will be backed
 * @return a list view of the specified array
 */
@SafeVarargs
@SuppressWarnings("varargs")
public static <T> List<T> asList(T... a) {
    return new ArrayList<>(a);
}
```

- To initialize an `ArrayList` and add data to it on the same line, there is the opportunity to use the following constructor:

```
public ArrayList(Collection<? extends E> c)
```

- Due to the fact that the previous `Lists` are all `Collections`, it is possible to combine both approaches to get the claimed behaviour in one line.
- The result of this call is an usual `ArrayList`, although the passed `Lists` were unmodifable or fixed-size.

- Example for the initialization of an `ArrayList` in one line:

```java
List<String> unmodifiableList = List.<String>of("hello", "world");
// java.lang.UnsupportedOperationException
//unmodifiableList.add("test");
//java.lang.UnsupportedOperationException
//unmodifiableList.remove(0);

// one line creation and initialization
ArrayList<String> arrayList = new ArrayList<>(List.of("hello", "world"));
// ok
arrayList.add("test");
//ok
arrayList.remove(0);


List<String> fixedSizeList = Arrays.asList("hello", "world");
// ok
fixedSizeList.set(0, "new hello");
// java.lang.UnsupportedOperationException
// fixedSizeList.remove(0);

// one line creation and initialization
arrayList = new ArrayList<>(Arrays.asList("hello", "world"));
// ok
arrayList.add("test");
//ok
arrayList.remove(0);
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java 'for each' loop work?”
11. Add-on: “Is Java 'pass-by-reference' or 'pass-by-value'?”

- Also for this requirement, there are different approaches:
  1. To simply create an empty file, you can use `File#createNewFile`:

     ```
     public boolean createNewFile()throws IOException
     ```

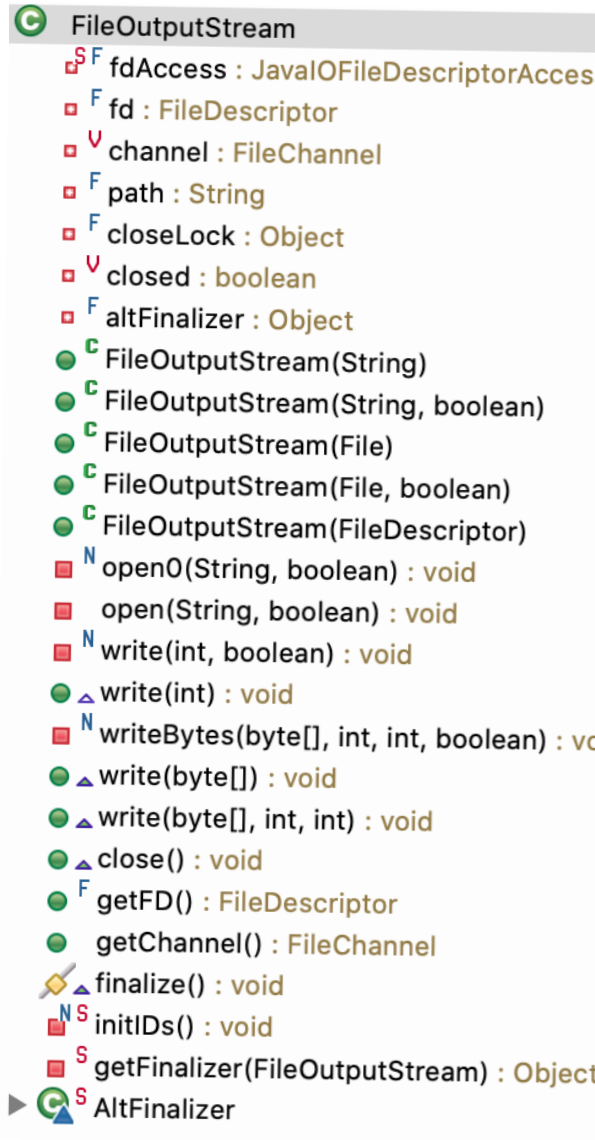
  2. If the task is to create and fill a file with data, there are these classes:
     - 2.1 `java.io.FileOutputStream`
     - 2.2 `java.io.PrintWriter`

- You can pass an instance of `File` to the constructor of `java.io.FileOutputStream`:

```
public FileOutputStream(File file)throws FileNotFoundException
```

- This invocation writes a file to the file system.
- Via the method `write` it is possible to write bytes to the stream:

```
public void write(byte b[])throws IOException
```



- The class `PrintWriter` has also a constrctor with a `File` parameter:

```
public PrintWriter(File file)throws FileNotFoundException
```

- These type of streams are character- or text-based. That's why the methods looks like that:

```
public void println(String x)

public void print(String s)

//etc.
```

- Example for creating a file and writing to it:

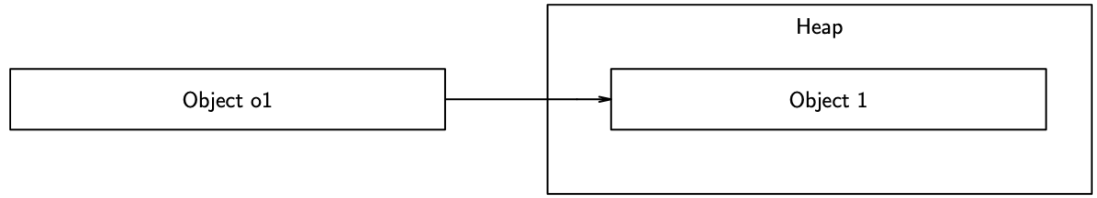
```java
// Simply create empty file, with size of 0 bytes
try {
    new File("emptyFile").createNewFile();
} catch (IOException e1) { e1.printStackTrace(); }

// Result: A file with size of 5 bytes
try(FileOutputStream out = new FileOutputStream(new File("FileOutputStreamTest"))) {

    byte[] byteArray = {1, 2, 3, 4, 5};
    out.write(byteArray);
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace(); }

// Result: A file with size of 16 bytes and with content:
//hello
//w
//o
//r
//l
//d
try(PrintWriter writer = new PrintWriter(new File("PrintWriterTest"))) {

    writer.print("hello");
    writer.println();
    "world".chars().forEach(eachChar -> writer.println(Character.toString(eachChar)));
} catch (FileNotFoundException e) {
    e.printStackTrace(); }
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java 'for each' loop work?”
11. Add-on: “Is Java 'pass-by-reference' or 'pass-by-value'?”

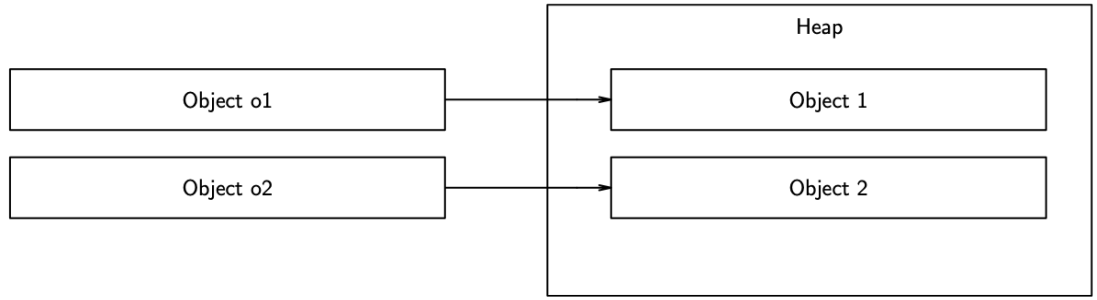
- The `NullPointerException` occurs, if you try to call a method on `null` reference.
- What is a `null` reference?

- Example for references (1/2):

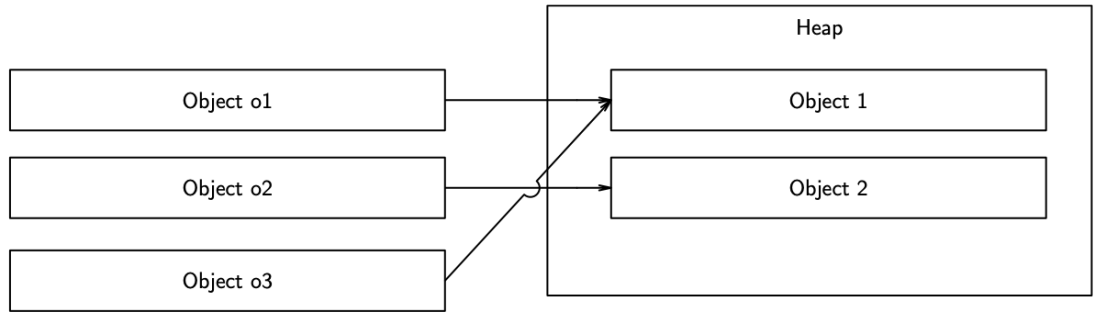
```
1 Object o1 = new Object(); // Object 1
```




```
1 Object o1 = new Object(); // Object 1
2 Object o2 = new Object(); // Object 2
```

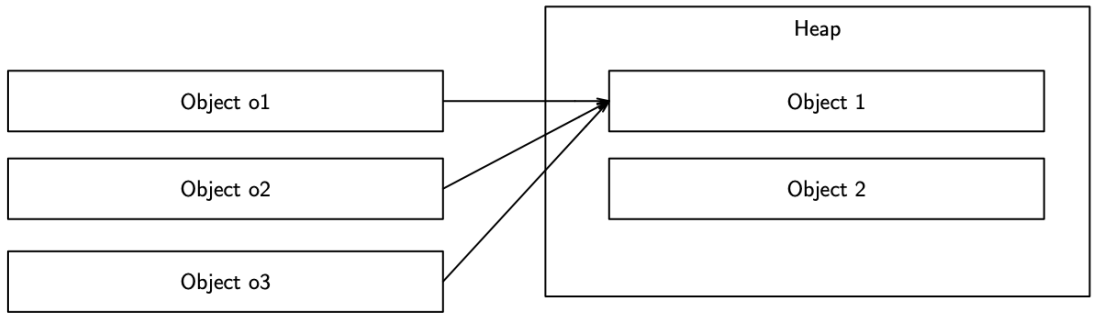



```
1 Object o1 = new Object(); // Object 1
2 Object o2 = new Object(); // Object 2
3 Object o3 = o1;
```

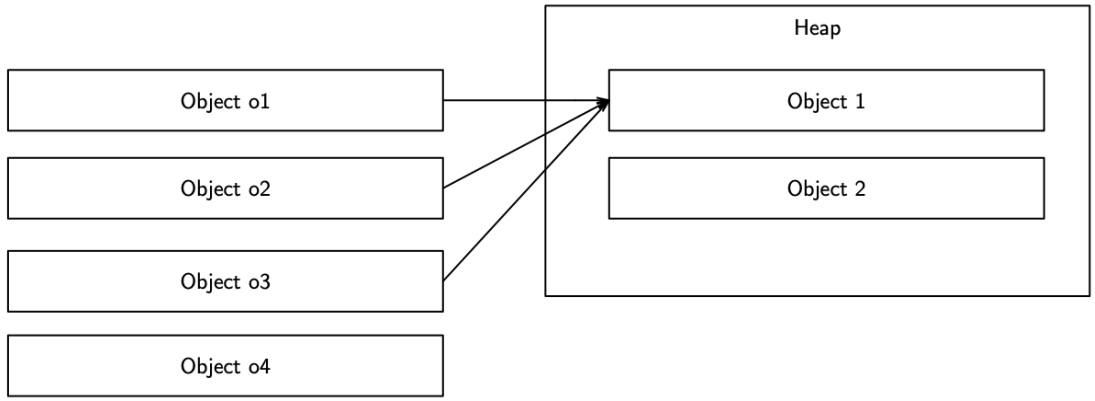


- Example for references (2/2):

```
1 Object o1 = new Object(); // Object 1
2 Object o3 = o1;
3 Object o2 = o3;
4 // or Object o2 = o1;
```




```
1 Object o1 = new Object(); // Object 1
2 Object o2 = o1;
3 Object o3 = o1;
4 Object o4  = null;
```



- How to provoke a `NullPointerException`:

```
1  public static void main(String[] args) {
2      Object o1 = new Object(); // Object 1
3      Object o2 = o1;
4      Object o3 = o1;
5      Object o4  = null;
6
7      System.out.println(o1.toString()); // java.lang.Object@6e8dacdf
8      System.out.println(o2.toString()); // java.lang.Object@6e8dacdf
9      System.out.println(o3.toString()); // java.lang.Object@6e8dacdf
10
11     // Exception in thread "main" java.lang.NullPointerException
12     System.out.println(o4.toString());
13 }
```

Question: How do I avoid such `NullPointerExceptions`?

- Such execptions may occur everywhere where references in general are used.
- Because this is quite everywhere, there isn't a single answer on the question on how avoiding these exceptions.
- An usual situation where NPEs may occur are methods which have parameters.
  The references which are committed to these methods may be null.
- Example for how to avoid a `NullPointerException`:

```
1 private static String convertToUpperstring(String s) {
2     if (s == null) {
3         throw new IllegalArgumentException("Argument may not be null");
4     }
5     // impossible that a NPE occurs here
6     return s.toUpperCase();
7 }
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java 'for each' loop work?”
11. Add-on: “Is Java 'pass-by-reference' or 'pass-by-value'?”

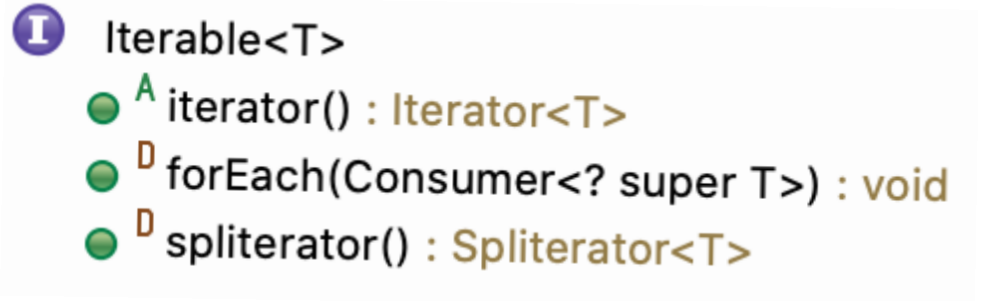
- A “for each loop” is an syntactical abbreviation.
- It supersedes the explicit use of the iterator of an given data structure.
- It was announced in Java 5.
- Example

```
1  for (Iterator<String> i = someIterable.iterator(); i.hasNext();) {
2       String nextItem = i.next();
3  }
4
5  // respectively
6
7  Iterator<String> iterator = someIterable.iterator();
8  while (iterator.hasNext()) {
9      String nextItem = iterator.next();
10 }
```

- With the help of the for each loop it is much easier:

```
1  for (String nextItem : someIterable) {
2      // do something with the item
3  }
```

- Every class that should be usable in such a loop, has to implement the interface `Iterable`.
- Arrays are the only exception to this. They are also usable in a for each loop!



- Example for each loop (1/2):

```
1  private static class MyStringContainer implements Iterable<String> {
2    private List<String> strings;
3
4    public MyStringContainer() {
5      strings = new ArrayList<>(); }
6
7    public void add(String s) {
8      strings.add(s); }
9
10   @Override
11   public Iterator<String> iterator() {
12     return strings.iterator(); }
13 }
14
15 public static void main(String[] args) {
16
17   MyStringContainer myStringContainer = new MyStringContainer();
18
19   myStringContainer.add("hello");
20   myStringContainer.add("world");
21
22   for (String s : myStringContainer) {
23     System.out.println("-> " + s);
24   }
25   // Output:
26   // -> hello
27   // -> world
28 }
```

- Example for each loop (2/2):

```java
public static void main(String[] args) {

    MyStringContainer myStringContainer = new MyStringContainer();

    myStringContainer.add("hello");
    myStringContainer.add("world");

    for (String s : myStringContainer) {
        System.out.println("-> " + s);
    }
    // Output:
    // -> hello
    // -> world

    // instead of using the iterator explicitly:
    Iterator<String> iterator = myStringContainer.iterator();

    while (iterator.hasNext()) {
        System.out.println("---> " + iterator.next());
    }
    // Output:
    // ---> hello
    // ---> world
}
```

1. “How do I convert a String to an int in Java?”
2. “How do I declare and initialize an array in Java?”
3. “How do I generate random integers within a specific range in Java?”
4. “How do I compare strings in Java?”
5. “How to split a string in Java”
6. “Iterate through a HashMap”
7. “Initialization of an ArrayList in one line”
8. “How do I create a file and write to it in Java?”
9. “What is a NullPointerException, and how do I fix it?”
10. “How does the Java ’for each’ loop work?”
11. Add-on: “Is Java ’pass-by-reference’ or ’pass-by-value’ ?”

- Also known as “call by reference” and “call by value”.
- They describe different concepts on how an invoked method gets its parameter(s).

  Call by reference The parameter is an explicit reference to the object’s reference, which is passed to the method.

  Call by value The parameter is a copy of the value of the object’s reference, which is passed to the method.
- Java is always and only “call by value”!

- Since Java is “call by value”, the parameter of a method is a copy of the value of the passed variable.
- It is easy to demonstrate this for primitive types:

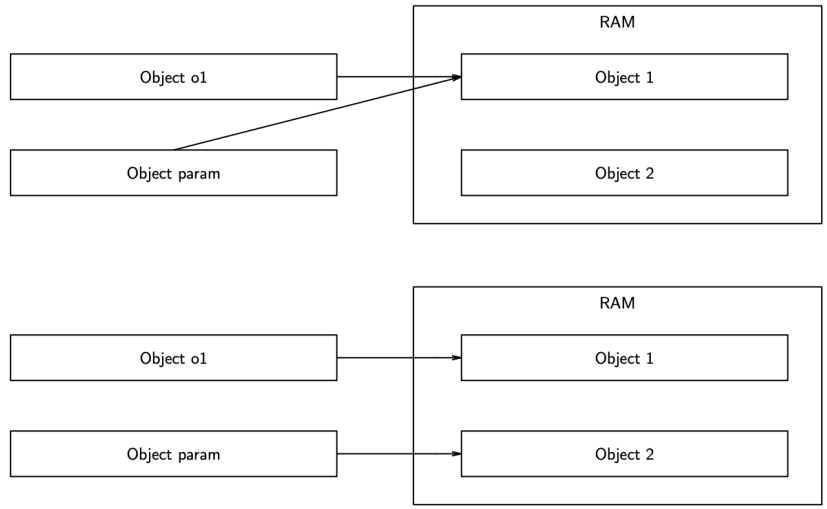
```
1  private static void increment(int paramI) {
2      paramI = paramI + 1;
3  }
4
5  public static void main(String[] args) {
6      int i = 0;
7      increment(i);
8      System.out.println(i); // 0
9  }
```

- Although `paramI` was incremented by 1, the value of the passed variable `i` is still 0.

- Example, how “call by value” works with objects (1/2):

```
1  private static void method(Object param) {
2      Object o2 = new Object();
3      System.out.println(o2); // java.lang.Object@65ae6ba4
4      param = o2;
5  }
6
7  public static void main(String[] args) {
8
9      Object o1 = new Object();
10     System.out.println(o1); // java.lang.Object@4d591d15
11     method(o1);
12     System.out.println(o1); // java.lang.Object@4d591d15
13 }
```

Process:

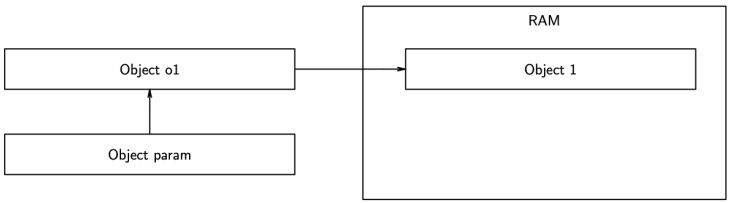


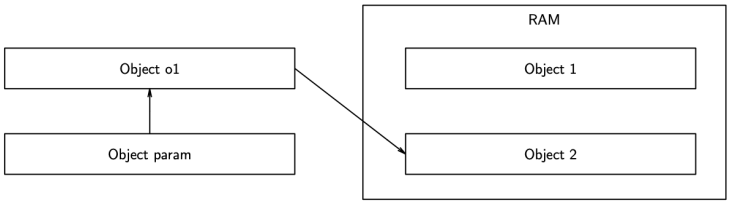
- Example, how “call by value” works with objects (2/2):

```
1  private static class Foo {
2      private String str;
3
4      public Foo(String str) {
5          this.str = str; }
6
7      public void setStr(String str) {
8          this.str = str; }
9
10     @Override
11     public String toString() {
12         return str; }
13 }
14
15 private static void method(Foo foo) {
16     foo = new Foo("Foo2"); }
17
18 private static void method2(Foo foo) {
19     foo.setStr("new string for foo"); }
20
21 public static void main(String[] args) {
22
23     Foo foo1 = new Foo("foo1");
24     method(foo1);
25     System.out.println(foo1); // foo1
26
27     // but:
28     method2(foo1);
29     System.out.println(foo1); // new string for foo
30 }
```

- If Java would be “call by reference“, then the following snippet would work as commented:

```
1  private static void method(Object param) {
2      Object o2 = new Object(); // Object 2
3      System.out.println(o2);
4      param = o2;
5  }
6
7  public static void main(String[] args) {
8      Object o1 = new Object();
9      System.out.println(o1); // Object1
10     methodo(o1);
11     System.out.println(o1); // Object2
12 }
```





- But the code works different!!

Thank you for your attention

I welcome suggestions and ideas from you

mail@javahochzwei.de