ЛЕКЦИЯ:
Packages & Information Hiding

- JAVA API

- Own packages

Information Hiding

- String manipulation

- Modifiers

22

Straight from Wikipedia:

> In computer science, an application programming interface (API) is an interface that defines the ways by which an application program may request services from libraries

Java contains many libraries in those packages (Swing, etc.), and the API is the interface by which we request services (perform actions, etc.).

share improve this answer

answered Aug 24 '09 at 14:48

geowa4

28.5k ● 13 ● 76 ● 104

14

The API (Application Programming Interface) is what a library looks like from the outside for a program that's using it. It's the "face" of a library to other programs. The API of a library is the set of publicly accessible classes, interfaces and methods in the library.

You wrote:

> I can connect to a class like System without using any API here...

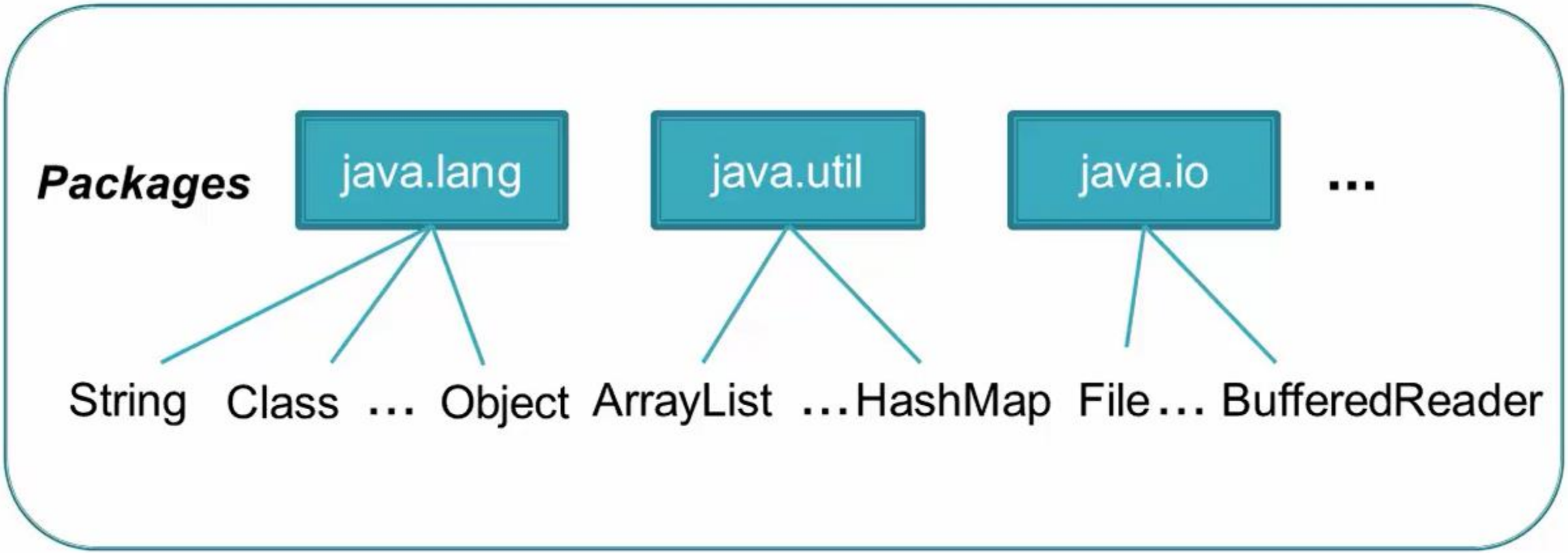
That's not true - class System is a public class in Java's standard library, so it's part of the API of the standard Java library. You *are* using the API of the standard Java library if you are using class System.

Why do you think you "are not using any API" when you use class System?

Java API

- Library of *well-tested* classes
- Java 8 ~ **4240** classes
- Developed by experts
- Used by *millions* of programmers
- Part of both JDK & JRE

Packages


Packages
java.lang
java.util
java.io
...
String
Class
...
Object
ArrayList
...
HashMap
File
...
BufferedReader

Why Packages?

- Meaningful organization

Why Packages?

- Meaningful organization
- Name scoping

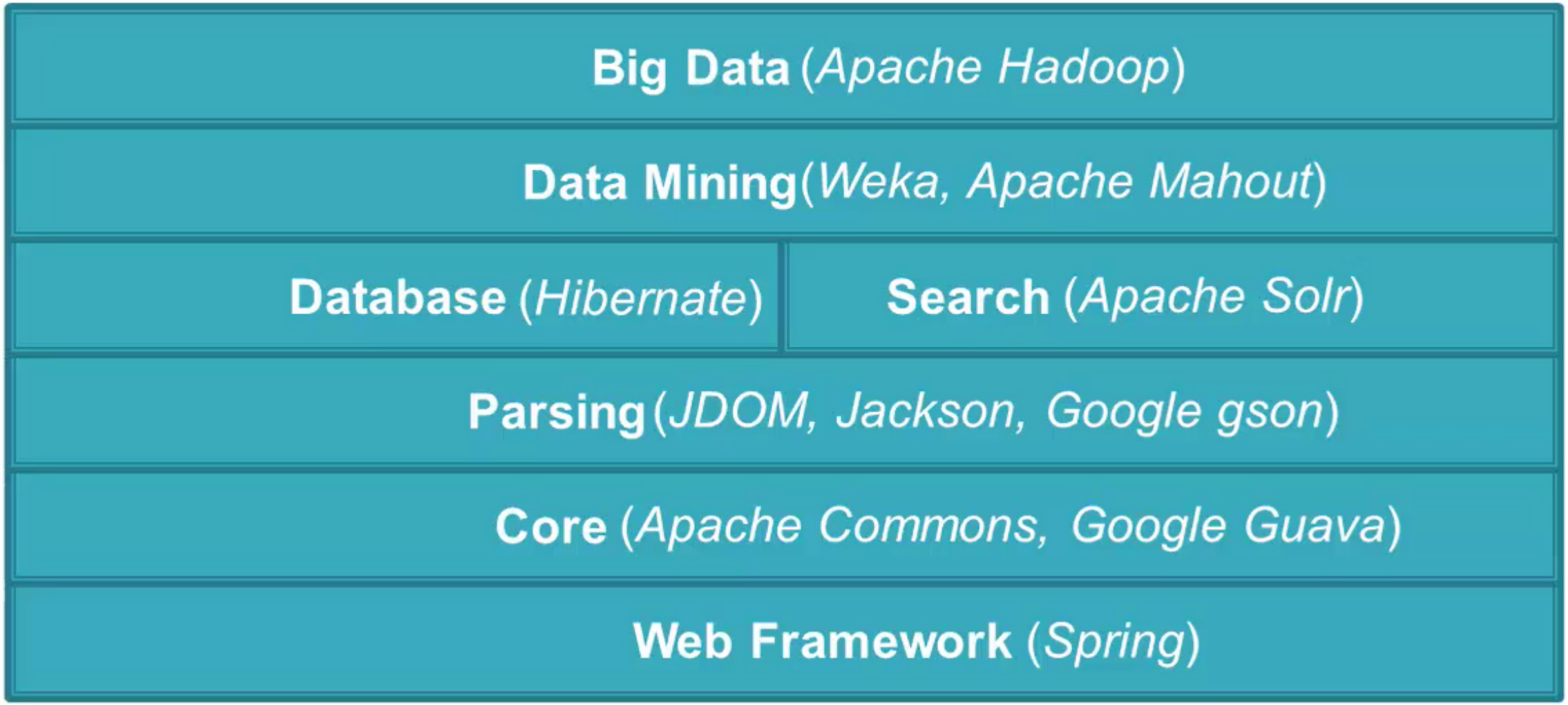
  java.util.Date *!=* java.sql.Date

- Security

Java API: Important Packages

- java.lang ~ Fundamental classes
- java.util ~ Data structures
- java.io ~ Reading & writing
- java.net ~ Networking
- java.sql ~ Databases

3rd Party APIs



- Spring: Popular framework for developing Web applications
- Apache Commons & Google Guava: Both of them include libraries for performing common programming tasks like string manipulation or using different data structures or using Math functions.
- JDOM: To parse XML documents
- Jackson or Google's gson: To parse JSON text. JSON is like XML but is more compact and in recent years, it is very commonly used to exchange data, e.g., Web services.
- Hibernate: Object-relational mapping (ORM) framework for interacting with databases

API Benefits

- Focus on writing new logic
- APIs performance over time
- Gain new functionality too

Използване на класове в JAVA

- Когато са от същия пакет – **директен** достъп
- От различни пакети
  - import
  - Цялото име на класа

import Statement

```
import java.util.ArrayList;

class FooClass {
   void foo() {
      ArrayList list = new ArrayList();
      …
   }
}
```

Importing Single vs Multiple Classes

- Import *single* class
  - Explicit import (or *Single-Type-Import*)
- Import *multiple* classes
  - Separate explicit import
  - * import (or *Import-On-Demand*)

* import

Imports *all* classes in a package

```
import java.util.*;
```

select * from <table_name>

Explicit import or * import?

- * import can ***break*** code

```
import java.util.*;
import java.sql.*;
…
Date date; // from util
```

+ *java.sql.Date* →

```
import java.util.*;
import java.sql.*;
…
Date date; // compiler error
```

- *Better clarity* with explicit import
- Explicit import seems to be preferred

Fully-qualified Class Name

- Alternative to *import*

```
java.util.ArrayList list = new java.util.ArrayList();
```

- Required if using *java.util.Date* & *java.sql.Date*

Solution 1

Use only **one explicit** import

```
import java.util.Date;


...

Date date; // from util
java.sql.Date date2;
```

Solution 2

Use **only fully-qualified** names

```
import java.util.*;
import java.sql.*;

…
java.util.Date date;
java.sql.Date date2;
```

Invalid Imports

```
import java.util.Date;
import java.sql.Date;
```

Any Side Affects in Using import?

Nope!!

- *Does not make your class bigger!!*
- *Does not affect runtime performance*
- Saves from typing fully-qualified name ~ compiler does this

java.lang is imported by ***default***

ДЕМО:

```
public static void main(String[] args) {
    // Language Basics 1
    //print();
    //primitives();
    //typeCasting();
    //arrays();
    //threeDimensionalArrays();
    /*varargsOverload(true, 1, 2, 3);
    varargsOverload(true, 1, 2, 3, 4, 5, 6, 7, 8);
    varargsOverload(true);*/
    //charTypePromotion();
    //bitwiseOperators();
    //switchExample();
    //labeledBreak();
    //labeledContinue();
    java.util.Date date = new java.util.Date();
    ArrayList list;

}
```

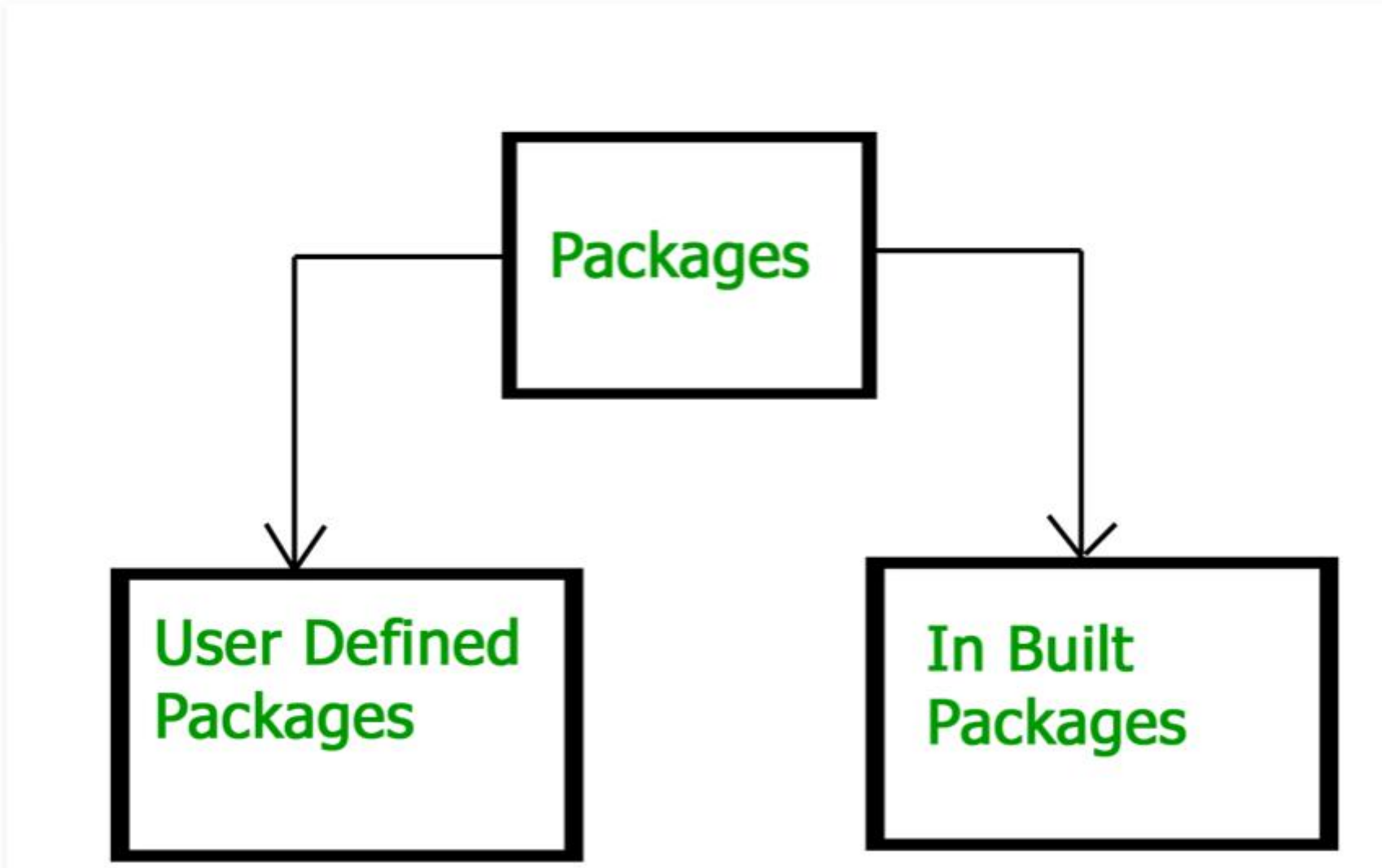
```
import java.util.Date;
import java.util.ArrayList;

class BasicsDemo {

Date date;
ArrayList list;
java.sql.Date date2;
```

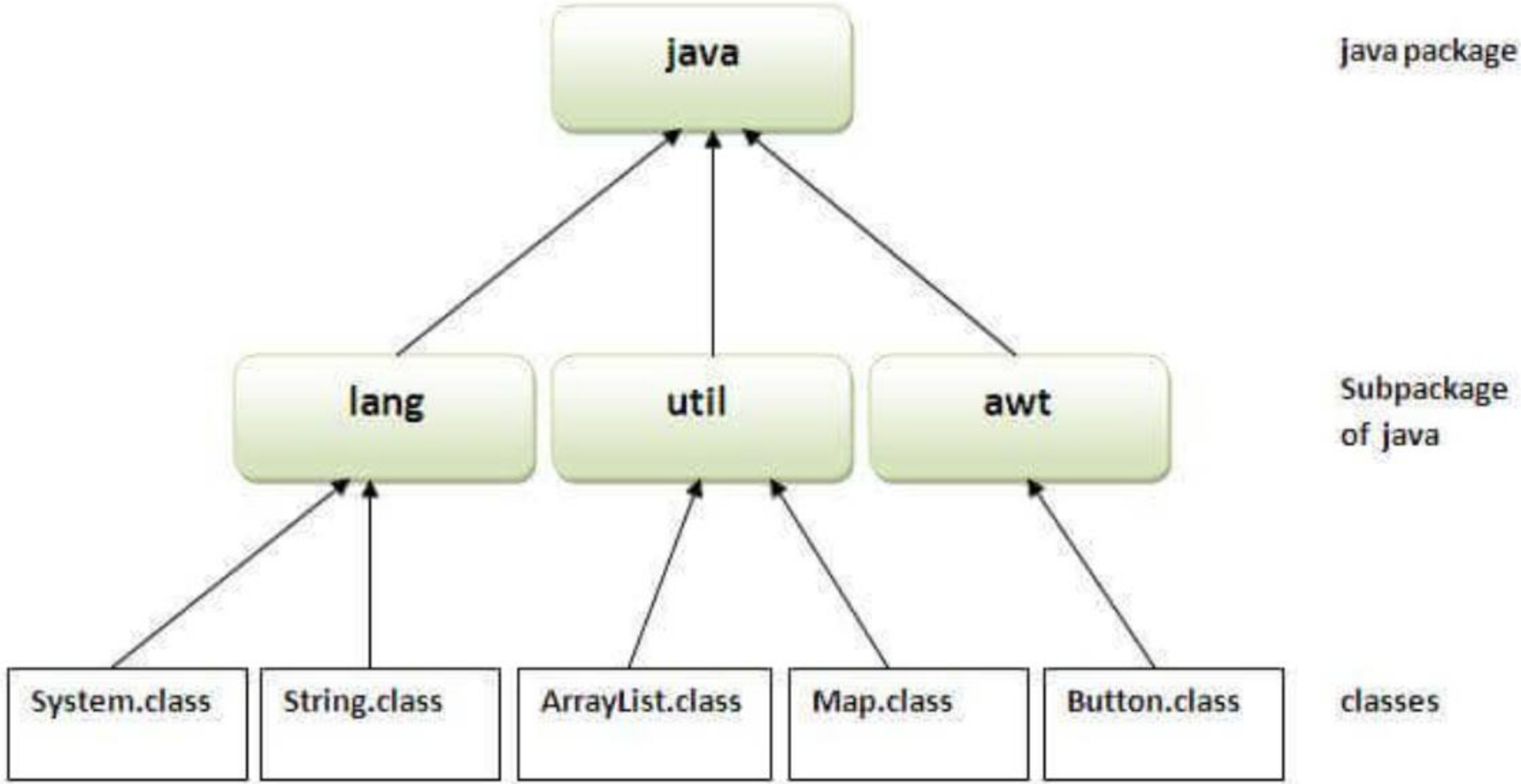
```
import java.util.Date;
import java.sql.Date;
import java.util.ArrayList;
```

```
Date date;
ArrayList list;
java.sql.Date date2
```

```
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
BasicsDemo.java:209: error: cannot find symbol
          Date date = new java.util.Date();
          ^
  symbol:   class Date
  location: class BasicsDemo
1 error

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java

C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
BasicsDemo.java:2: error: a type with the same simple name is already defined by
 the single-type-import of Date
  import java.sql.Date;
  ^
1 error

C:\javaindepth\src\com\semanticsquare\basics>
```

Types of packages:




java
lang
util
awt
System.class
String.class
ArrayList.class
Map.class
Button.class
Java package
Subpackage of java
classes

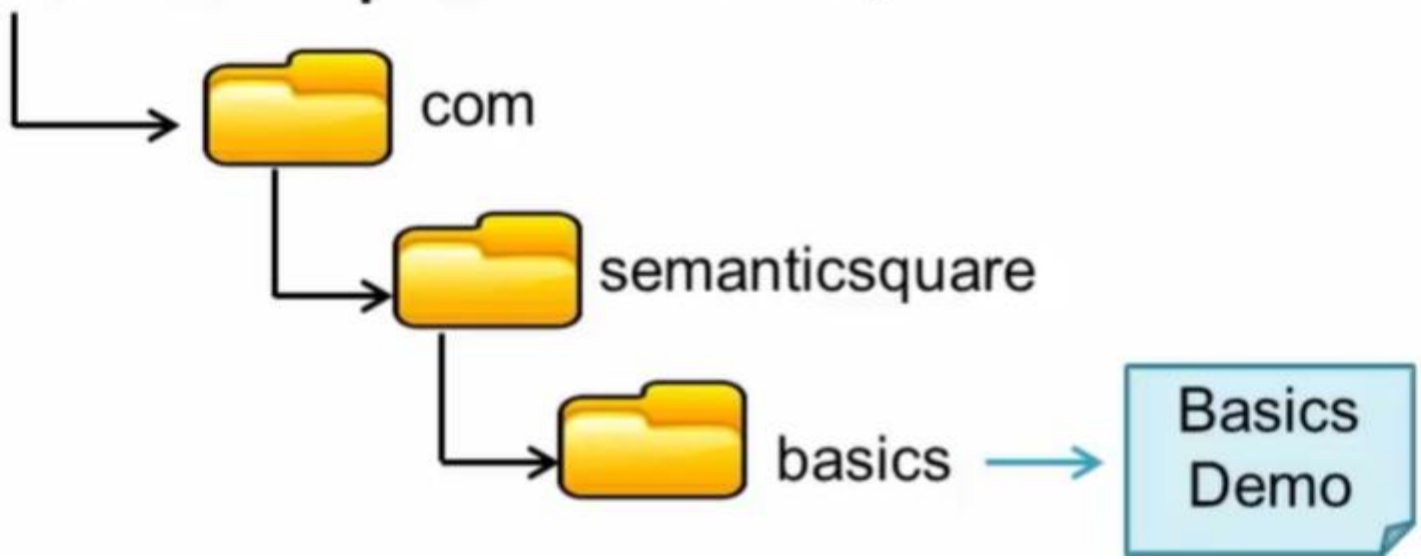
Създаване на собствен *package*

```
import java.util.Date;

com.semanticsquare
```

Set-up Matching Directory Structure

- basics;

- com.semanticsquare.basics;



package Statement

- **package** package-name;

```
package com.semanticsquare.basics;
```

- Must be **first** statement above any imports

```
package com.semanticsquare.basics;
import java.util.ArrayList;
class BasicsDemo {
   …
}
```

Class & Package

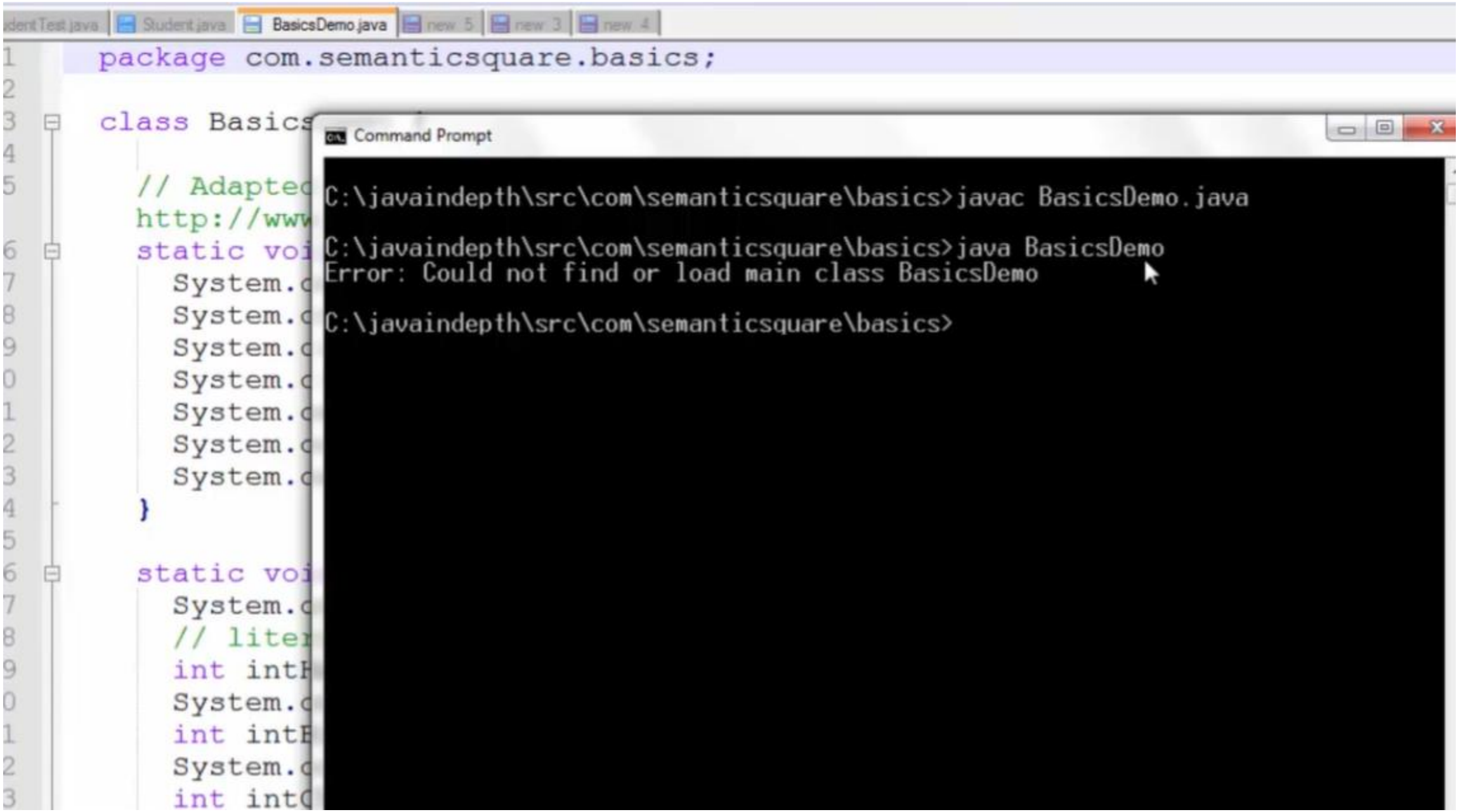
- ✓ Ensure *matching directory structure* exists
- ✓ Use *package* statement

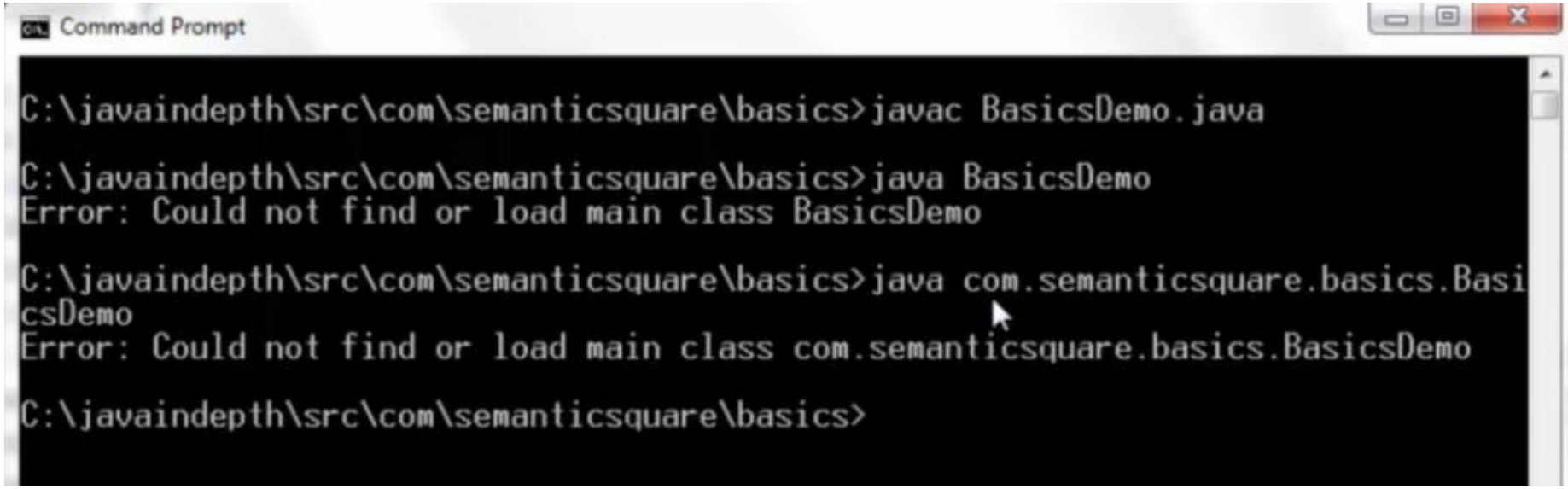
Effect of Creating Package

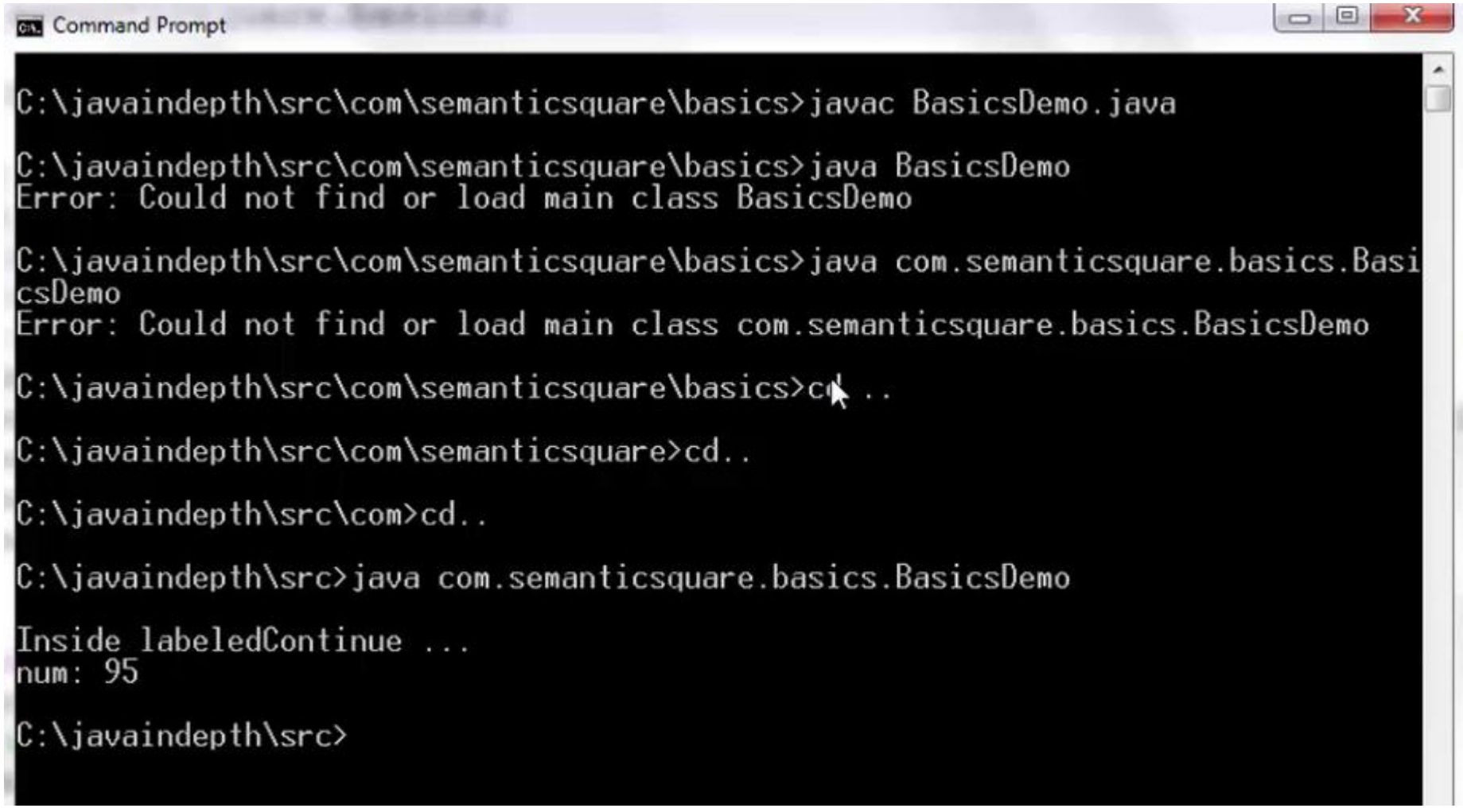
Package name is part of class name

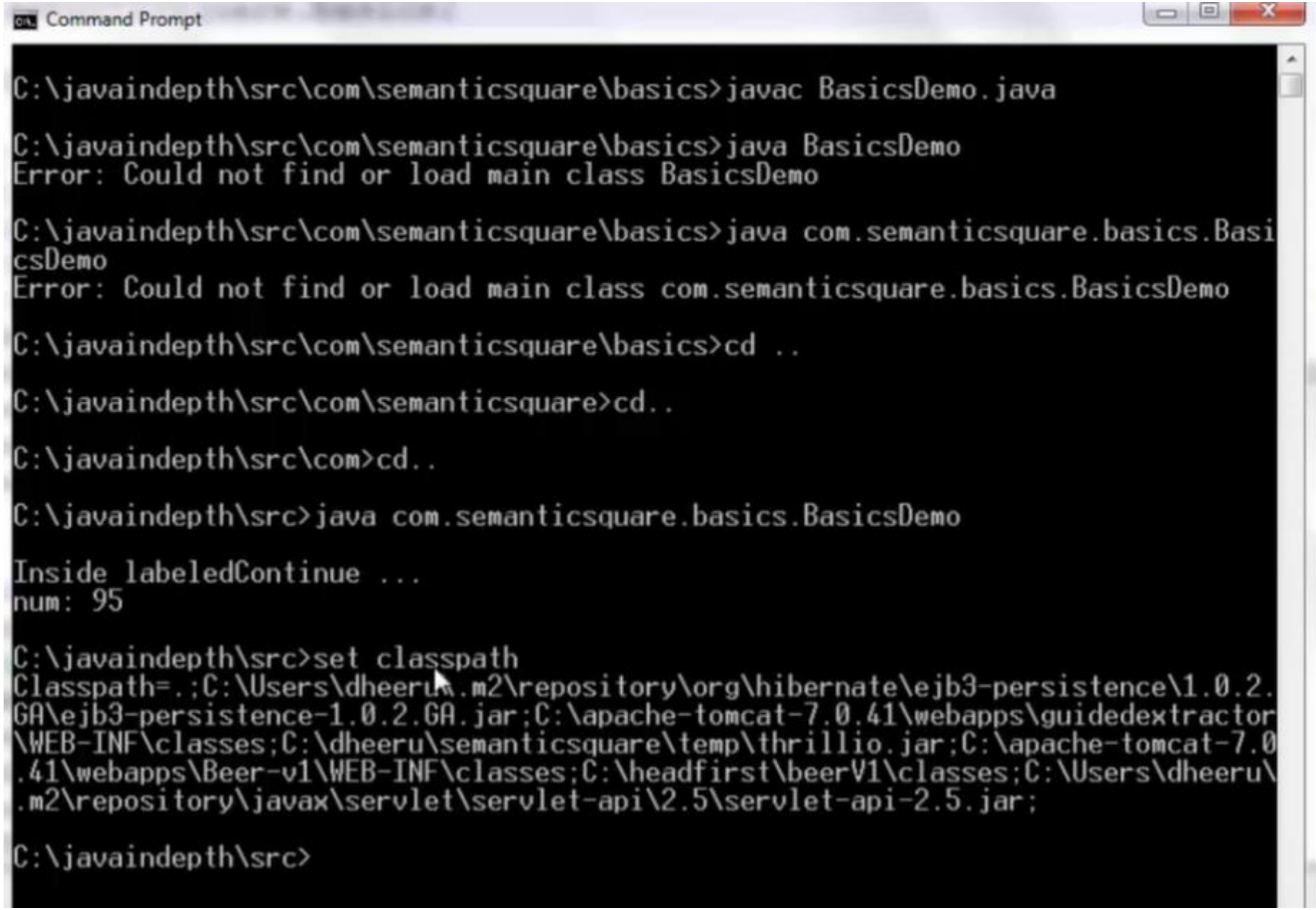
java *BasicsDemo* ~ *will not work*

java *com.semanticsquare.basics.BasicsDemo*


dentTest.java
Student.java
BasicsDemo.java
new 5
new 3
new 4
package com.semanticsquare.basics;
class Basics
// Adapted
http://www
static voi
System.
System.
System.
System.
System.
System.
System.
}
static voi
System.
// liter
int intH
System.
int intE
System.
int intC
Command Prompt
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Error: Could not find or load main class BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>

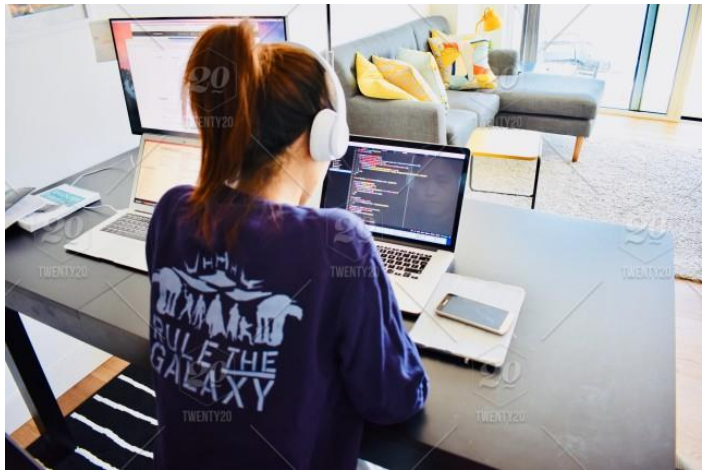

Command Prompt
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Error: Could not find or load main class BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>java com.semanticsquare.basics.Basi
csDemo
Error: Could not find or load main class com.semanticsquare.basics.BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>


Command Prompt
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Error: Could not find or load main class BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>java com.semanticsquare.basics.Basi
csDemo
Error: Could not find or load main class com.semanticsquare.basics.BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>cd ..
C:\javaindepth\src\com\semanticsquare>cd..
C:\javaindepth\src\com>cd..
C:\javaindepth\src>java com.semanticsquare.basics.BasicsDemo
Inside labeledContinue ...
num: 95
C:\javaindepth\src>


Command Prompt
C:\javaindepth\src\com\semanticsquare\basics>javac BasicsDemo.java
C:\javaindepth\src\com\semanticsquare\basics>java BasicsDemo
Error: Could not find or load main class BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>java com.semanticsquare.basics.Basi
csDemo
Error: Could not find or load main class com.semanticsquare.basics.BasicsDemo
C:\javaindepth\src\com\semanticsquare\basics>cd ..
C:\javaindepth\src\com\semanticsquare>cd..
C:\javaindepth\src\com>cd..
C:\javaindepth\src>java com.semanticsquare.basics.BasicsDemo
Inside labeledContinue ...
num: 95
C:\javaindepth\src>set classpath
Classpath=.;C:\Users\dheeru\.m2\repository\org\hibernate\ejb3-persistence\1.0.2.
GA\ejb3-persistence-1.0.2.GA.jar;C:\apache-tomcat-7.0.41\webapps\guidedextractor
\WEB-INF\classes;C:\dheeru\semanticsquare\temp\thrillio.jar;C:\apache-tomcat-7.0
.41\webapps\Beer-v1\WEB-INF\classes;C:\headfirst\beerV1\classes;C:\Users\dheeru\
.m2\repository\javax\servlet\servlet-api\2.5\servlet-api-2.5.jar;
C:\javaindepth\src>

Наименоване на packages

stanford.edu



math.geometry

oracle.org

math.geometry

package name conflict

Avoiding Package Name Conflicts

Use organization's *reverse internet domain name*

edu.stanford.math.geometry
com.oracle.math.geometry

Components Naming Conventions

- **Lowercase alphabets**, *rarely digits*
- Short ~ generally, *less than* ***8*** characters
- Meaningful abbreviations, e.g., util for utilities
- Acronyms are fine, e.g., awt for Abstract Window Toolkit
- Generally, single word
- Never start with *java* or *javax*

Strings

```
char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };
String helloString = new String(helloArray);
System.out.println(helloString);
```

Strings

Object of class ***java.lang.String***

```
String s = new String(); // empty string
String s = new String("hello!");

char[] cArray = {'h', 'e', 'l', 'l', 'o', '!'};
String s = new String(cArray);
```

(not recommended)

```
String s = "hello!"; // string literal
```

(recommended)

Strings

- String class uses **character array** to store text
- Java uses **UTF-16** for characters
- String is *sequence of unicode characters*
- String is *immutable*

String object ~ *immutable sequence of unicode characters*

String is Special

- String literal
- **+** operator

```
String s = “hello” + “ world!”; // “hello world!”
```

- String pool ~ *saves memory*

Операции със стрингове

- Comparing
- Concatenating Strings
- Formatting Strings
- Converting Strings to Numbers
- Manipulating characters in a String
- Searching in a String
- Replacing Characters and Substrings into a String
- Case translation
- Split

- Formatting strings

```
System.out.printf("The value of the float " +
                  "variable is %f, while " +
                  "the value of the " +
                  "integer variable is %d, " +
                  "and the string is %s",
                  floatVar, intVar, stringVar);
```

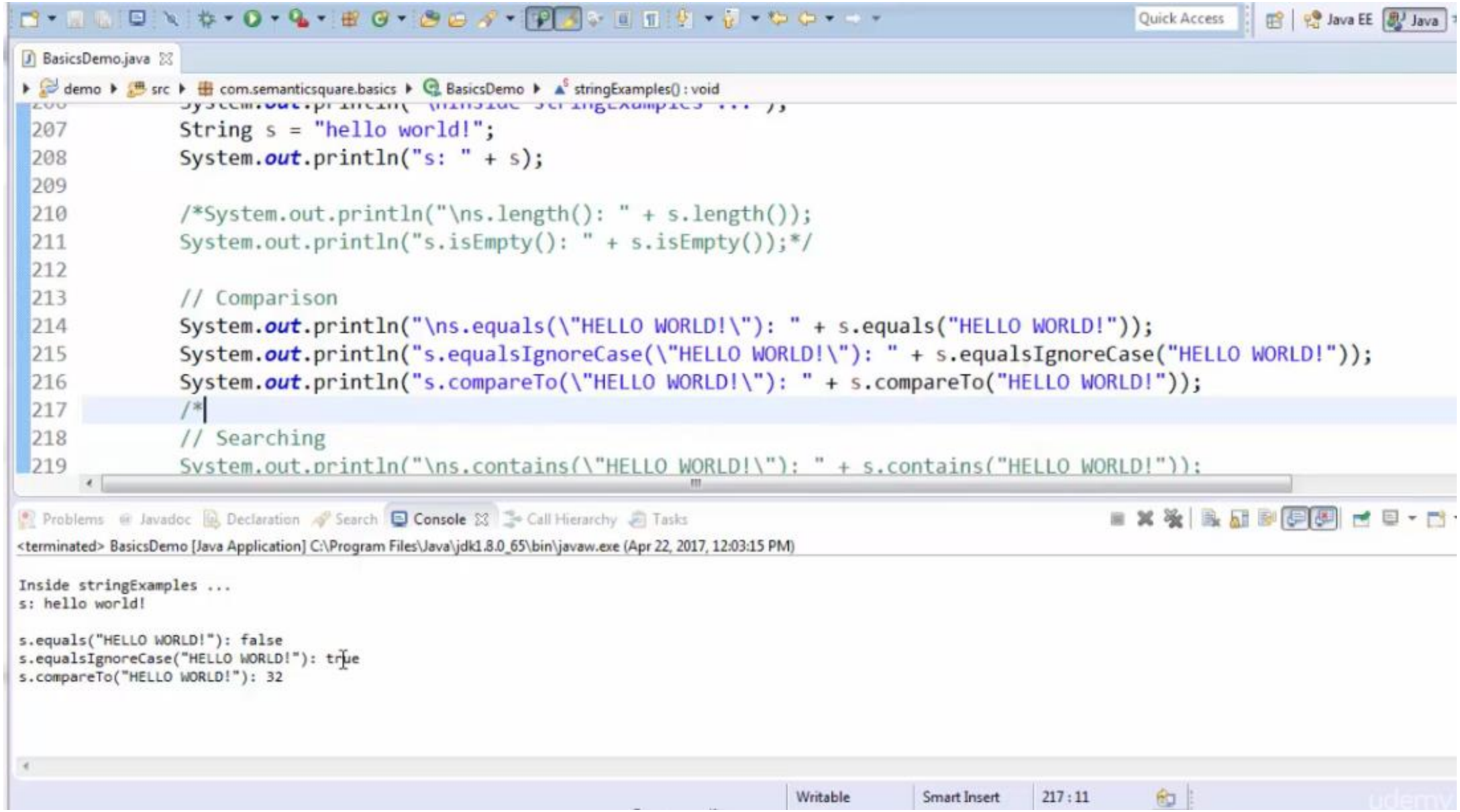
```
String fs;
fs = String.format("The value of the float " +
                   "variable is %f, while " +
                   "the value of the " +
                   "integer variable is %d, " +
                   " and the string is %s",
                   floatVar, intVar, stringVar);
System.out.println(fs);
```

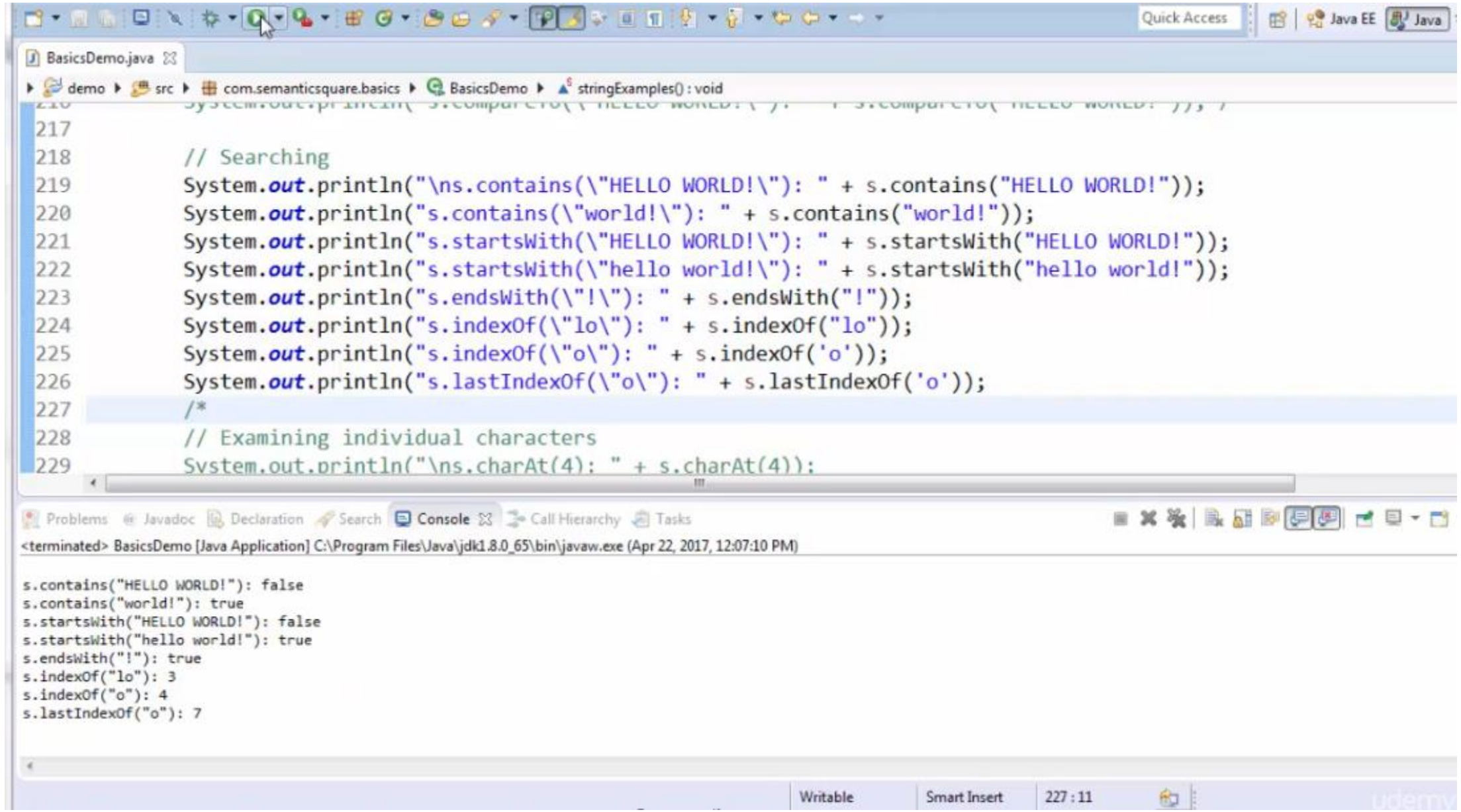
Other Methods in the `String` Class for Manipulating Strings

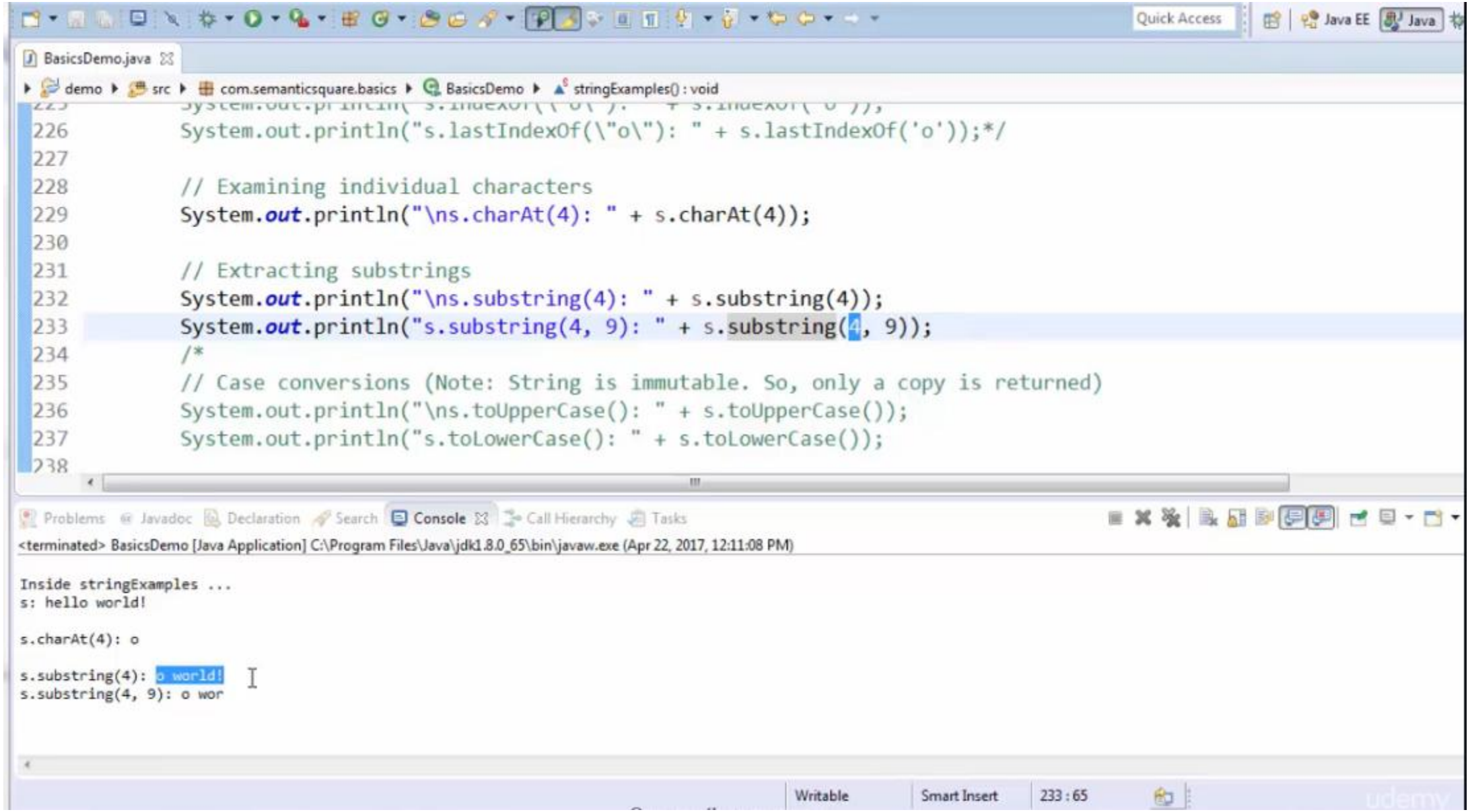
Method	Description
`String[] split(String regex)` `String[] split(String regex, int limit)`	Searches for a match as specified by the string argument (which contains a regular expression) and splits this string into an array of strings accordingly. The optional integer argument specifies the maximum size of the returned array. Regular expressions are covered in the lesson titled "Regular Expressions."
`CharSequence subSequence(int beginIndex, int endIndex)`	Returns a new character sequence constructed from `beginIndex` index up until `endIndex` - 1.
`String trim()`	Returns a copy of this string with leading and trailing white space removed.
`String toLowerCase()` `String toUpperCase()`	Returns a copy of this string converted to lowercase or uppercase. If no conversions are necessary, these methods return the original string.

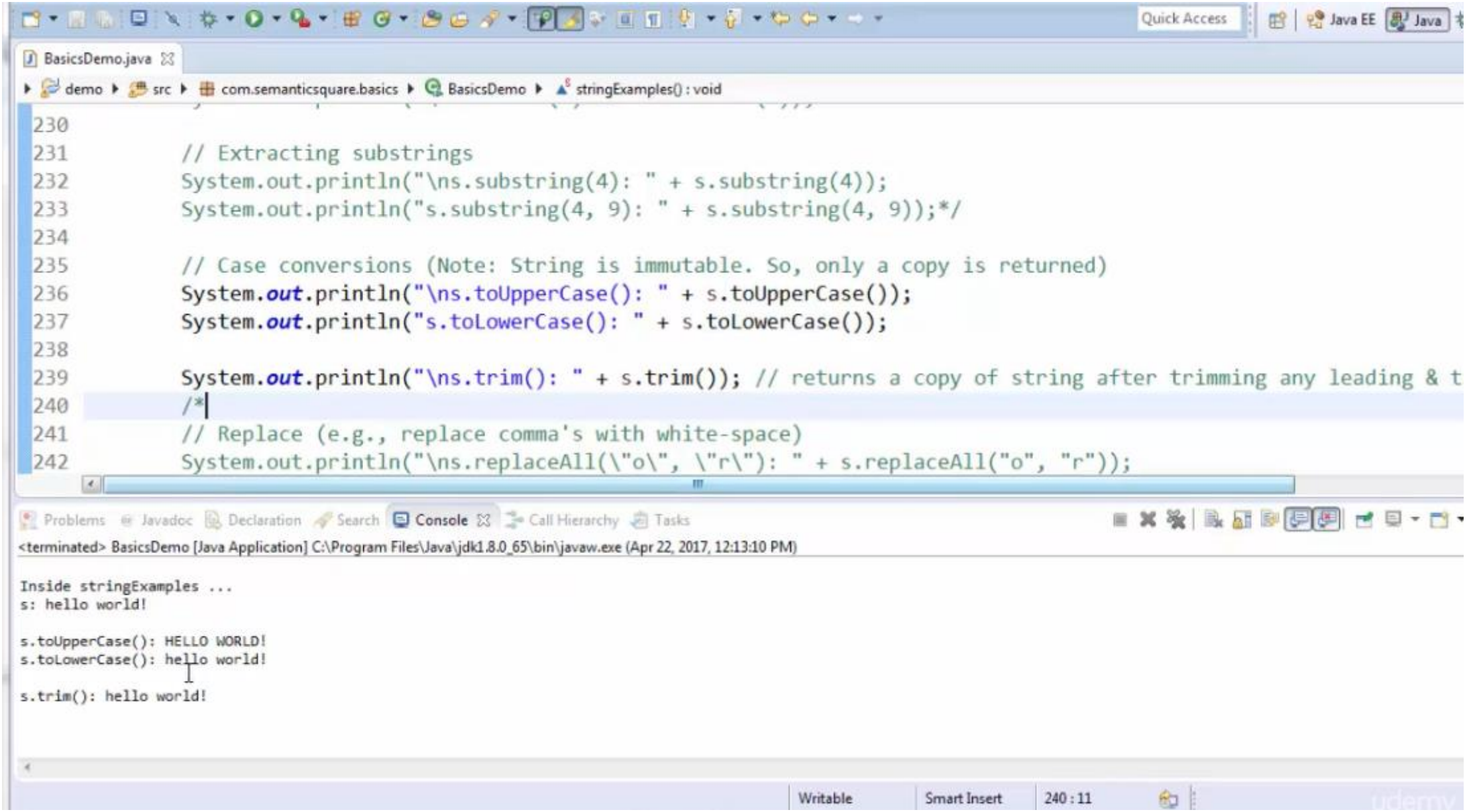
The Search Methods in the `String` Class

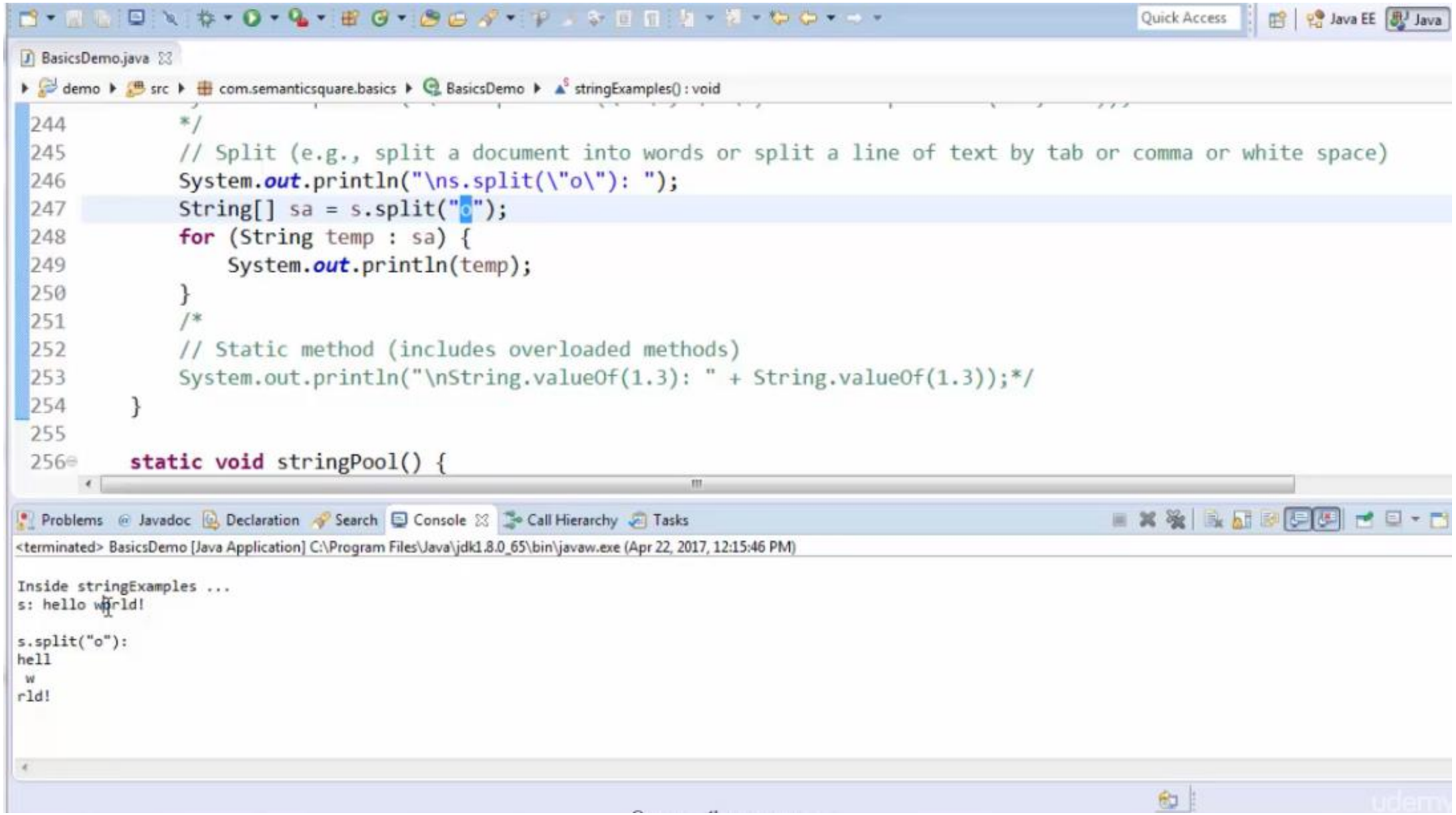
Method	Description
`int indexOf(int ch)` `int lastIndexOf(int ch)`	Returns the index of the first (last) occurrence of the specified character.
`int indexOf(int ch, int fromIndex)` `int lastIndexOf(int ch, int fromIndex)`	Returns the index of the first (last) occurrence of the specified character, searching forward (backward) from the specified index.
`int indexOf(String str)` `int lastIndexOf(String str)`	Returns the index of the first (last) occurrence of the specified substring.
`int indexOf(String str, int fromIndex)` `int lastIndexOf(String str, int fromIndex)`	Returns the index of the first (last) occurrence of the specified substring, searching forward (backward) from the specified index.
`boolean contains(CharSequence s)`	Returns true if the string contains the specified character sequence.

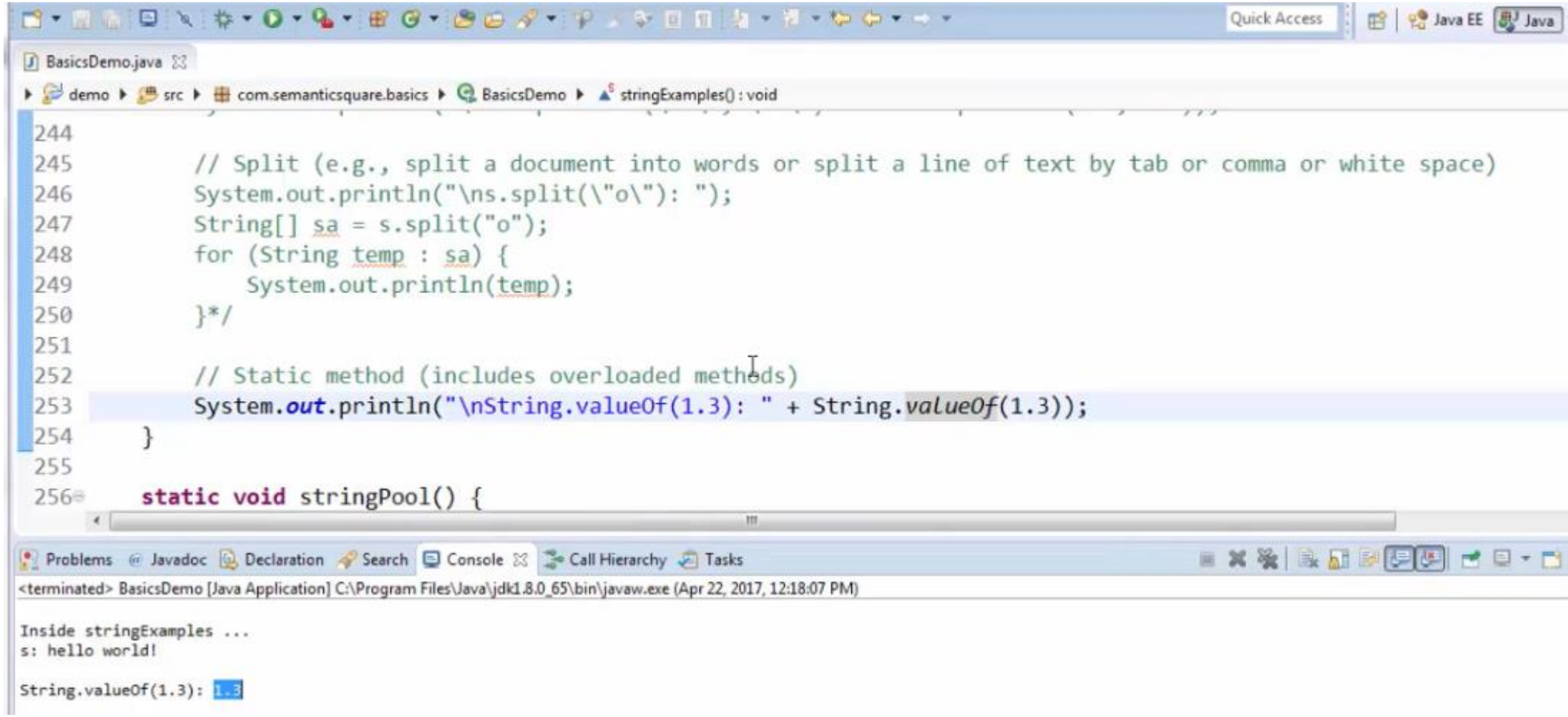

BasicsDemo.java
demo ▸ src ▸ com.semanticsquare.basics ▸ BasicsDemo ▸ stringExamples() : void
207 String s = "hello world!";
208 System.out.println("s: " + s);
209
210 /*System.out.println("\ns.length(): " + s.length());
211 System.out.println("s.isEmpty(): " + s.isEmpty());*/
212
213 // Comparison
214 System.out.println("\ns.equals(\"HELLO WORLD!\"): " + s.equals("HELLO WORLD!"));
215 System.out.println("s.equalsIgnoreCase(\"HELLO WORLD!\"): " + s.equalsIgnoreCase("HELLO WORLD!"));
216 System.out.println("s.compareTo(\"HELLO WORLD!\"): " + s.compareTo("HELLO WORLD!"));
217 /*
218 // Searching
219 System.out.println("\ns.contains(\"HELLO WORLD!\"): " + s.contains("HELLO WORLD!"));
Problems Javadoc Declaration Search Console Call Hierarchy Tasks
<terminated> BasicsDemo [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Apr 22, 2017, 12:03:15 PM)
Inside stringExamples ...
s: hello world!
s.equals("HELLO WORLD!"): false
s.equalsIgnoreCase("HELLO WORLD!"): true
s.compareTo("HELLO WORLD!"): 32
Writable
Smart Insert
217 : 11


BasicsDemo.java
demo ▸ src ▸ com.semanticsquare.basics ▸ BasicsDemo ▸ stringExamples() : void
217
218 // Searching
219 System.out.println("\ns.contains(\"HELLO WORLD!\"): " + s.contains("HELLO WORLD!"));
220 System.out.println("s.contains(\"world!\"): " + s.contains("world!"));
221 System.out.println("s.startsWith(\"HELLO WORLD!\"): " + s.startsWith("HELLO WORLD!"));
222 System.out.println("s.startsWith(\"hello world!\"): " + s.startsWith("hello world!"));
223 System.out.println("s.endsWith(\"!\"): " + s.endsWith("!"));
224 System.out.println("s.indexOf(\"lo\"): " + s.indexOf("lo"));
225 System.out.println("s.indexOf(\"o\"): " + s.indexOf('o'));
226 System.out.println("s.lastIndexOf(\"o\"): " + s.lastIndexOf('o'));
227 /*
228 // Examining individual characters
229 System.out.println("\ns.charAt(4): " + s.charAt(4));
Problems Javadoc Declaration Search Console Call Hierarchy Tasks
<terminated> BasicsDemo [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Apr 22, 2017, 12:07:10 PM)
s.contains("HELLO WORLD!"): false
s.contains("world!"): true
s.startsWith("HELLO WORLD!"): false
s.startsWith("hello world!"): true
s.endsWith("!"): true
s.indexOf("lo"): 3
s.indexOf("o"): 4
s.lastIndexOf("o"): 7
Writable
Smart Insert
227 : 11


BasicsDemo.java
demo
src
com.semanticsquare.basics
BasicsDemo
stringExamples() : void
System.out.println("s.lastIndexOf(\"o\"): " + s.lastIndexOf('o'));*/
// Examining individual characters
System.out.println("\ns.charAt(4): " + s.charAt(4));
// Extracting substrings
System.out.println("\ns.substring(4): " + s.substring(4));
System.out.println("s.substring(4, 9): " + s.substring(4, 9));
/*
// Case conversions (Note: String is immutable. So, only a copy is returned)
System.out.println("\ns.toUpperCase(): " + s.toUpperCase());
System.out.println("s.toLowerCase(): " + s.toLowerCase());
Problems
Javadoc
Declaration
Search
Console
Call Hierarchy
Tasks
<terminated> BasicsDemo [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Apr 22, 2017, 12:11:08 PM)
Inside stringExamples ...
s: hello world!
s.charAt(4): o
s.substring(4): o world!
s.substring(4, 9): o wor
Writable
Smart Insert
233 : 65


BasicsDemo.java
demo ▸ src ▸ com.semanticsquare.basics ▸ BasicsDemo ▸ stringExamples() : void
230
231 // Extracting substrings
232 System.out.println("\ns.substring(4): " + s.substring(4));
233 System.out.println("s.substring(4, 9): " + s.substring(4, 9));*/
234
235 // Case conversions (Note: String is immutable. So, only a copy is returned)
236 System.out.println("\ns.toUpperCase(): " + s.toUpperCase());
237 System.out.println("s.toLowerCase(): " + s.toLowerCase());
238
239 System.out.println("\ns.trim(): " + s.trim()); // returns a copy of string after trimming any leading & t
240 /*
241 // Replace (e.g., replace comma's with white-space)
242 System.out.println("\ns.replaceAll(\"o\", \"r\"): " + s.replaceAll("o", "r"));
Problems Javadoc Declaration Search Console Call Hierarchy Tasks
<terminated> BasicsDemo [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Apr 22, 2017, 12:13:10 PM)
Inside stringExamples ...
s: hello world!
s.toUpperCase(): HELLO WORLD!
s.toLowerCase(): hello world!
s.trim(): hello world!
Writable
Smart Insert
240 : 11

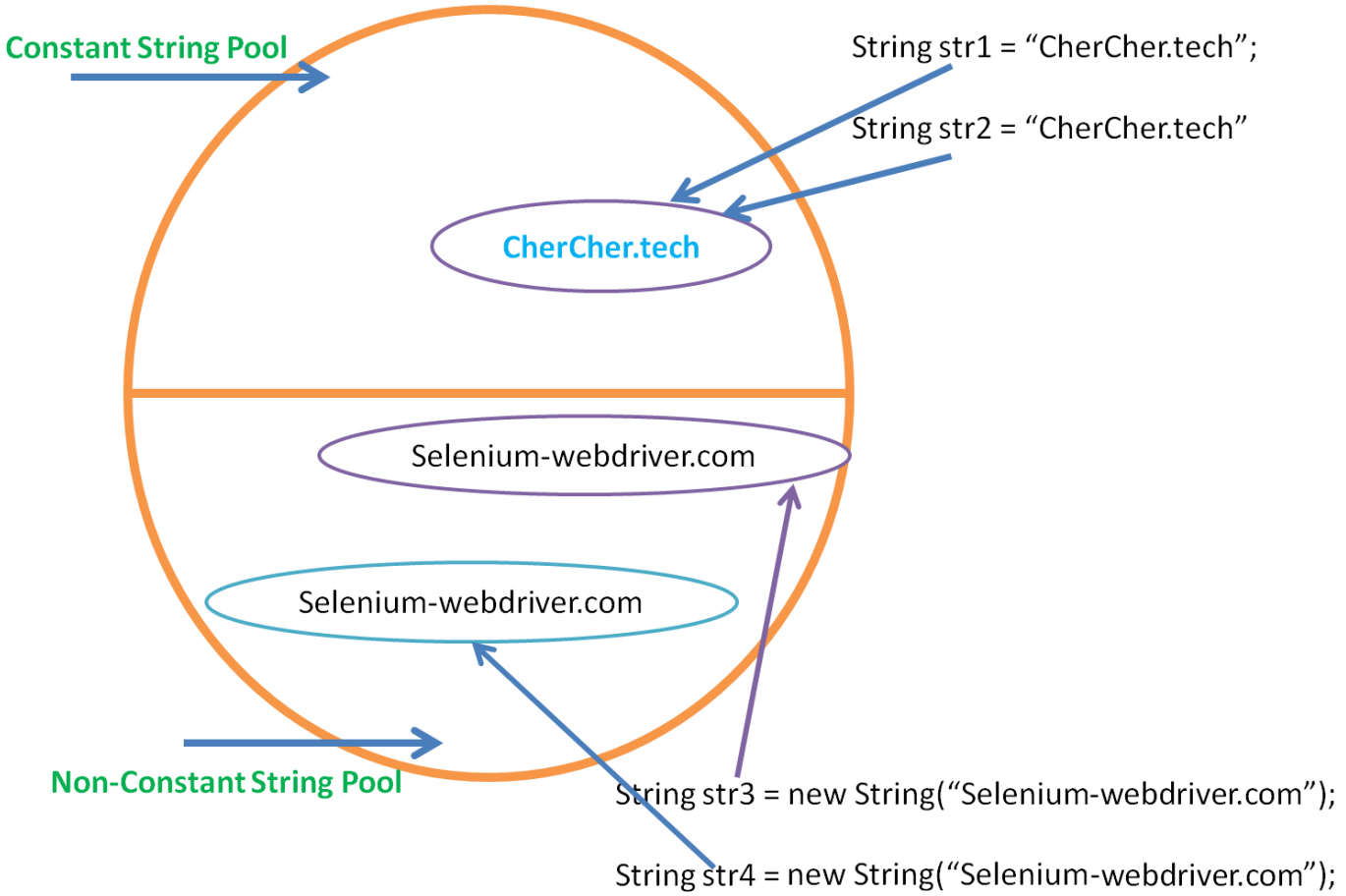

Quick Access
Java EE
Java
BasicsDemo.java
demo
src
com.semanticsquare.basics
BasicsDemo
stringExamples() : void
244 */
245 // Split (e.g., split a document into words or split a line of text by tab or comma or white space)
246 System.out.println("\ns.split(\"o\"): ");
247 String[] sa = s.split("o");
248 for (String temp : sa) {
249 System.out.println(temp);
250 }
251 /*
252 // Static method (includes overloaded methods)
253 System.out.println("\nString.valueOf(1.3): " + String.valueOf(1.3));*/
254 }
255
256 static void stringPool() {
Problems
Javadoc
Declaration
Search
Console
Call Hierarchy
Tasks
<terminated> BasicsDemo [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Apr 22, 2017, 12:15:46 PM)
Inside stringExamples ...
s: hello world!
s.split("o"):
hell
w
rld!


Quick Access
Java EE
Java
BasicsDemo.java
demo
src
com.semanticsquare.basics
BasicsDemo
stringExamples() : void
244
245 // Split (e.g., split a document into words or split a line of text by tab or comma or white space)
246 System.out.println("\ns.split(\"o\"): ");
247 String[] sa = s.split("o");
248 for (String temp : sa) {
249 System.out.println(temp);
250 }*/
251
252 // Static method (includes overloaded methods)
253 System.out.println("\nString.valueOf(1.3): " + String.valueOf(1.3));
254 }
255
256 static void stringPool() {
Problems
Javadoc
Declaration
Search
Console
Call Hierarchy
Tasks
<terminated> BasicsDemo [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Apr 22, 2017, 12:18:07 PM)
Inside stringExamples ...
s: hello world!
String.valueOf(1.3): 1.3

3rd Party String Utilities

- Apache Commons Lang ~ **StringUtils**
- **Guava's** String Utility Classes

String Pool


Constant String Pool
String str1 = “CherCher.tech”;
String str2 = “CherCher.tech”
CherCher.tech
Selenium-webdriver.com
Selenium-webdriver.com
Non-Constant String Pool
String str3 = new String(“Selenium-webdriver.com”);
String str4 = new String(“Selenium-webdriver.com”);

String Pool

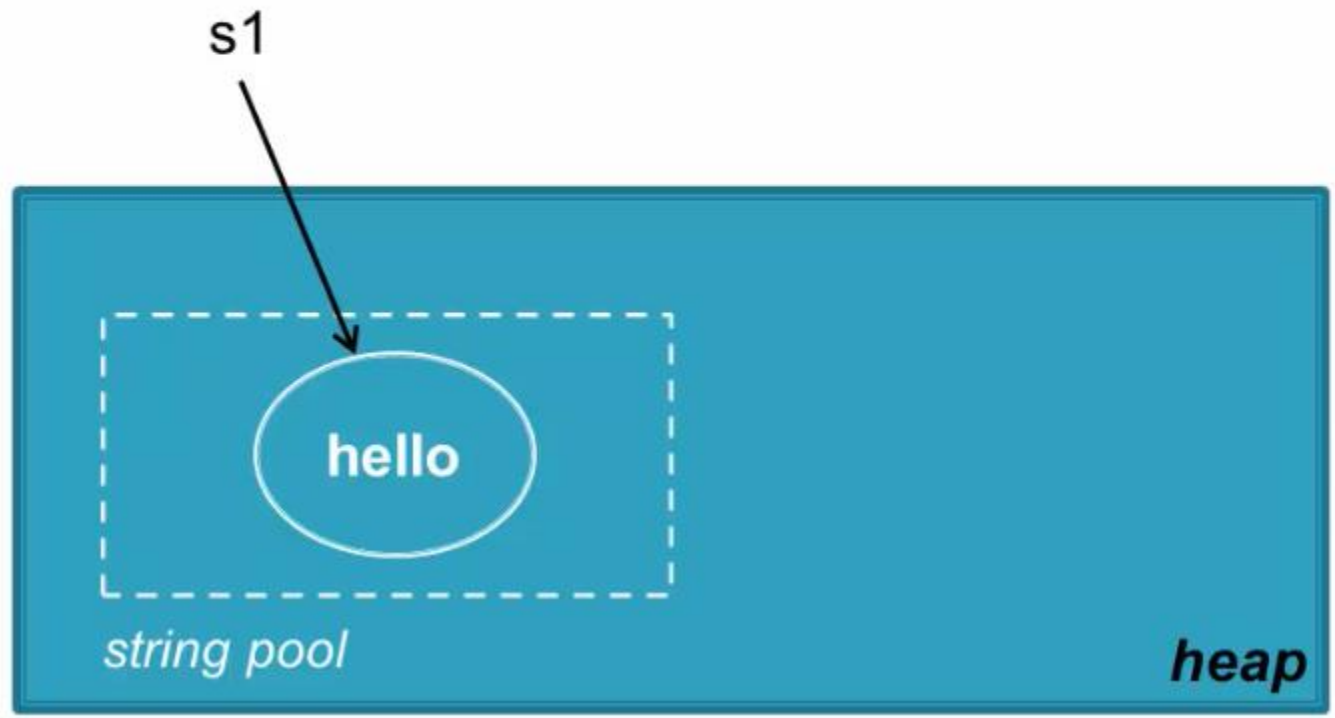
- Stores ***single*** copy of each string literal as **string object**
- Only ***one*** string pool

Използването на *стрингов литерал* или използването на *new*

- Чрез стрингов литерал
  - Записва се *string pool* в heap
  - Литералите с едно и също състояние споделят памета
- Чрез *new*
  - *Еднакви са като обект*
  - *Не споделят една памет*

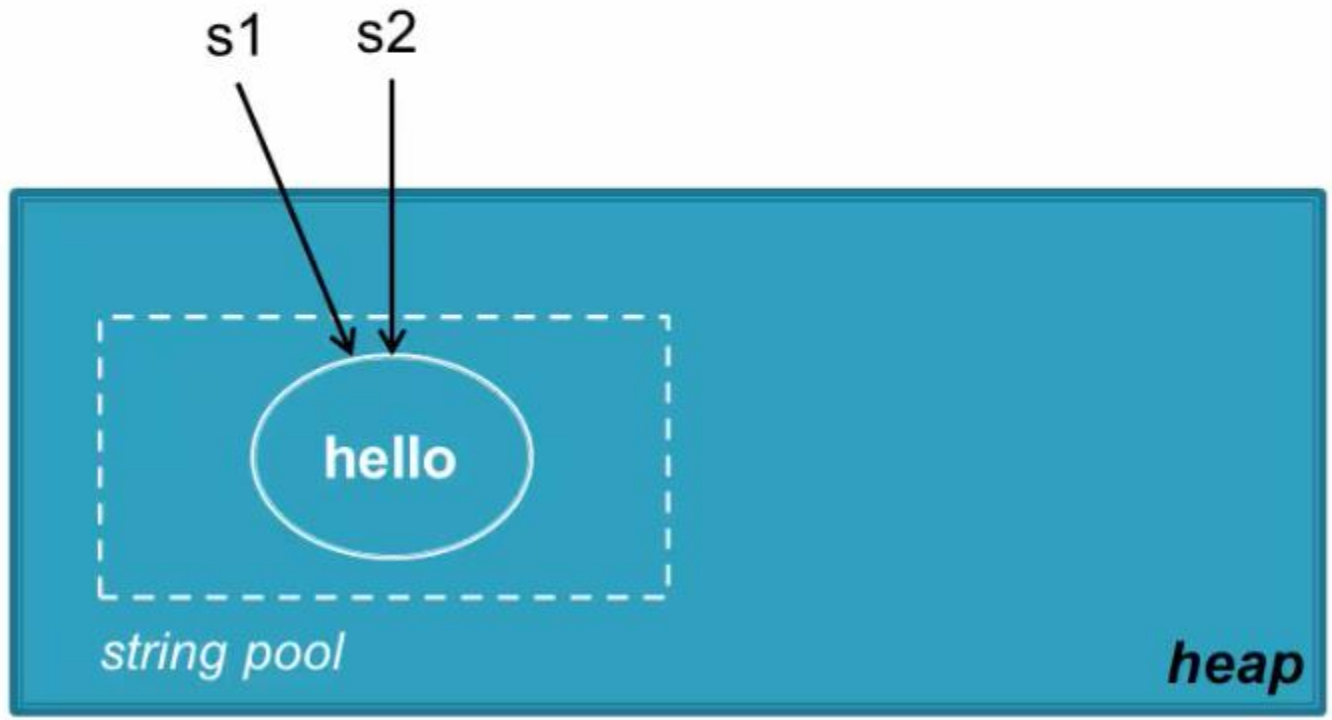
String *s1* = "hello";

String *s2* = "hello";



String *s1* = "hello";

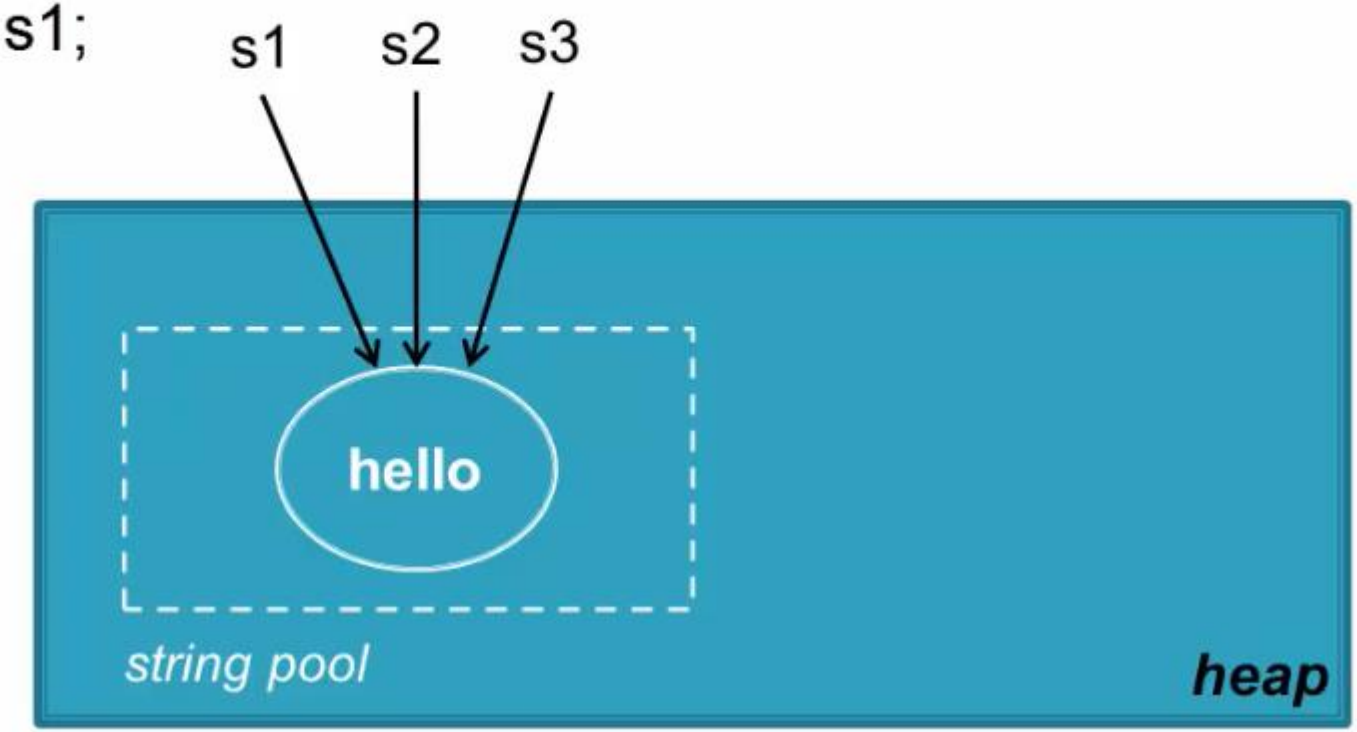
String *s2* = "hello";



String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

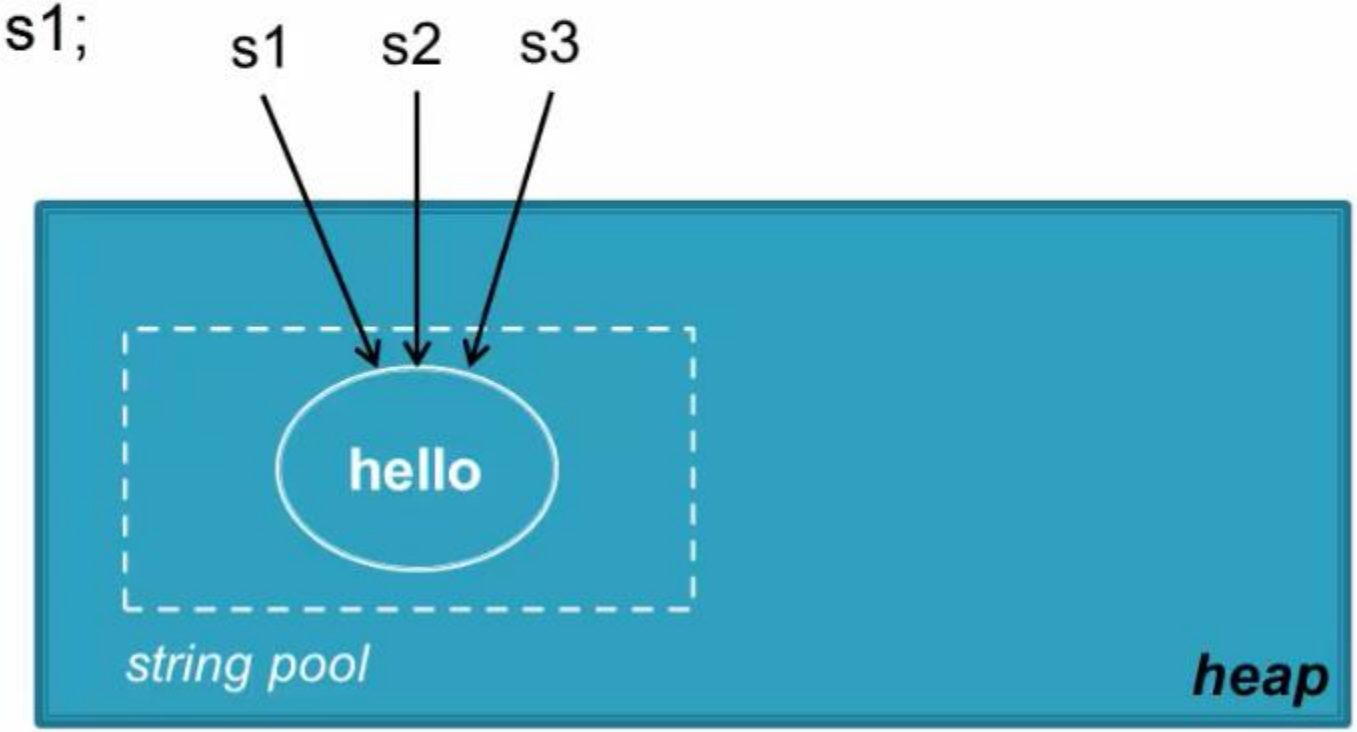


String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

String *s4* = **new** String("hello");

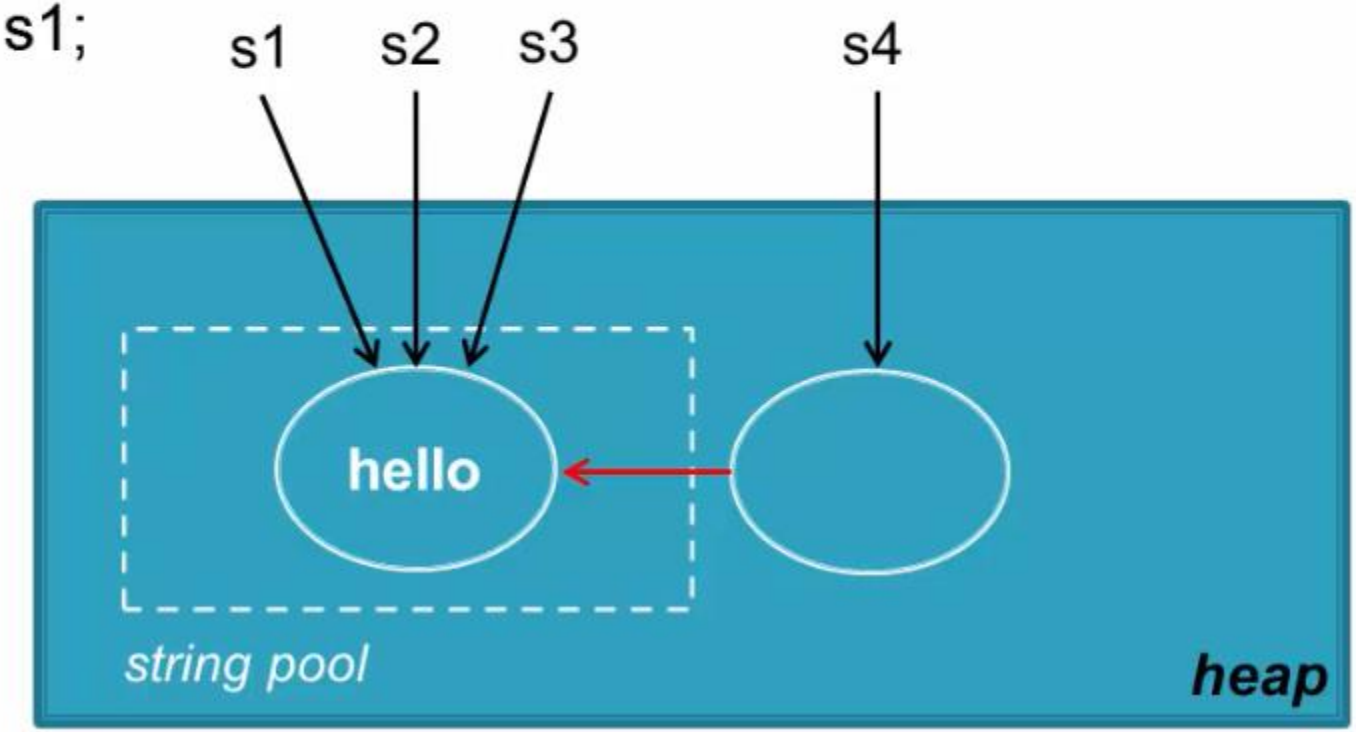


String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

String *s4* = **new** String("hello");



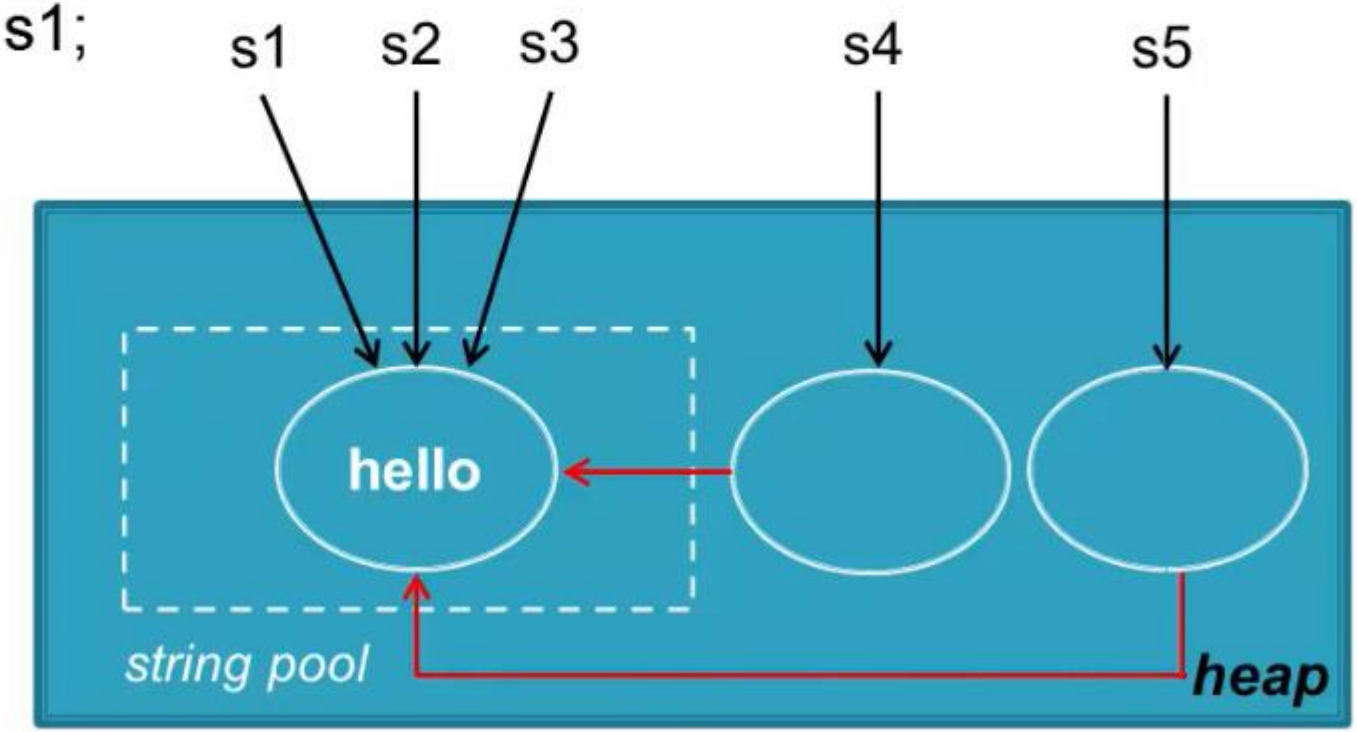
String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

String *s4* = **new** String("hello");

String *s5* = **new** String("hello");



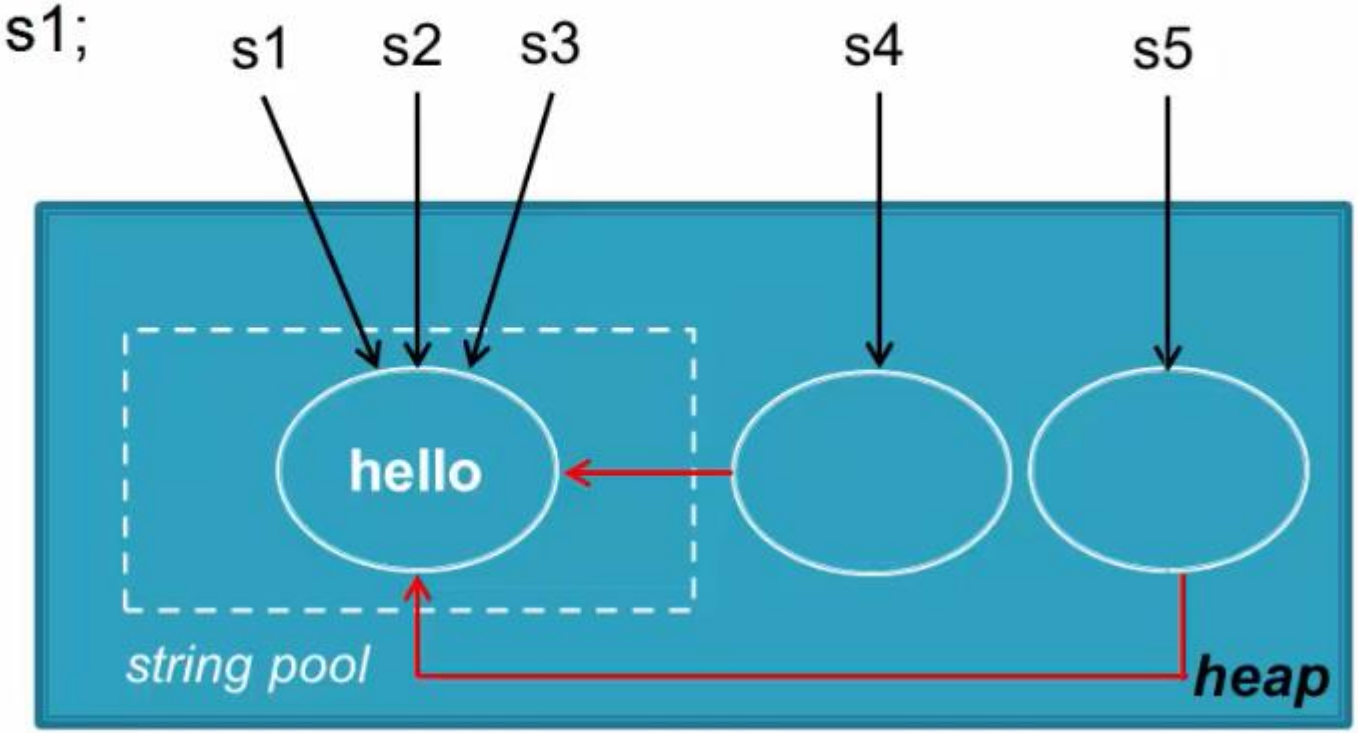
String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

String *s4* = **new** String("hello");

String *s5* = **new** String("hello");



s1 == s2? True

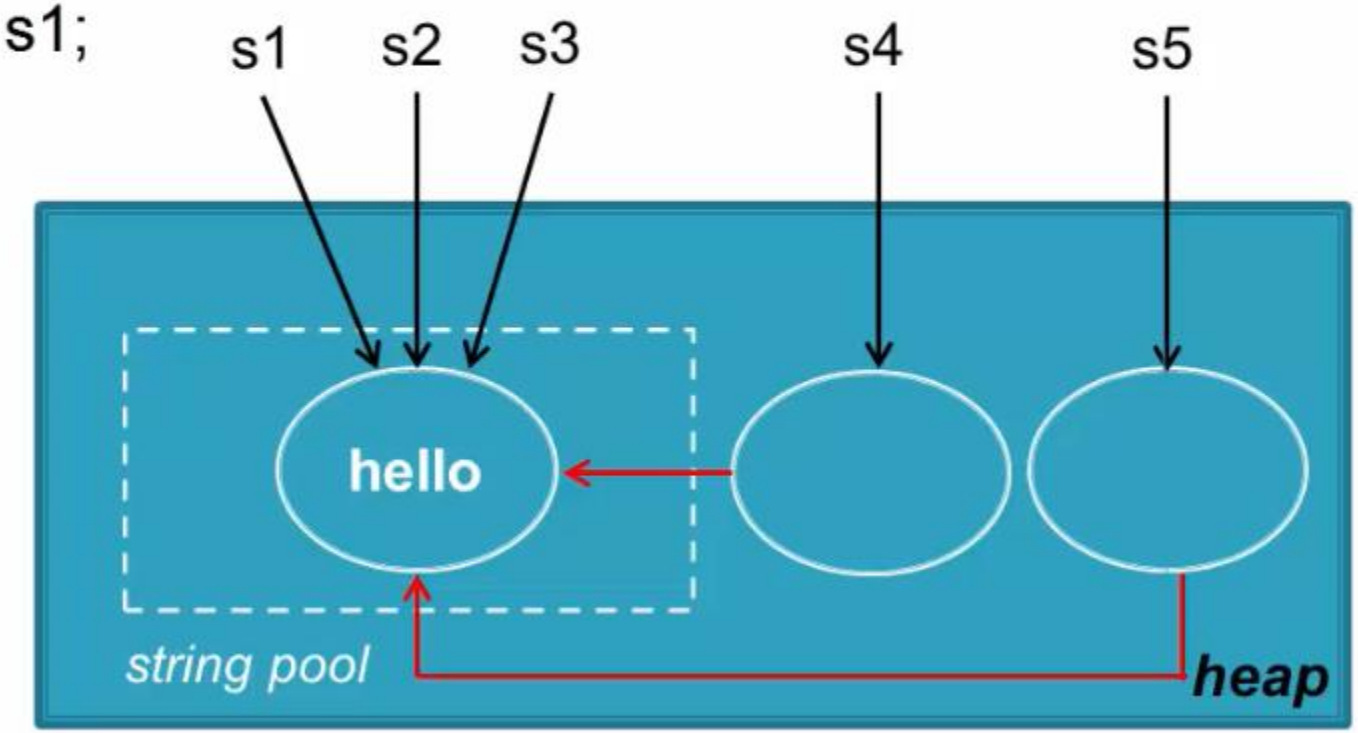
String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

String *s4* = **new** String("hello");

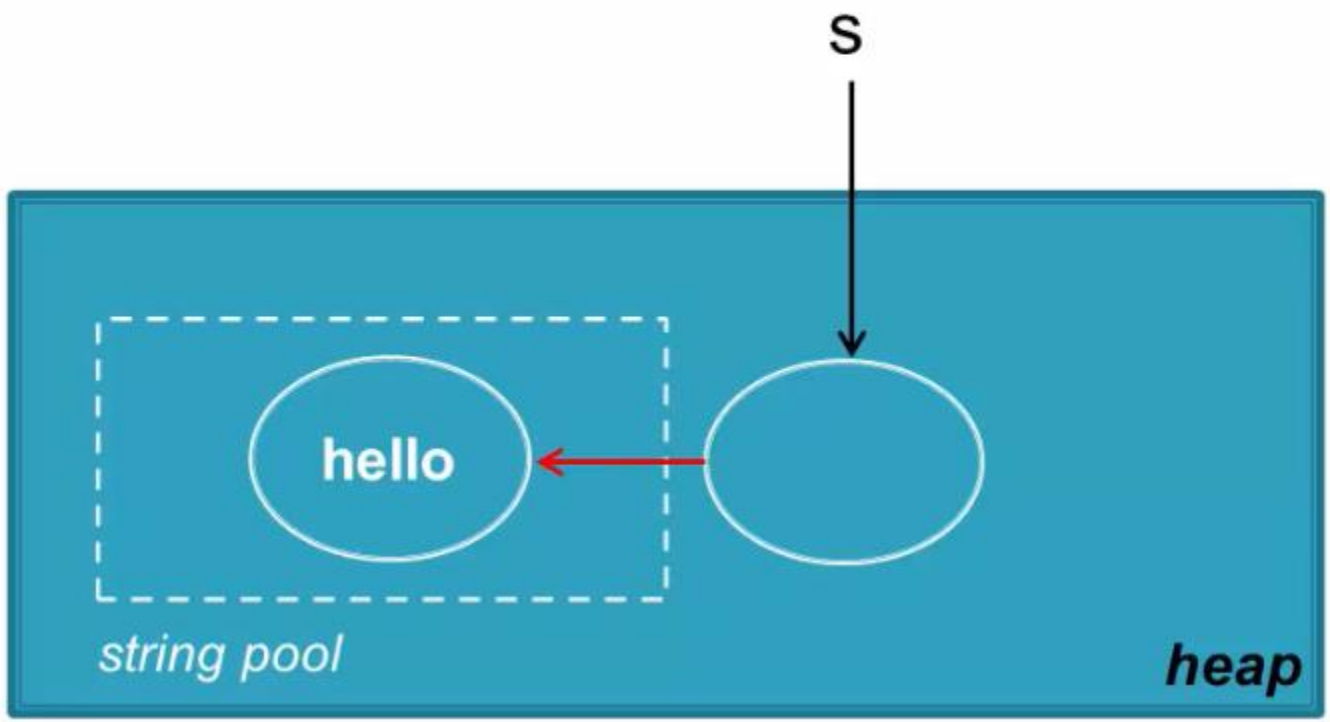
String *s5* = **new** String("hello");



s1 == s2? True

s4 == s5? False

String *s* = **new** String("hello");



String Interning

- *Process* of building string pool
- ***intern*** ~ a string pool element
- Supported by *Python*, *Ruby*, *C#*, *JavaScript*

String Interning by JVM

encountering a string literal for first time

✓ Create new **String** object with given literal

✓ Invoke ***intern***()

abandoned & garbage collected

if (string *in* **string pool**)
  return *existing* reference
else
  add to string pool and return reference

String Interning by JVM

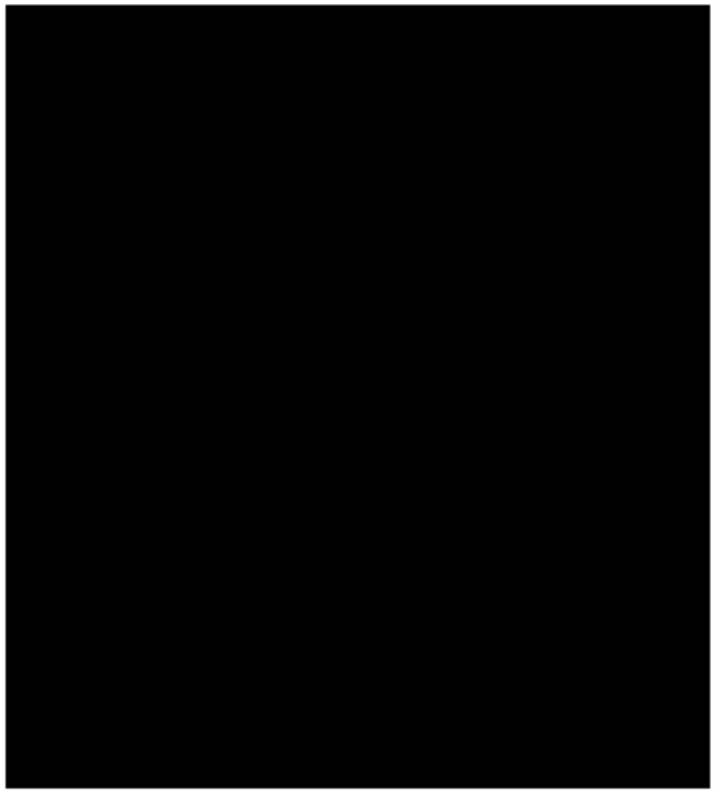
encountering a string literal for first time

✓ Create new **String** object with given literal

✓ Invoke ***intern***()

added to string pool

if (string *in* **string pool**)
  return *existing* reference
else
  add to string pool and return reference



```
String a = "Hoit";
String b = "Hoit";
String c = "Hoit";
String aa = new String("Hoit");
String bb = new String("Hoit");
bb = bb.intern();
```