EXPLORE DESIGN PERFECTION

Introduction to Java

Carlos Kavka

ESTECO SpA

esteco.com

About this course

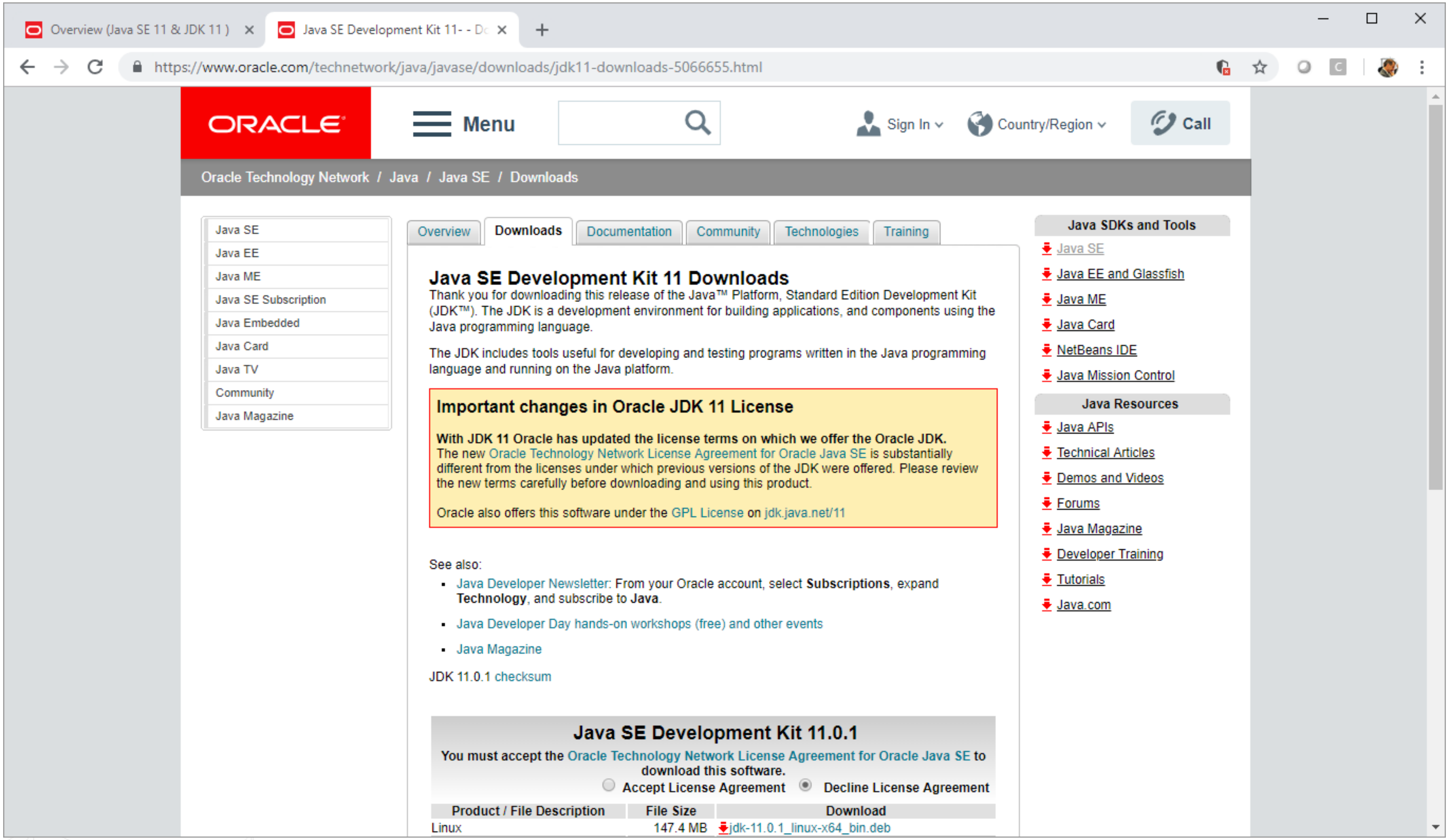
Introduction to the language

many examples

covers up to Java 9

Tools

Java SDK



Java SE Development Kit 11 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Important changes in Oracle JDK 11 License

With JDK 11 Oracle has updated the license terms on which we offer the Oracle JDK.
The new Oracle Technology Network License Agreement for Oracle Java SE is substantially different from the licenses under which previous versions of the JDK were offered. Please review the new terms carefully before downloading and using this product.

Oracle also offers this software under the GPL License on jdk.java.net/11

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 11.0.1 checksum

Java SE Development Kit 11.0.1

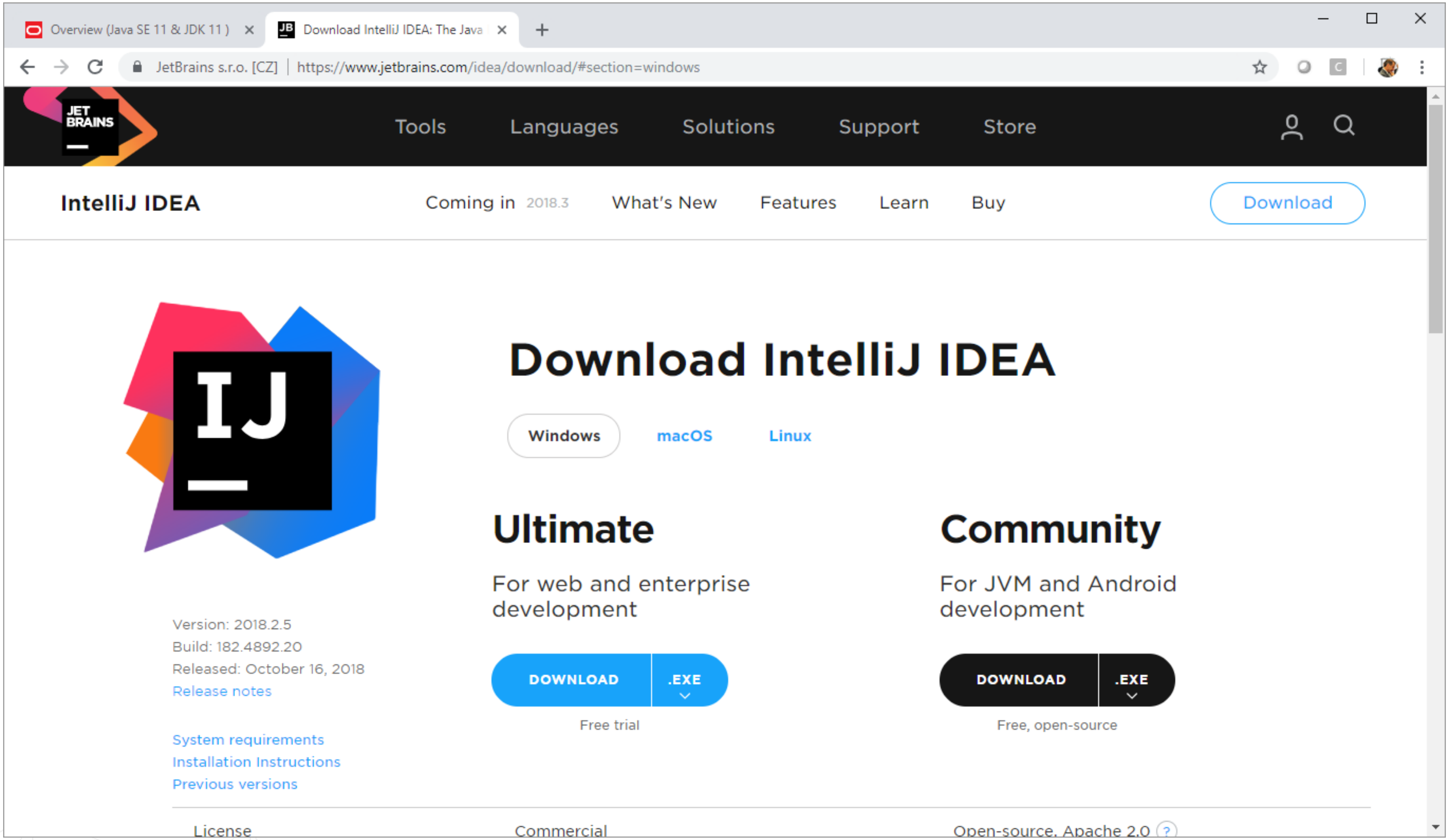
You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.

○ Accept License Agreement ● Decline License Agreement

Product / File Description	File Size	Download
Linux	147.4 MB	jdk-11.0.1_linux-x64_bin.deb

Tools

intelliJ IDEA Java IDE



Overview (Java SE 11 & JDK 11)

Download IntelliJ IDEA: The Java

JetBrains s.r.o. [CZ] | https://www.jetbrains.com/idea/download/#section=windows

Download IntelliJ IDEA

Ultimate

For web and enterprise development

Free trial

Community

For JVM and Android development

Free, open-source

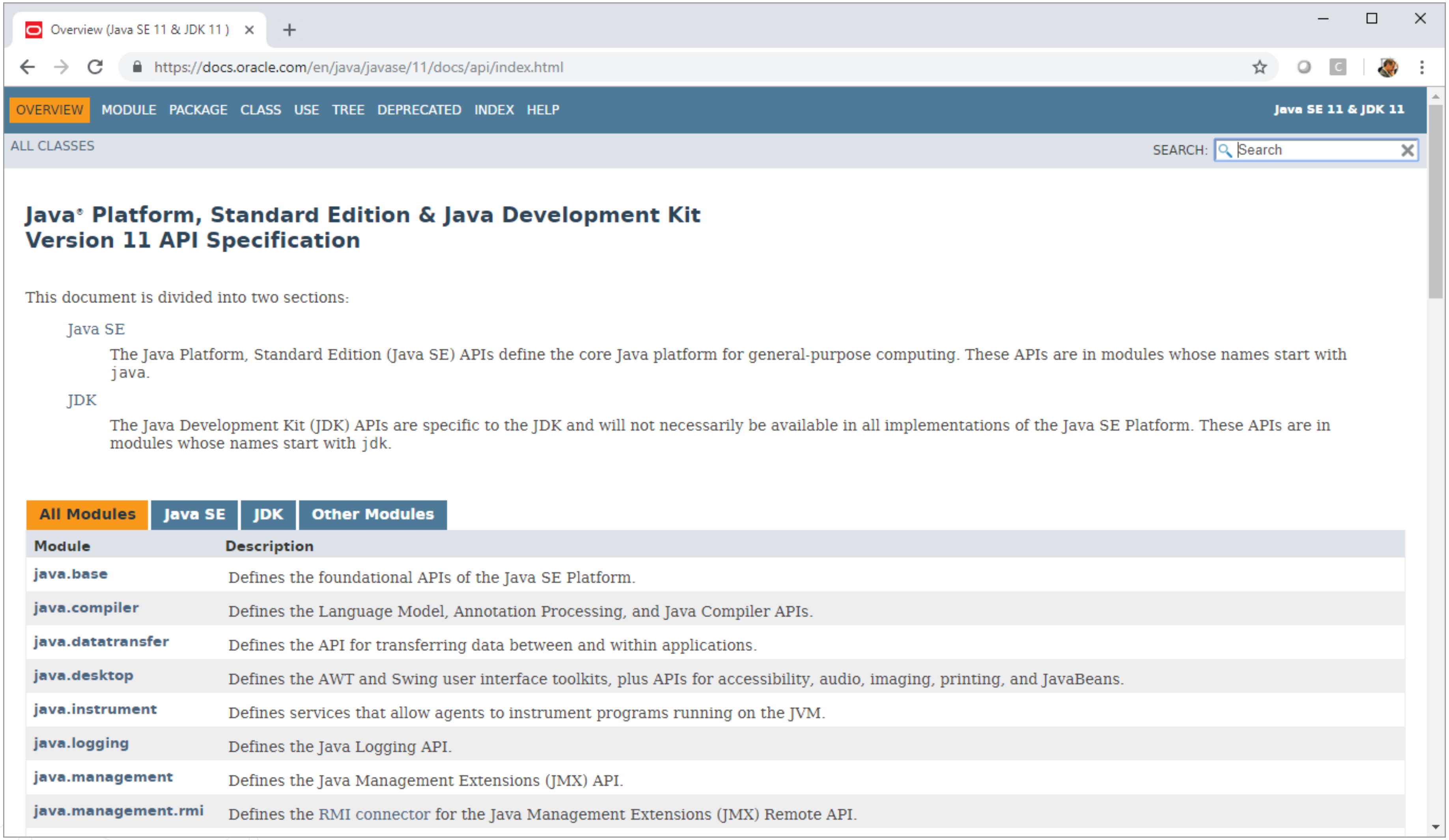
Version: 2018.2.5
Build: 182.4892.20
Released: October 16, 2018
Release notes

System requirements
Installation Instructions
Previous versions

License	Commercial	Open-source, Apache 2.0 ?

Tools

Java API



Overview (Java SE 11 & JDK 11)
https://docs.oracle.com/en/java/javase/11/docs/api/index.html
OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
Java SE 11 & JDK 11
ALL CLASSES
SEARCH: Search

Java® Platform, Standard Edition & Java Development Kit Version 11 API Specification

This document is divided into two sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with java.

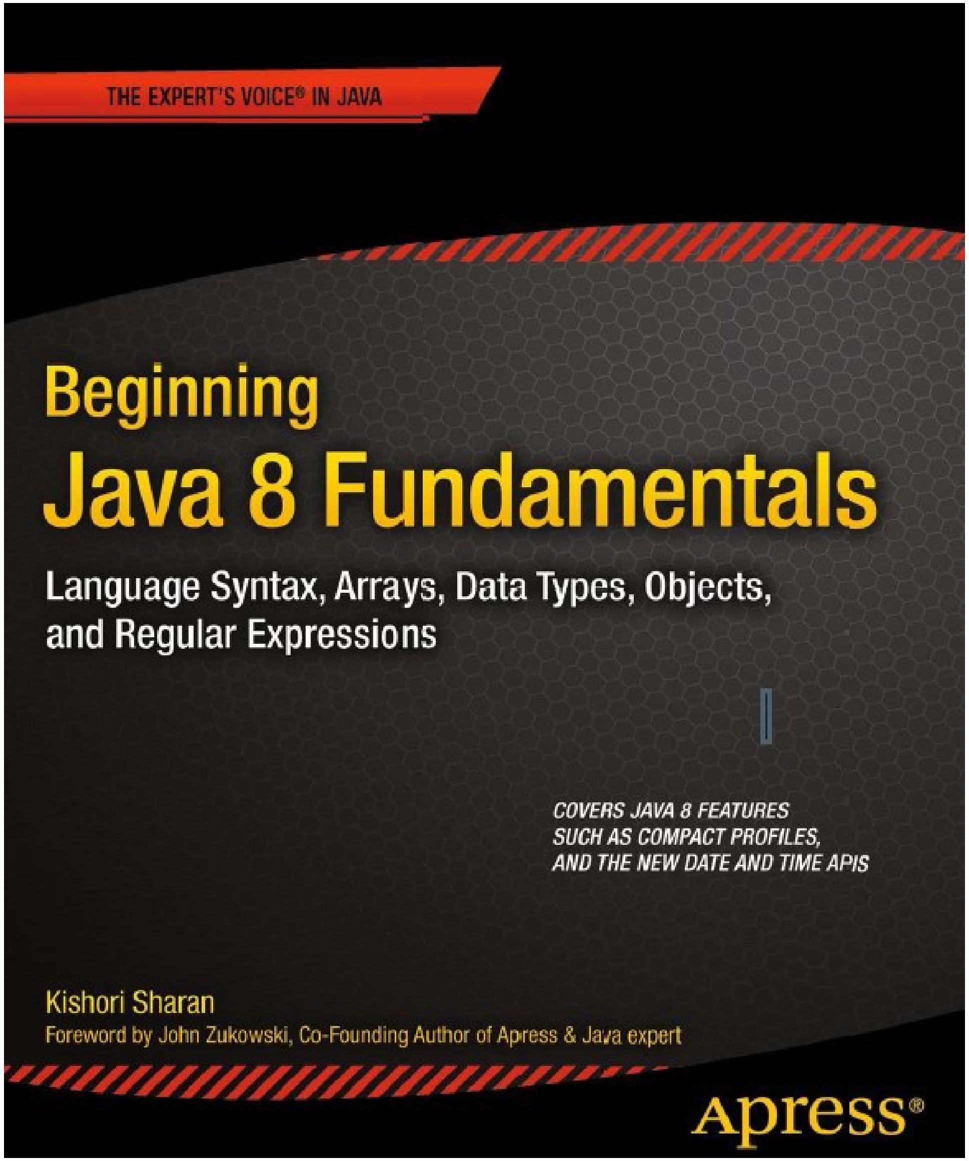
JDK

The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all implementations of the Java SE Platform. These APIs are in modules whose names start with jdk.

All Modules | Java SE | JDK | Other Modules

Module	Description
java.base	Defines the foundational APIs of the Java SE Platform.
java.compiler	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.datatransfer	Defines the API for transferring data between and within applications.
java.desktop	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.
java.instrument	Defines services that allow agents to instrument programs running on the JVM.
java.logging	Defines the Java Logging API.
java.management	Defines the Java Management Extensions (JMX) API.
java.management.rmi	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.

Books


THE EXPERT'S VOICE® IN JAVA

Beginning
Java 8 Fundamentals

Language Syntax, Arrays, Data Types, Objects, and Regular Expressions

COVERS JAVA 8 FEATURES SUCH AS COMPACT PROFILES, AND THE NEW DATE AND TIME APIS

Kishori Sharan

Foreword by John Zukowski, Co-Founding Author of Apress & Java expert

Apress®

many good books available!

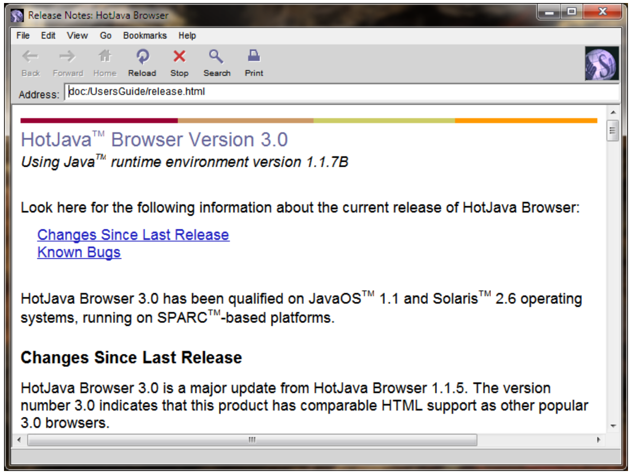
Introduction to Java

Part I - basic concepts

A bit of history

*7

HotJava browser


Release Notes: HotJava Browser
File Edit View Go Bookmarks Help
Back Forward Home Reload Stop Search Print

Address: doc:/UsersGuide/release.html

HotJava™ Browser Version 3.0

Using Java™ runtime environment version 1.1.7B

Look here for the following information about the current release of HotJava Browser:

- Changes Since Last Release
- Known Bugs

HotJava Browser 3.0 has been qualified on JavaOS™ 1.1 and Solaris™ 2.6 operating systems, running on SPARC™-based platforms.

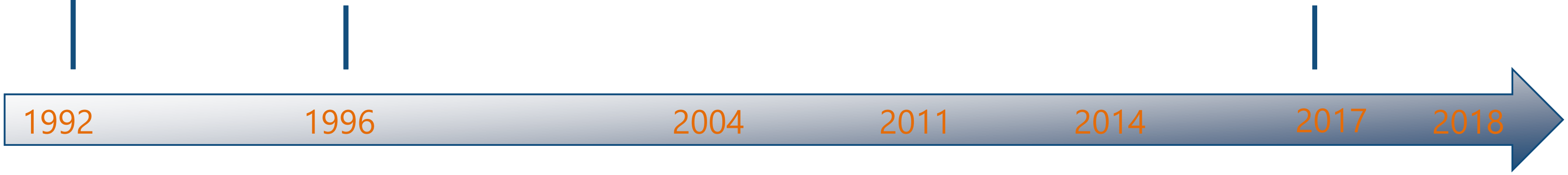
Changes Since Last Release

HotJava Browser 3.0 is a major update from HotJava Browser 1.1.5. The version number 3.0 indicates that this product has comparable HTML support as other popular 3.0 browsers.

Status: Registering Java Runtime Environment

3 Billion Devices Run Java

Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...



1992 Oak

1996 Java 1.0

2004 Java 5

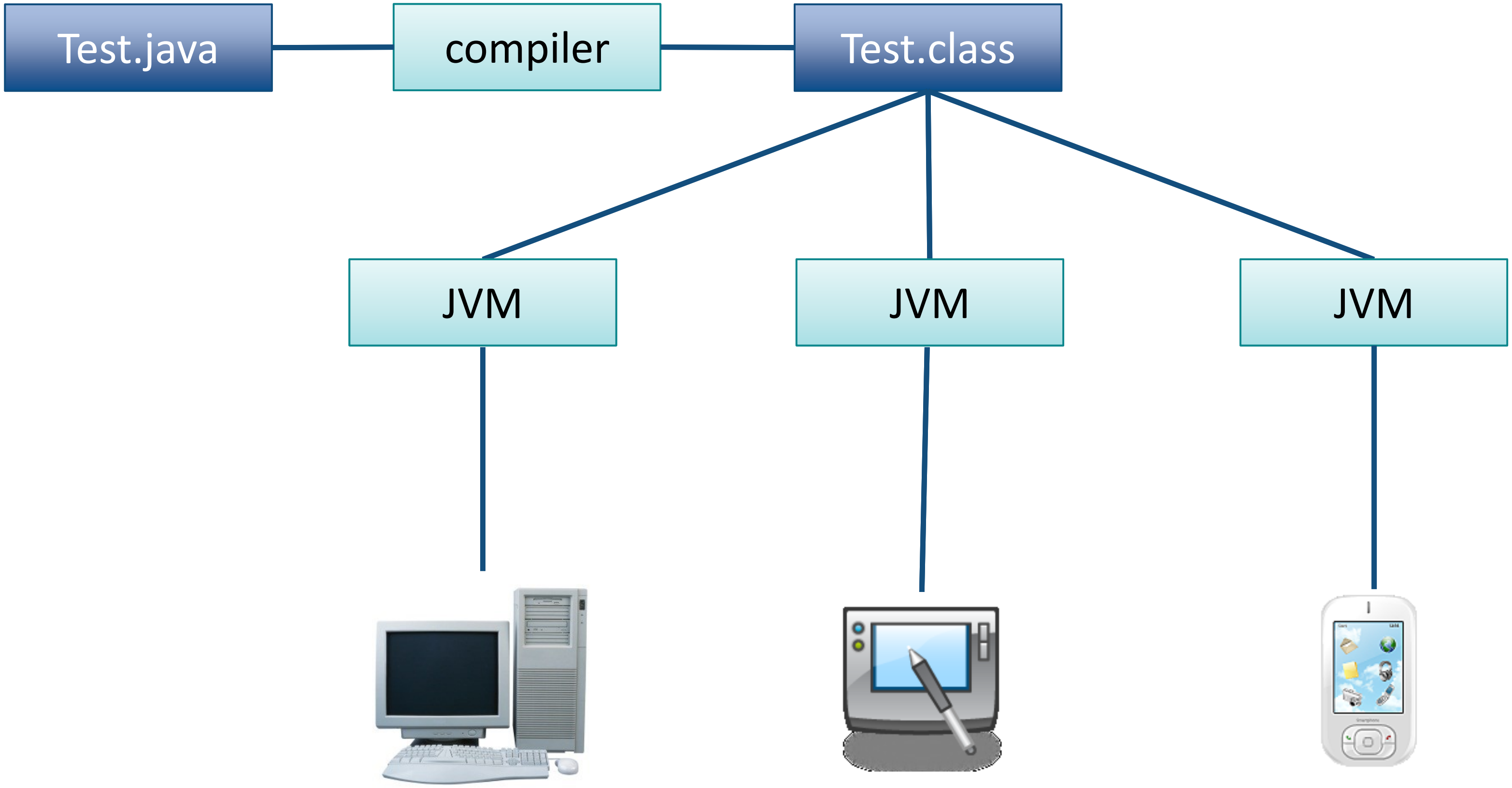
2011 Java 7

2014 Java 8

2017 Java 9

2018 Java 10/11

Java platform



The compiled code is independent of the architecture of the computer

A first example

```java
/**
 * * Hello World Application
 */
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");     // display output
    }
}
```

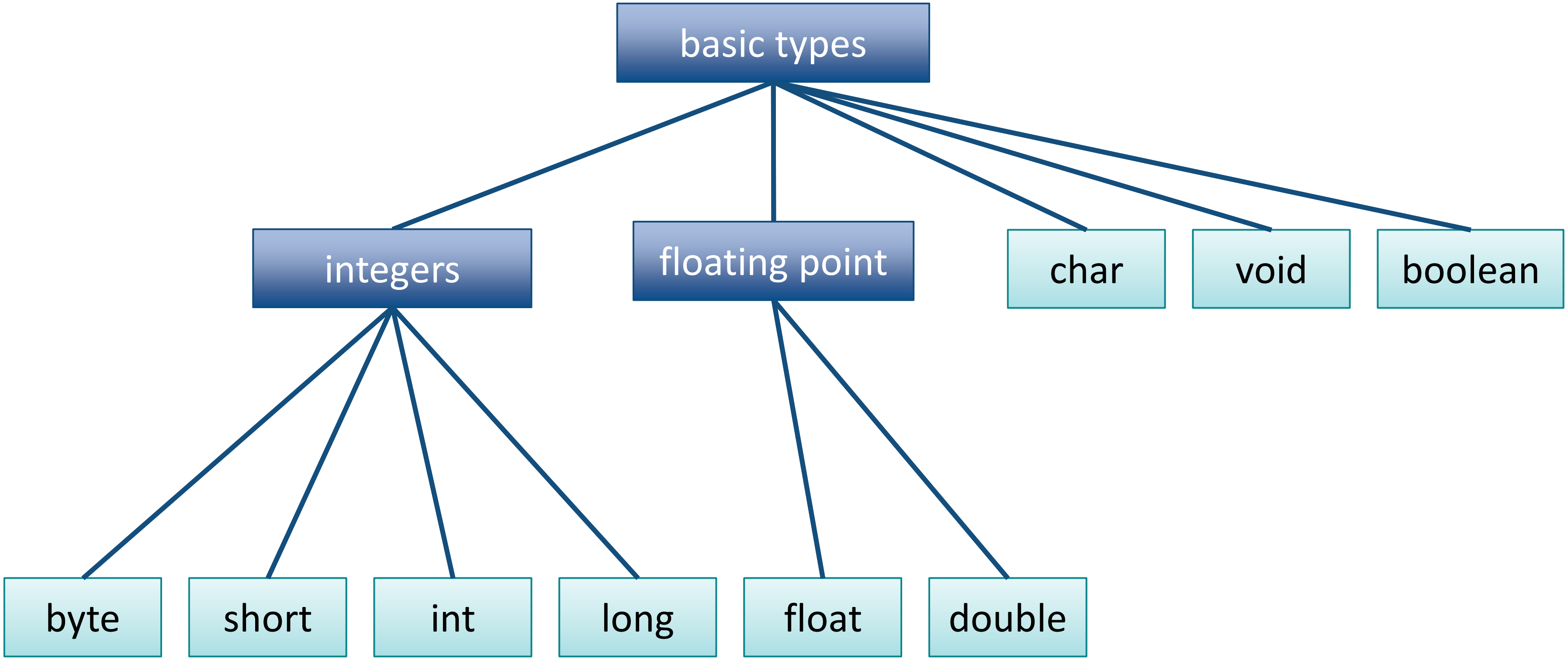
```
$ javac HelloWorld.java

$ ls
HelloWorld.class
HelloWorld.java

$ java HelloWorld
Hello World
```

Basic types

Java provides the following basic types



Variables and constants definition

```
int x;
double d = 0.33;
float f = 0.22F;
char c = 'a';
boolean ready = true;

x = 55;
```

The variables are declared specifying its type and name, and initialized in the point of declaration, or later with the assignment expression

```
final double pi = 3.1415;
final int maxSize = 100;
final char lastLetter = 'z';
```

Constants are declared with the word final in front. The specification of the initial value is compulsory

Strings

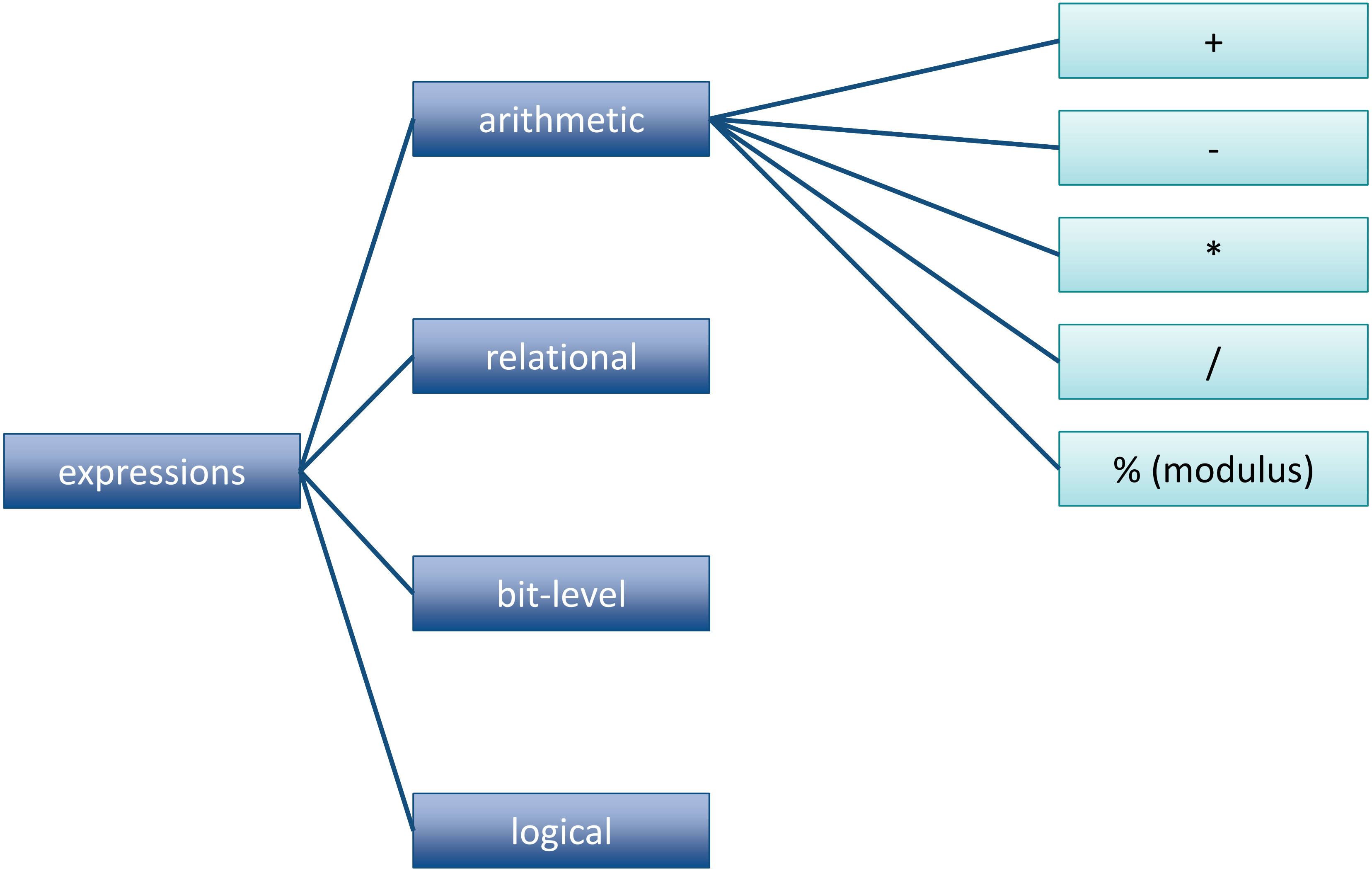
Strings are not a basic type, but defined as a class, more details later!

```
String a = "abc";
```

If the expression begins with a string and uses the + operator, then the next argument is converted to a string

```
int cost = 22;
String b = "the cost is " + cost + " euro";
```

Arithmetic expressions



Example with arithmetic operators

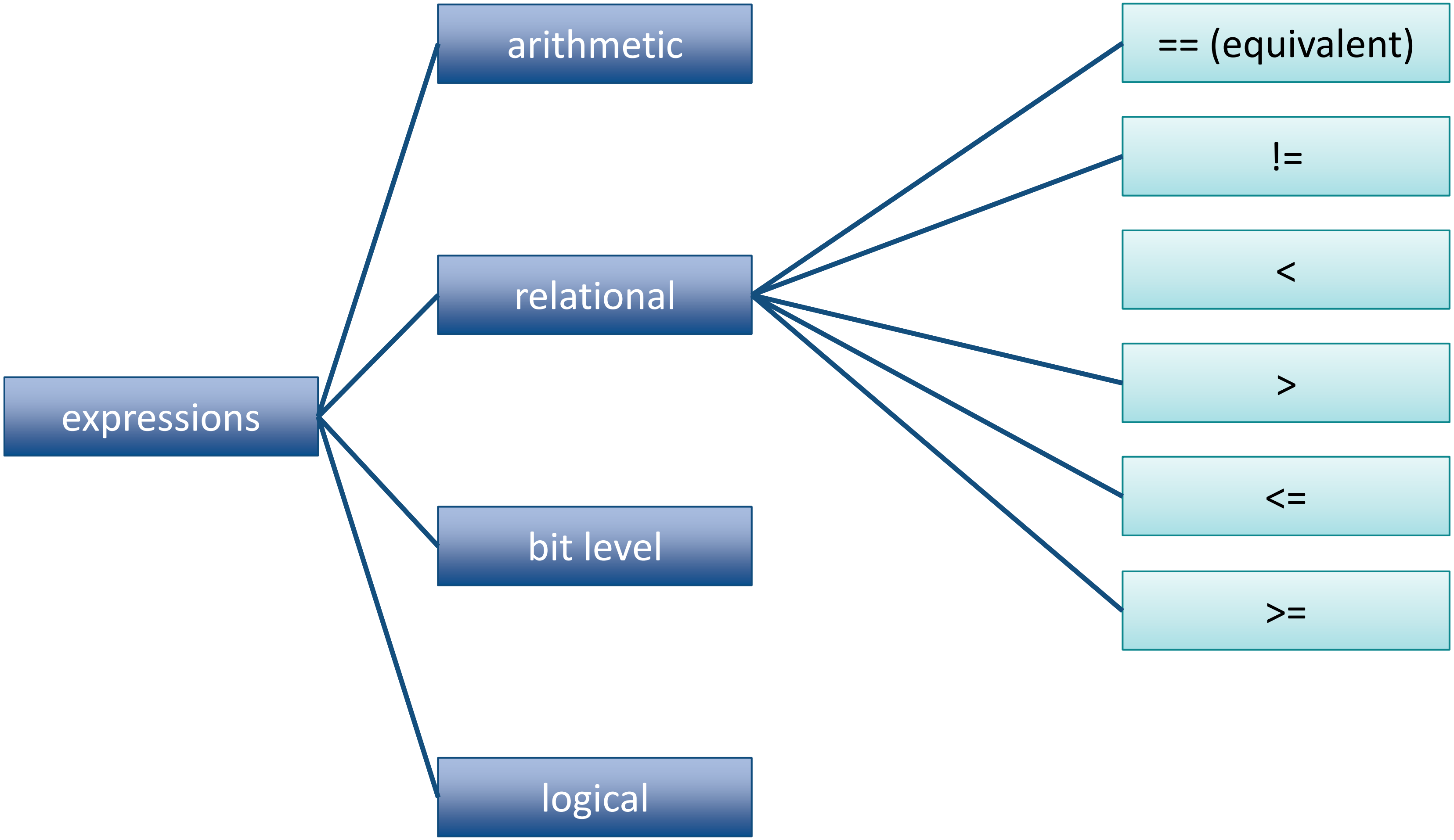
```
public class Arithmetic{
 public static void main(String[] args) {
  int x = 12;
  x += 5;                               // x = x + 5
  System.out.println(x);

  int a = 12,b = 12;
  System.out.print(a++);            // printed and then incremented
  System.out.print(a);

  System.out.print(++b);            // incremented and then printed
  System.out.println(b);
 }
}
```

```
$ java Arithmetic
17
12 13 13 13
```

Relational expressions


expressions

arithmetic

relational

bit level

logical

== (equivalent)

!=

<

>

<=

>=

Example with relational operators

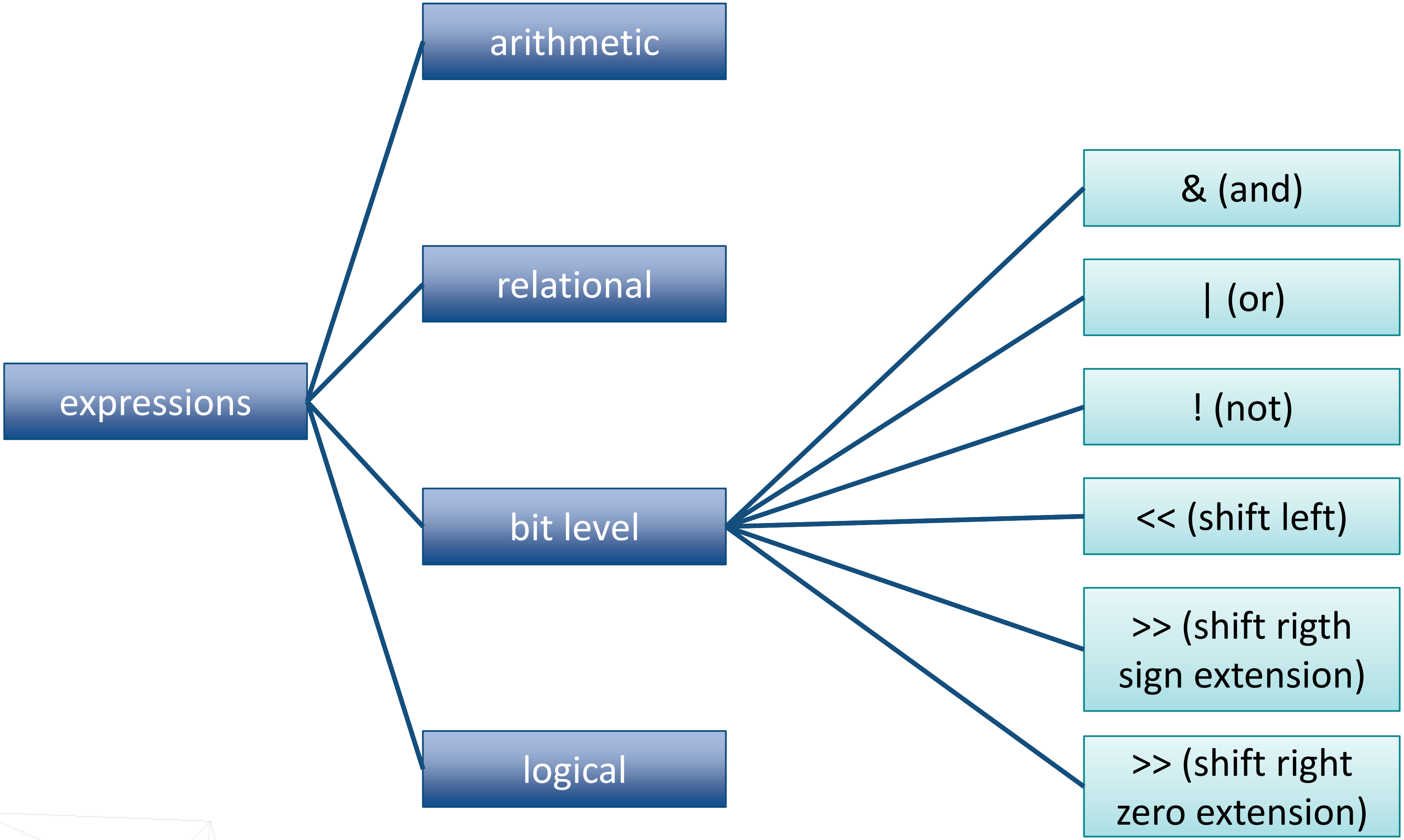
```java
public class Boolean {
  public static void main(String[] args) {
    int x = 12,y = 33;

    System.out.println(x < y);
    System.out.println(x != y - 21);

    boolean test = x >= 10;
    System.out.println(test);
  }
}
```

```
$ java Boolean
true
false
true
```

Bit level expressions



expressions

arithmetic

relational

bit level

logical

& (and)

| (or)

! (not)

<< (shift left)

>> (shift rigth sign extension)

>> (shift right zero extension)

Example with bit-level operators

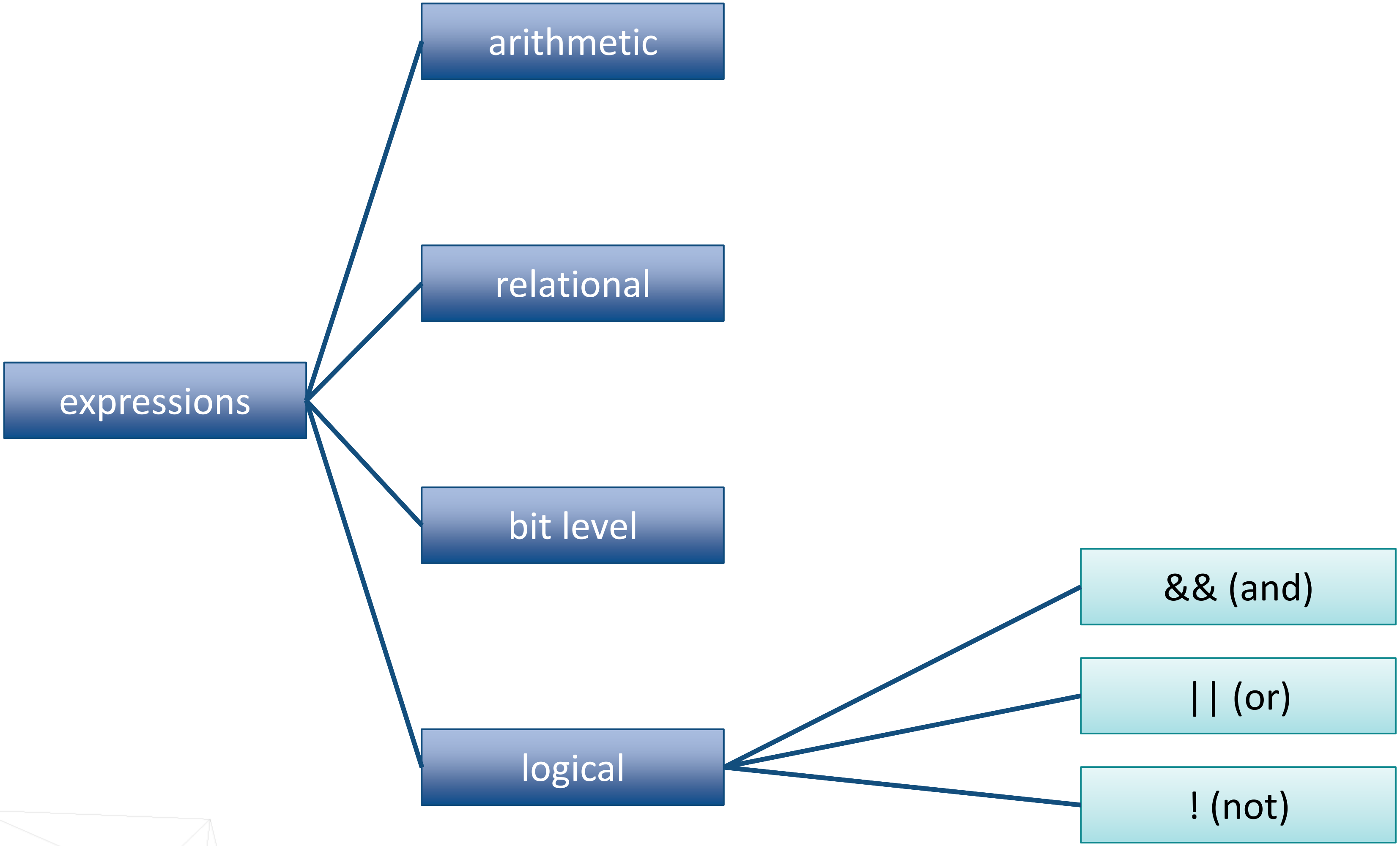
```
public class Bits {
 public static void main(String[] args) {
  int x = 0x16;                           // 00000000000000000000000000010110
  int y = 0x33;                           // 00000000000000000000000000110011

  System.out.println(x & y);              // 00000000000000000000000000010010
  System.out.println(x | y);              // 00000000000000000000000000110111
  System.out.println(˜x);                 // 11111111111111111111111111101001

   x = 9;                                  //00000000000000000000000000001001
  System.out.println(x >> 3);             //00000000000000000000000000000001
  System.out.println(x >>>3);             //00000000000000000000000000000001

  x = -9;                                  //11111111111111111111111111110111
  System.out.println(x >> 3);             //11111111111111111111111111111110
  System.out.println(x >>>3);             //00011111111111111111111111111110
 }
}
```

Logical expressions



Example with logical operators

```java
public class Logical {
  public static void main(String[] args) {
    int x = 12,y = 33;
    double d = 2.45,e = 4.54;

    System.out.println(x < y && d < e);
    System.out.println(!(x < y));

    boolean test = 'a' > 'z';
    System.out.println(test || d - 2.1 > 0);
  }
}
```

```
$ java Logical
true
false
true
```

Casting

Java performs a automatic type conversion when there is no risk for data to be lost.

In order to specify conversions where data can be lost it is necessary to use the cast operator.

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

    int a = 'x';          // 'x' is a character
    long b = 34;          // 34 is an int
    float c = 1002;       // 1002 is an int
    double d = 3.45F;     // 3.45F is a float

    long e = 34;
    int f = (int)e;       // e is a long
    double g = 3.45;
    float h = (float)g;   // g is a double
  }
}
```

Control structures: if

```java
public class If {
 public static void main(String[] args) {
  char c = 'x';

  if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))
   System.out.println("letter: " + c);
  else
   if (c >= '0' && c <= '9')
    System.out.println("digit: " + c);
   else {
    System.out.println("the character is: " + c);
    System.out.println("it is not a letter nor a digit");
   }
 }
}
```

```
$ java If
letter: x
```

Control structures: while

```java
public class While {
  public static void main(String[] args) {
    final float initialValue = 2.34F;
    final float step = 0.11F;
    final float limit = 4.69F;
    float var = initialValue;

    int counter = 0;
    while (var < limit) {
      var += step;
      counter++;
    }
    System.out.println("Incremented " + counter + " times");
  }
}
```

```
$ java While
Incremented 22 times
```

Control structures: for

```java
public class For {
 public static void main(String[] args) {
  final float initialValue = 2.34F;
  final float step = 0.11F;
  final float limit = 4.69F;
  int counter = 0;

  for (float var = initialValue;var < limit;var += step)
   counter++;
  System.out.println("Incremented " + counter + " times");
 }
}
```

```
$ java For
Incremented 22 times
```

Control structures: break/continue

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

    for (int counter = 0;counter < 10;counter++) {

      if (counter % 2 == 1) continue; // start a new iteration if the counter is odd
      if (counter == 8) break; // abandon the loop if the counter is equal to 8

      System.out.println(counter);
    }
    System.out.println("done.");
  }
}
```

```
$ java BreakContinue
0 2 4 6 done.
```

Control structures: switch

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

  boolean leapYear = true;
  int days = 0;

  for(int month = 1;month <= 12;month++) {
   switch(month) {
    case 1:// months with 31 days
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12: days += 31;
     break;
```

```
 case 2: // February is a special case
    if (leapYear)
     days += 29;
    else
     days += 28;
    break;
 default: // a month with 30 days
    days += 30;
    break;
   }
  }
  System.out.println(days);
 }
}
```

```
$ java Switch
366
```

The switch-expression must evaluate to byte, short, char, int, enum, or String

Arrays

Arrays can be used to store a number of elements of the same type

```
int[] a;
float[] b;
String[] c;
```

```
int[] a = {13,56,2034,4,55};
float[] b = {1.23F,2.1F};
String[] c = {"Java","is","great"};
```

Important: The declaration does not specify a size. However, it can be inferred when initialized

Other possibility to allocate space for arrays consists in the use of the operator new

```
int i = 3,j = 5;
double[] d;

d = new double[i+j];
```

Arrays

Components can be accessed with an integer index with values from 0 to length minus 1.

```
a[2] = 1000;
```

```
int len = a.length;
```

Every array has a member called length that can be used to get the length of the array

Components of the arrays are initialized with default values

```
int []a = new int[3];
for(int i = 0;i < a.length;i++)
  System.out.println(a[i]);
}
```

```
0
0
0
```

Arrays

```
public class Arrays {
 public static void main(String[] args) {
  int[] a = {2,4,3,1};

  // compute the summation of the elements of a
  int sum = 0;
  for(int i = 0;i < a.length;i++) sum += a[i];

  // create an array of the size computed before
  float[] d = new float[sum];
  for(int i = 0;i < d.length;i++) d[i] = 1.0F / (i+1);

   // print values in odd positions
  for(int i = 1;i < d.length;i += 2)
   System.out.println("d[" + i + "]=" + d[i]);
 }
}
```

```
$ java Arrays
d[1]=0.5
d[3]=0.25
d[5]=0.16666667
d[7]=0.125
d[9]=0.1
```

The for-each iteration

```java
public class ForEach {
  public static void main(String[] args) {
    int[] a = {2,4,3,1};

    // compute the summation of the elements of a
    int sum = 0;
    for(int x : a) sum += x;

    // create an array of the size computed before
    float[] d = new float[sum];
    for(int i = 0;i < d.length;i++) d[i] = 1.0F / (i+1);

     // print all values
    for(float f : d)
      System.out.println(f);
  }
}
```

Methods with variable number of arguments

A variable length argument list is specified with three periods:

```
int add(int … values) {
  int summation = 0;
  for(int i = 0;i < values.length;i++) {
     summation += values[i];
  }
  return summation;
}
```

```
int sum = add(1, 2, 3, 4, 5);
```

The argument is implicitly declared as an array, however, it can be called with a variable number of arguments

Thank you for your attention!

ESTECO

EXPLORE DESIGN PERFECTION

f 🐦 in YouTube v

esteco.com