Java: Build-In Data Types

Computer Science 2

January 31, 2017

Irina Shablinsky

Acknowledgement:

Some slides in this presentation are compliments

of Robert Sedgewick and Kevin Wayne

JetBrain IntelliJ IDEA

Available for OS X, Windows and Linux (http://jetbrains.com)

Code completion, great way to learn libraries

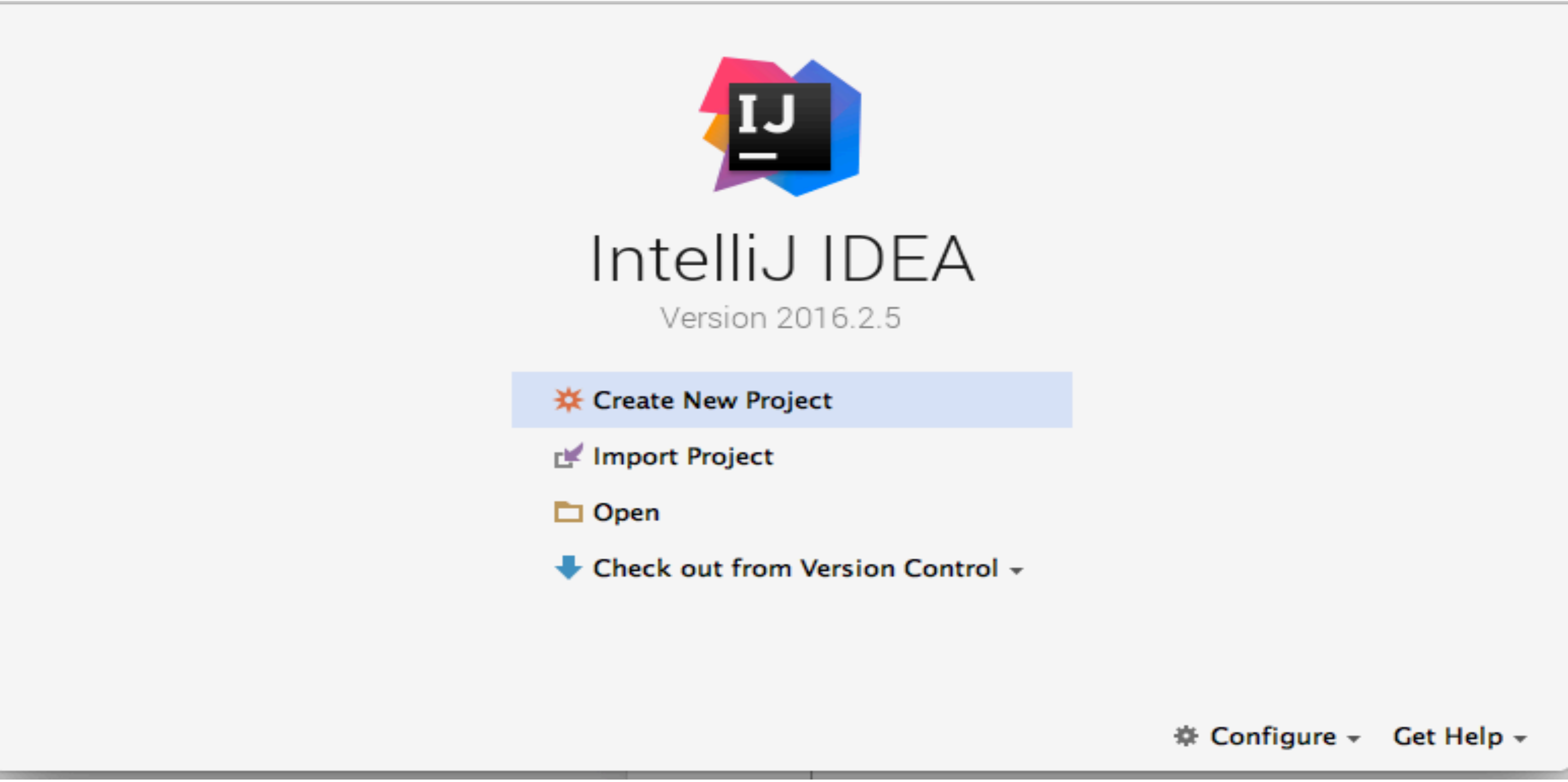
Syntax errors detected quickly, no more missing curly brackets

IntelliJ IDEA on your computer

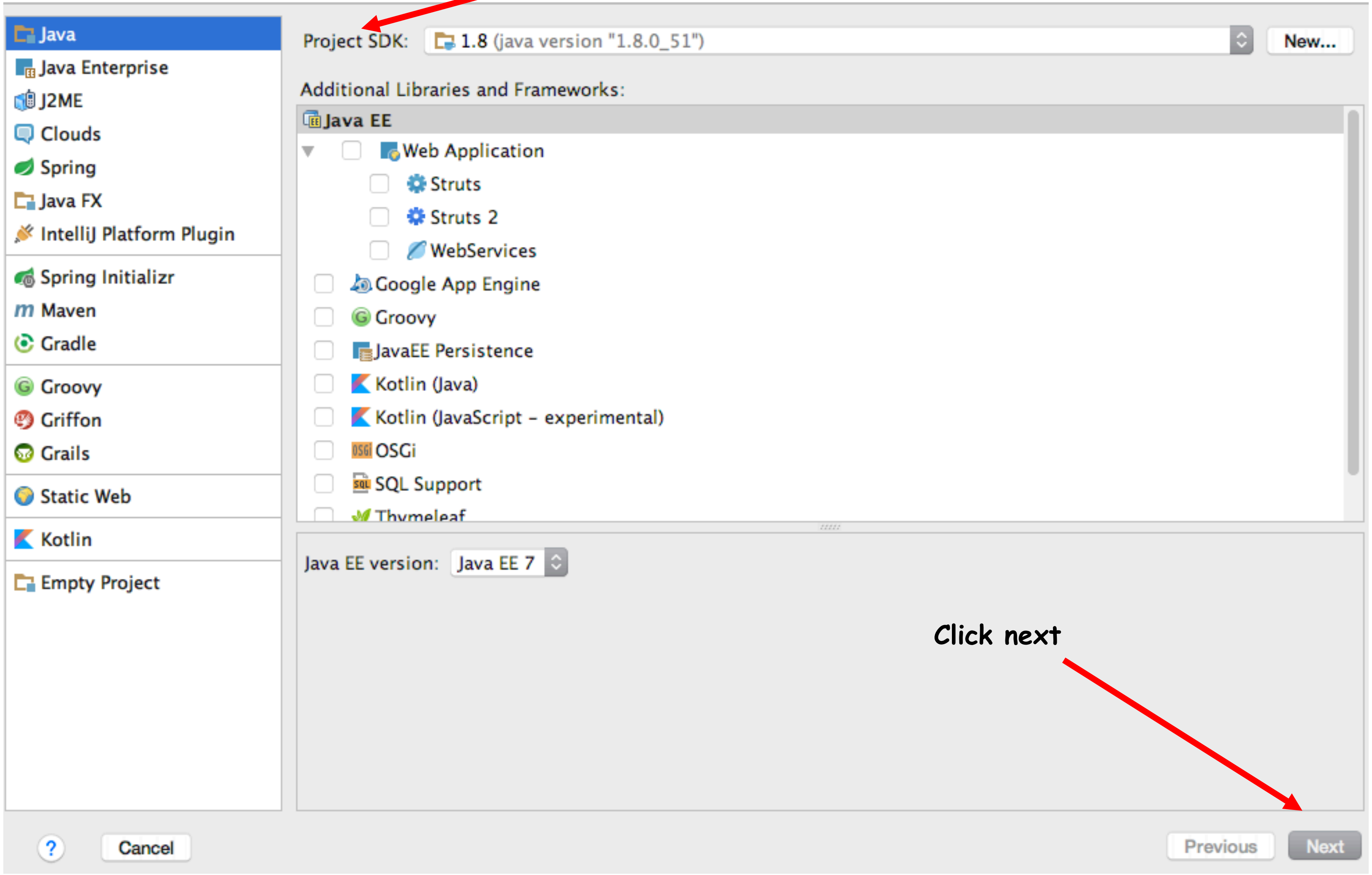
- Download and install the Java Development Kit (JDK 8) from http://www.oracle.com/technetwork/java/javase/downloads/index.html
- The latest version so far is **Java SE u121**
- Download and install the Ultimate Edition of the Jetbrains IntelliJ IDE from
- https://www.jetbrains.com/idea/download/
- Apply for a free full license https://www.jetbrains.com/shop/eform/students using your Purchase email

Configuring IDEA

https://www.jetbrains.com/idea/documentation/



Specify Project SDK



Project SDK: 1.8 (java version "1.8.0_51")

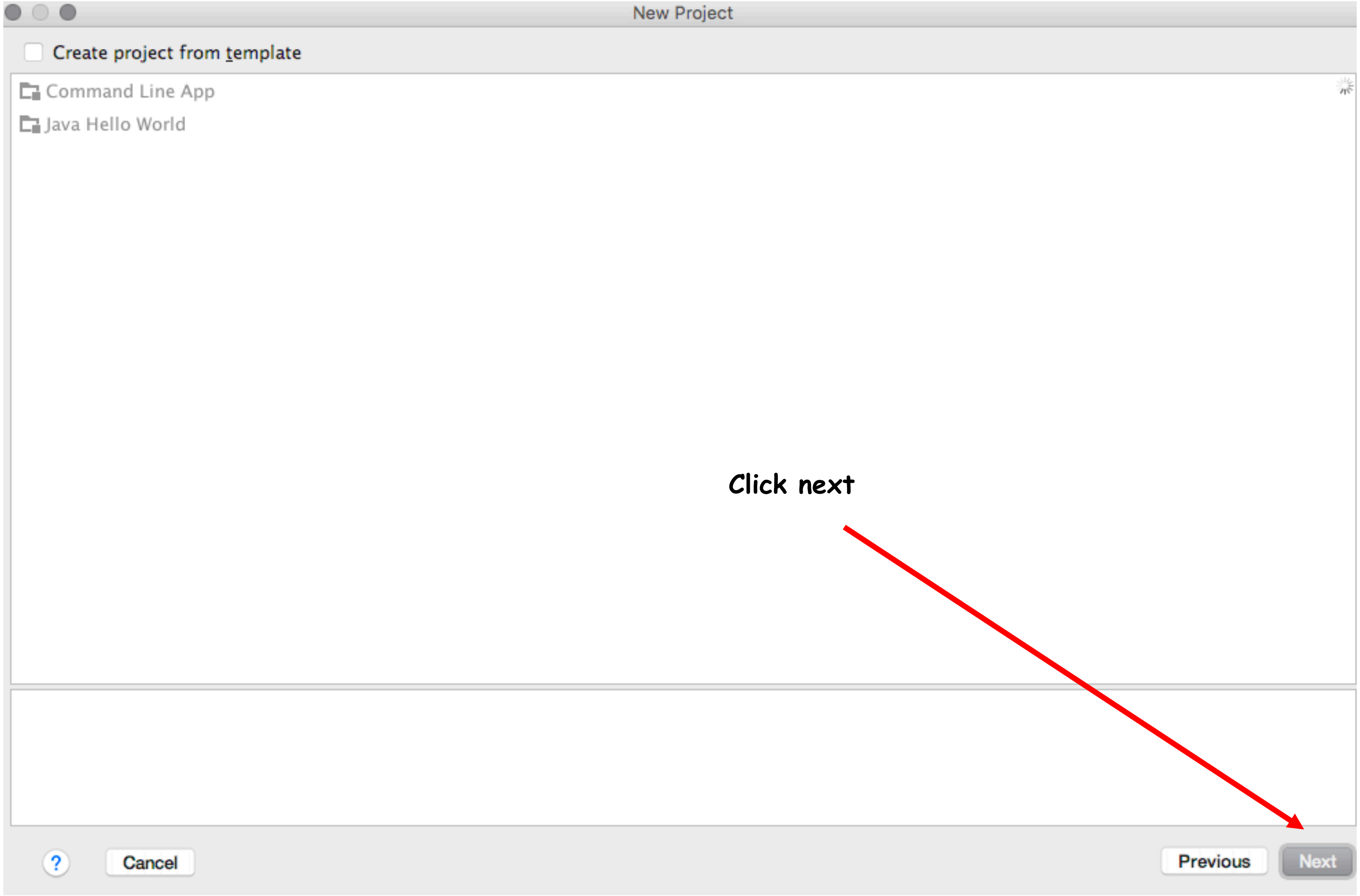
Additional Libraries and Frameworks:

Java EE

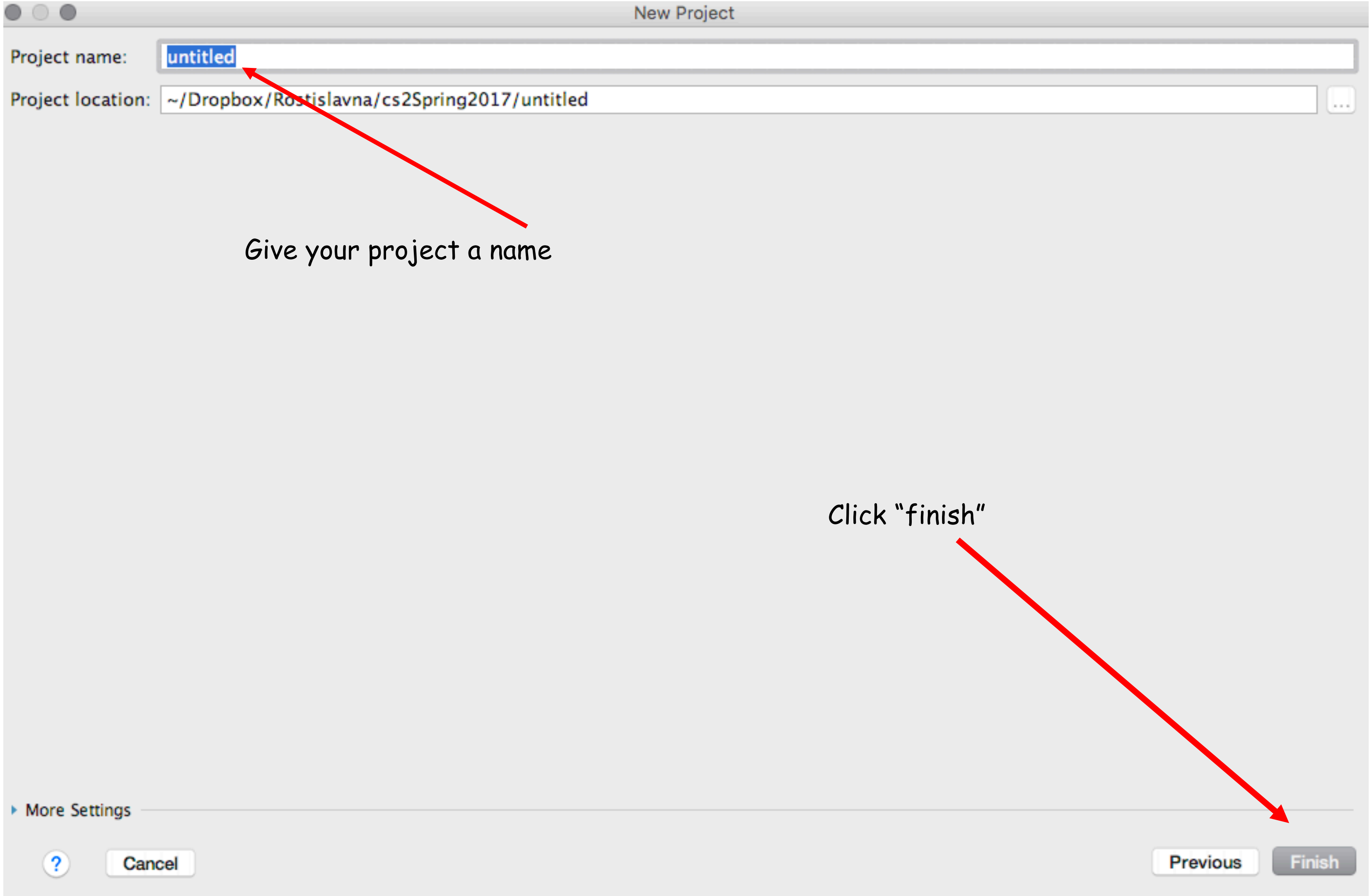
- □ Web Application
  - □ Struts
  - □ Struts 2
  - □ WebServices
- □ Google App Engine
- □ Groovy
- □ JavaEE Persistence
- □ Kotlin (Java)
- □ Kotlin (JavaScript - experimental)
- □ OSGi
- □ SQL Support
- □ Thymeleaf

Java EE version: Java EE 7

Click next


New Project
□ Create project from template
Command Line App
Java Hello World
?
Cancel
Previous
Next

Click next

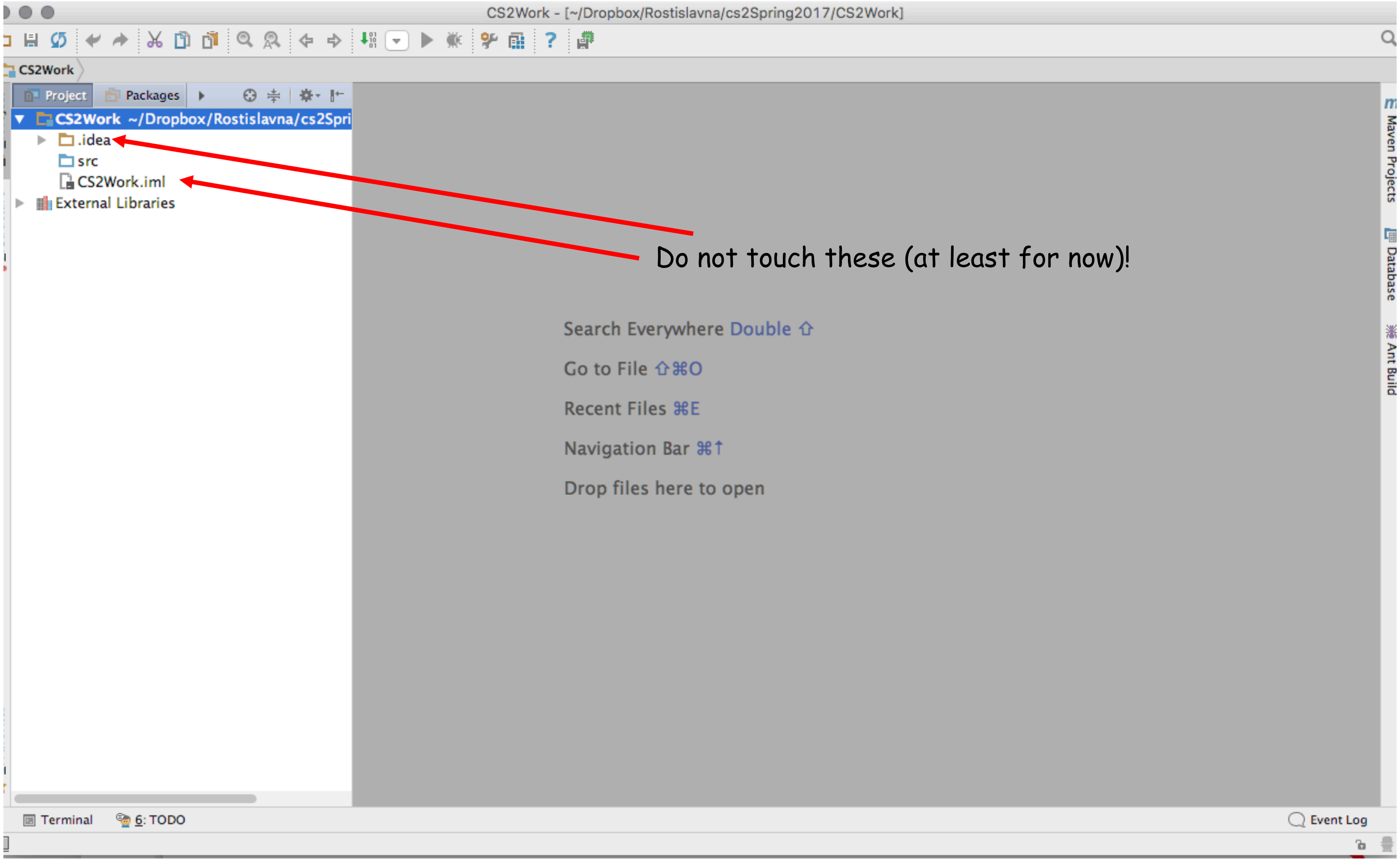


Project name: untitled

Project location: ~/Dropbox/Rostislavna/cs2Spring2017/untitled

Give your project a name

Click “finish”



Do not touch these (at least for now)!

Look at the Project Structure

A **project** is usually a collection of packages

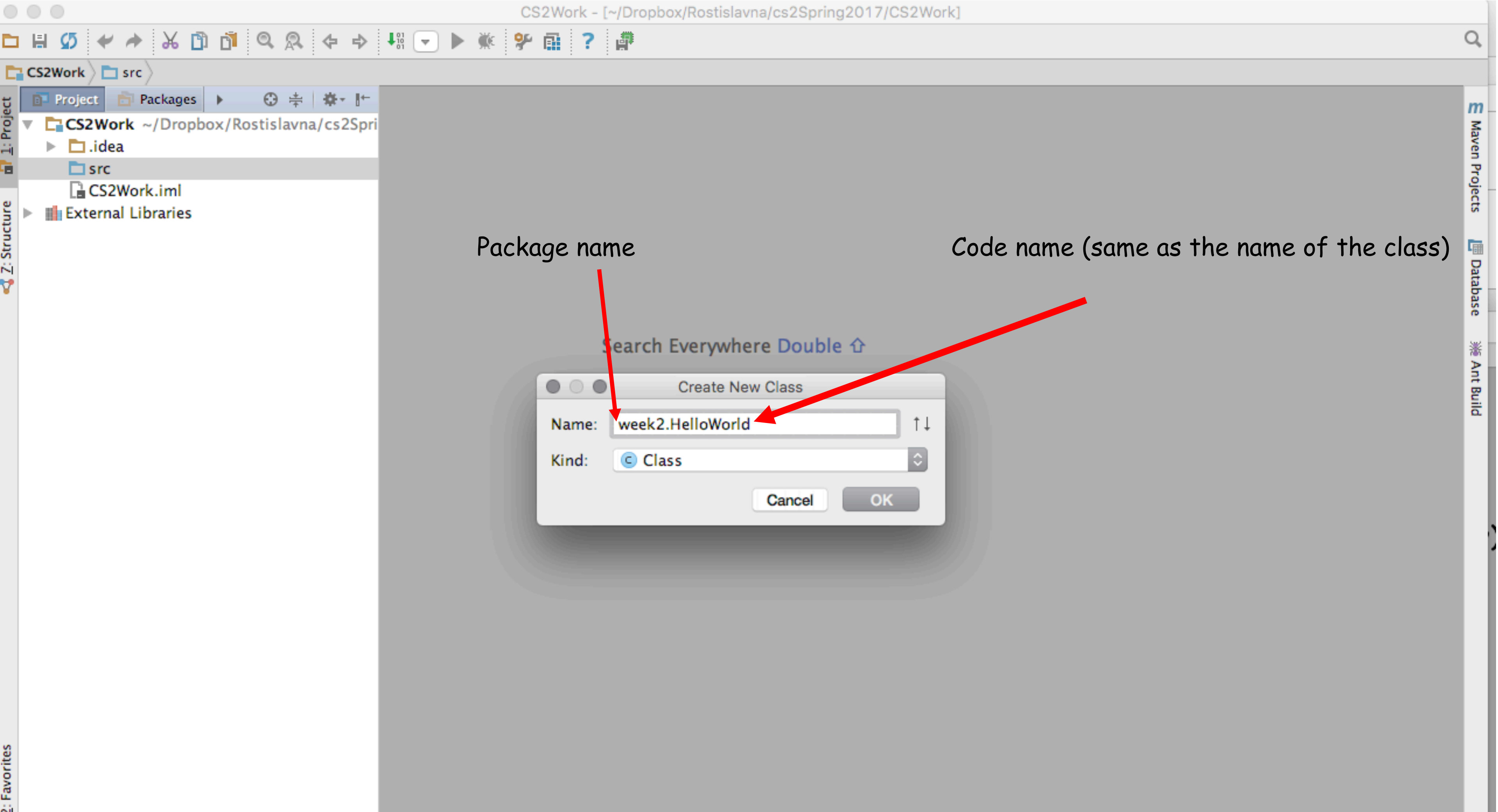
A **package** is a group of related classes

Packages help to organize your code

Libraries are collections of "helper" code you can use in your program. They usually have a .jar extension

Create a New Package and New Sourse Code

File➔New➔Java Class



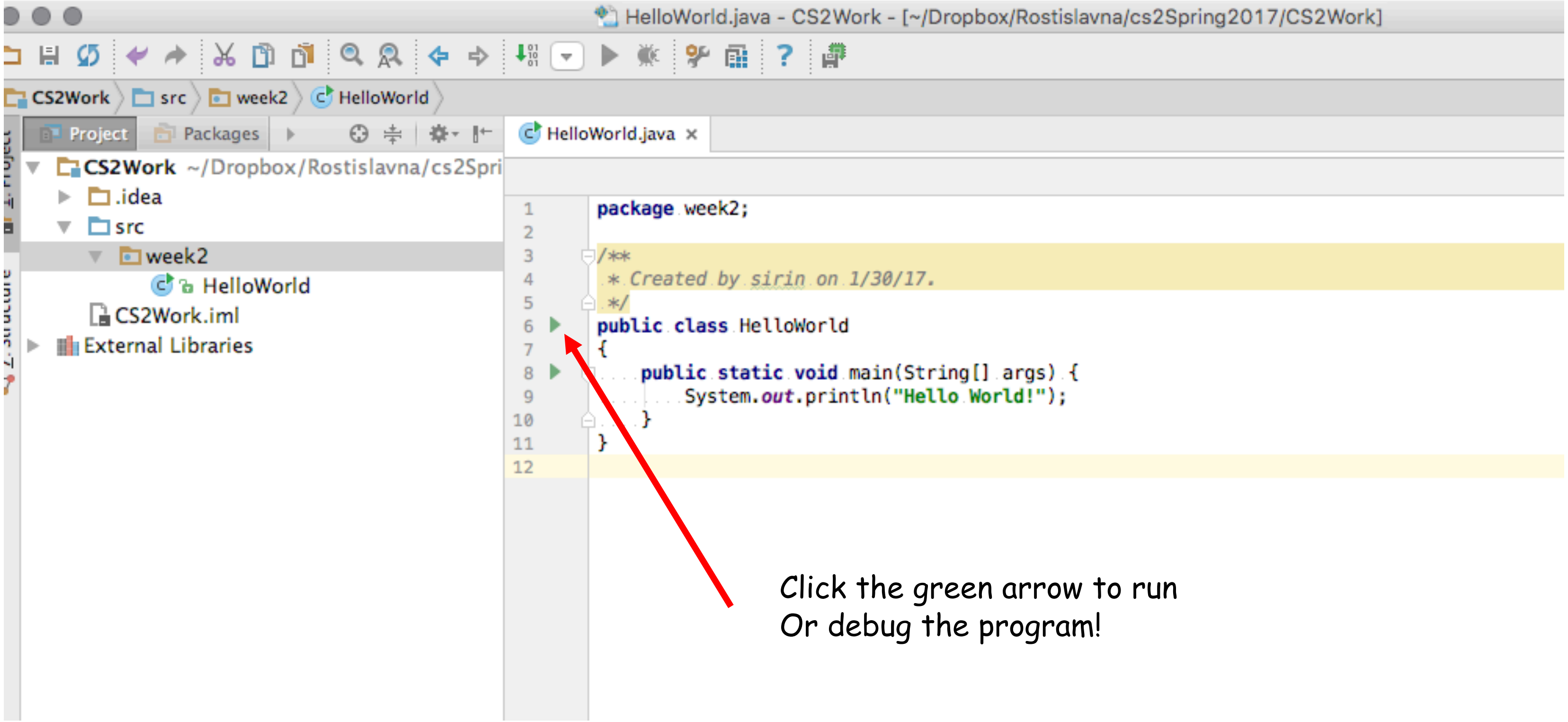
Package name

Code name (same as the name of the class)

Name: week2.HelloWorld

Kind: Class

Compile and Run!

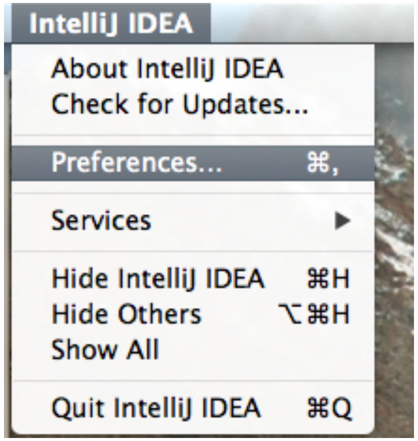


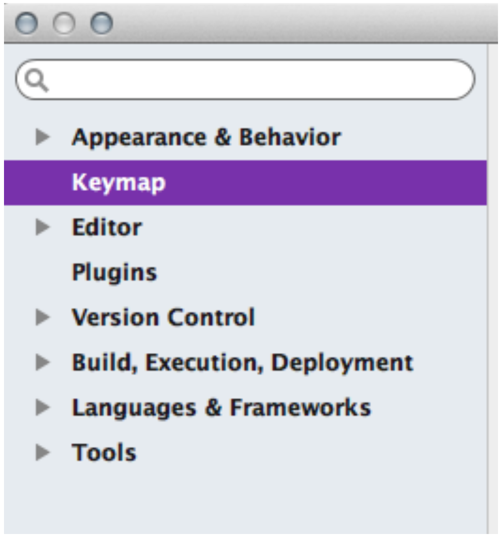
HelloWorld.java - CS2Work - [~/Dropbox/Rostislavna/cs2Spring2017/CS2Work]
CS2Work
src
week2
HelloWorld
Project
Packages
HelloWorld.java
CS2Work ~/Dropbox/Rostislavna/cs2Spri
.idea
src
week2
HelloWorld
CS2Work.iml
External Libraries

```
package week2;

/**
 * Created by sirin on 1/30/17.
 */
public class HelloWorld
{
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Click the green arrow to run
Or debug the program!

You Can Create your Own Keyboard Shortcuts





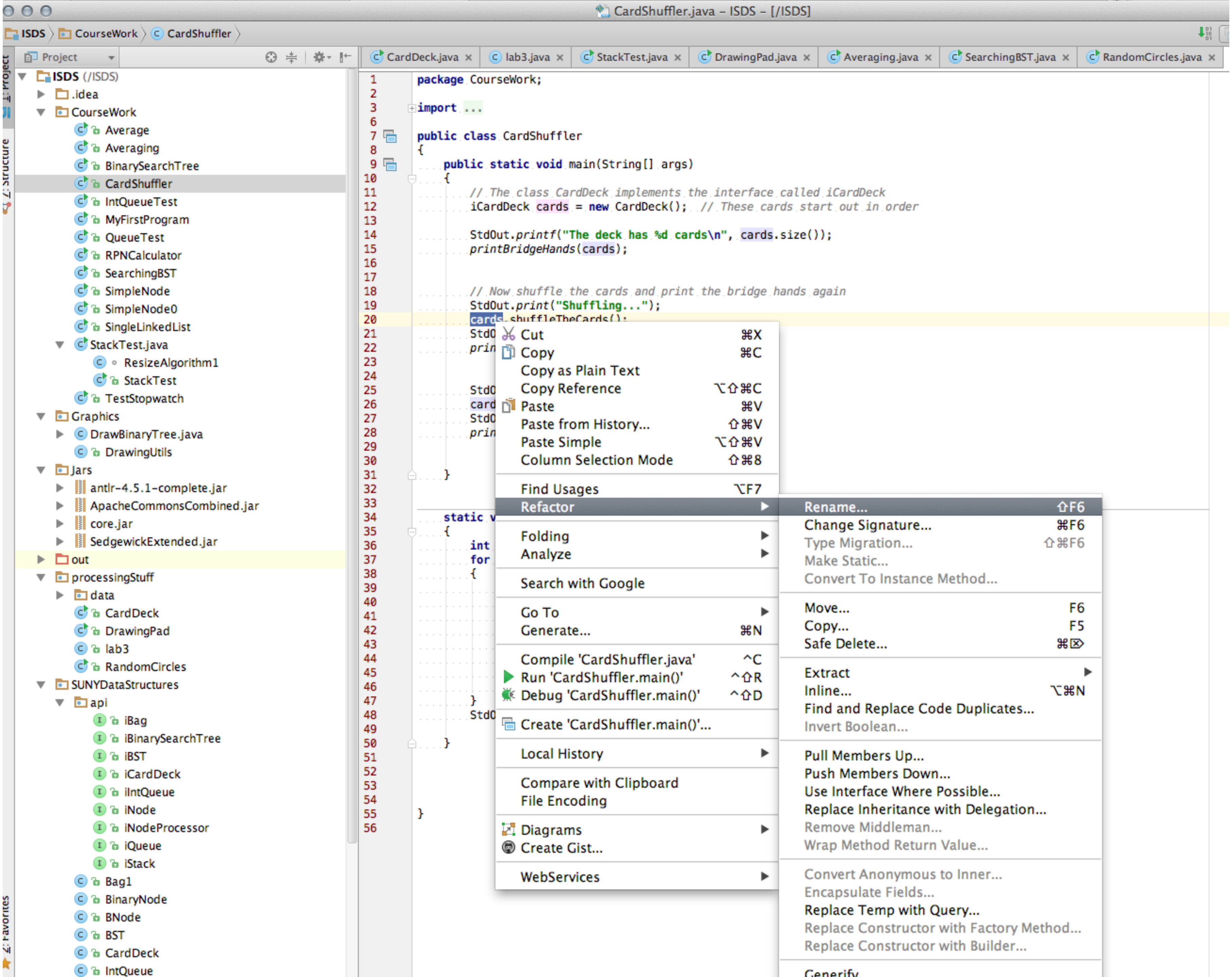
Open Preferences (and go to the Keymap section

Create your own shortcuts to compile and run the programs

Refactor: renaming

The ability to quickly rename an identifier globally is incredibly useful. IDEA will make sure that ALL occurrences of the identifier are renamed even in different files.

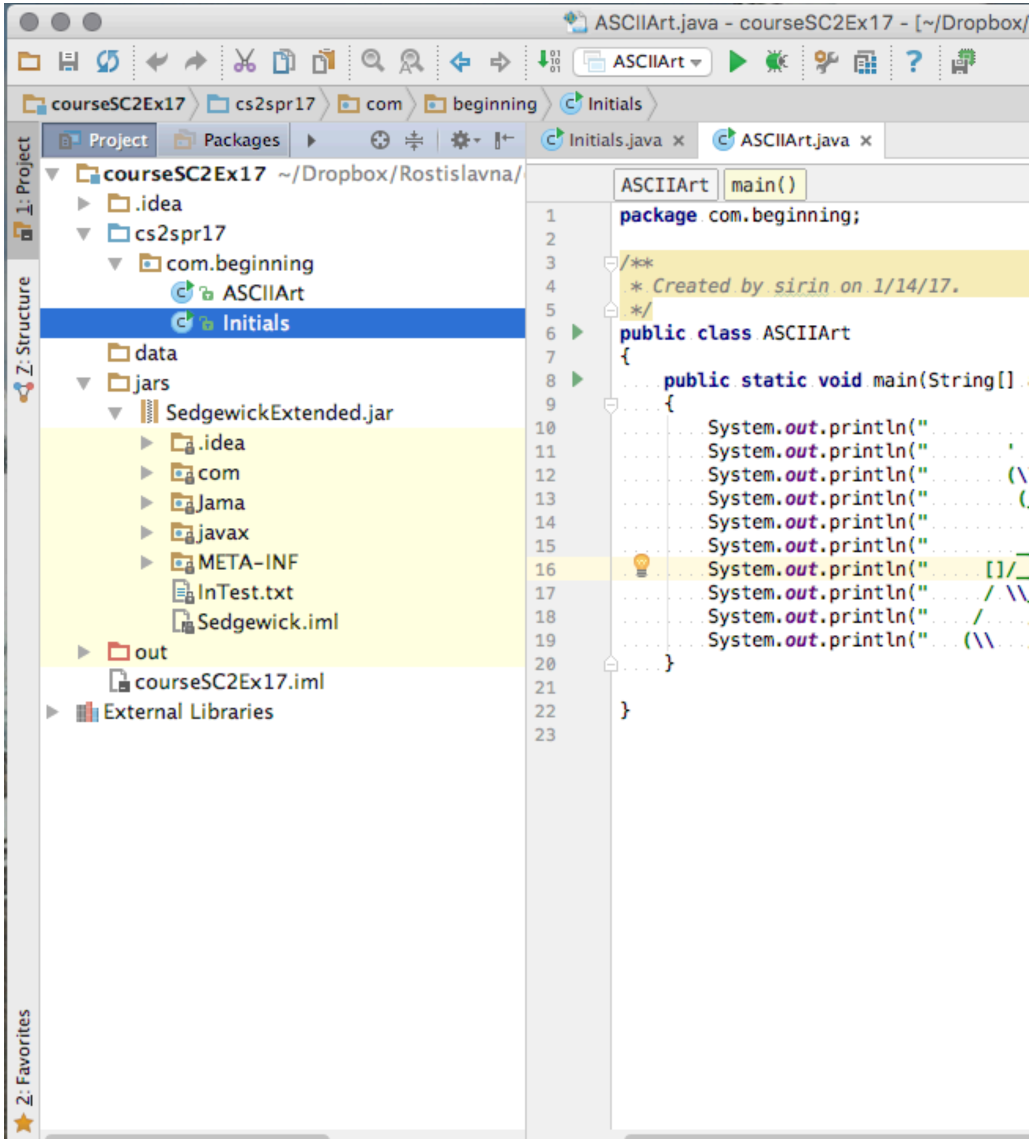
No more silly variable names please!



```
package CourseWork;

import ...

public class CardShuffler
{
    public static void main(String[] args)
    {
        // The class CardDeck implements the interface called iCardDeck
        iCardDeck cards = new CardDeck();  // These cards start out in order

        StdOut.printf("The deck has %d cards\n", cards.size());
        printBridgeHands(cards);


        // Now shuffle the cards and print the bridge hands again
        StdOut.print("Shuffling...");
        cards.shuffleTheCards();
        StdO
        prin



        StdO
        card
        StdO
        prin


    }


    static v
    {
        int
        for
        {











        }
        StdO

    }





}
```

Cut ⌘X
Copy ⌘C
Copy as Plain Text
Copy Reference ⌥⇧⌘C
Paste ⌘V
Paste from History... ⇧⌘V
Paste Simple ⌥⇧⌘V
Column Selection Mode ⇧⌘8
Find Usages ⌥F7
Refactor
Folding
Analyze
Search with Google
Go To
Generate... ⌘N
Compile 'CardShuffler.java' ^C
Run 'CardShuffler.main()' ^⇧R
Debug 'CardShuffler.main()' ^⇧D
Create 'CardShuffler.main()'...
Local History
Compare with Clipboard
File Encoding
Diagrams
Create Gist...
WebServices
Rename... ⇧F6
Change Signature... ⌘F6
Type Migration... ⇧⌘F6
Make Static...
Convert To Instance Method...
Move... F6
Copy... F5
Safe Delete... ⌘⌦
Extract
Inline... ⌥⌘N
Find and Replace Code Duplicates...
Invert Boolean...
Pull Members Up...
Push Members Down...
Use Interface Where Possible...
Replace Inheritance with Delegation...
Remove Middleman...
Wrap Method Return Value...
Convert Anonymous to Inner...
Encapsulate Fields...
Replace Temp with Query...
Replace Constructor with Factory Method...
Replace Constructor with Builder...
Generify

Adding more Libraries



```
courseSC2Ex17 ~/Dropbox/Rostislavna/
    .idea
    cs2spr17
        com.beginning
            ASCIIArt
            Initials
    data
    jars
        SedgewickExtended.jar
            .idea
            com
            Jama
            javax
            META-INF
            InTest.txt
            Sedgewick.iml
    out
    courseSC2Ex17.iml
External Libraries
```

```java
package com.beginning;

/**
 * Created by sirin on 1/14/17.
 */
public class ASCIIArt
{
    public static void main(String[]
    {
        System.out.println("
        System.out.println("       '
        System.out.println("       (\
        System.out.println("        (
        System.out.println("
        System.out.println("        _
        System.out.println("    []/_
        System.out.println("    / \\
        System.out.println("   /
        System.out.println("  (\\
    }

}
```

Built-in Data Types

Data type. A set of values and operations defined on those values.

type	set of values	literal values	operations
`char`	characters	`'A'` `'@'`	compare

`String`	sequences of characters	`"Hello World"`
		`"126 is fun"`
			concatenate

`int`	integers	`17`
		`12345`
			add, subtract, multiply, divide

`double`	floating-point numbers	`3.1415`
		`6.022e23`
			add, subtract, multiply, divide

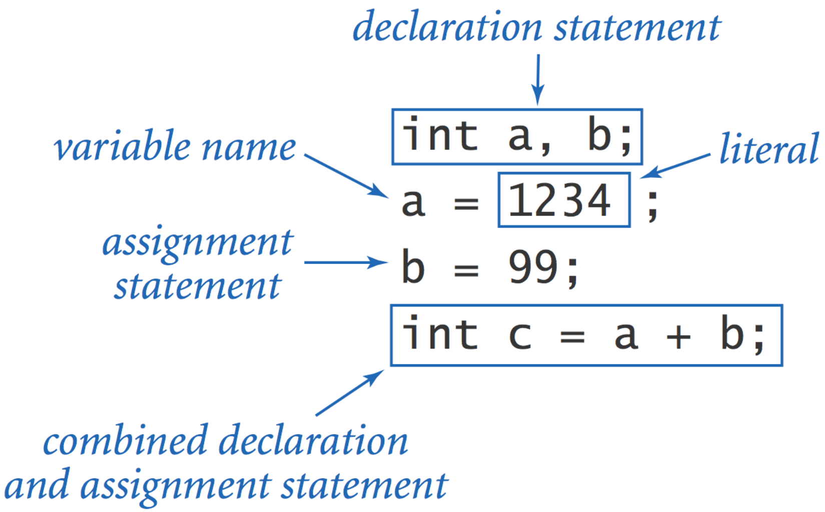
`boolean`	truth values	`true` `false`	and, or, not

Basic Definitions

Variable. A name that refers to a value of declared type.

Literal. Programming language representation of a value.

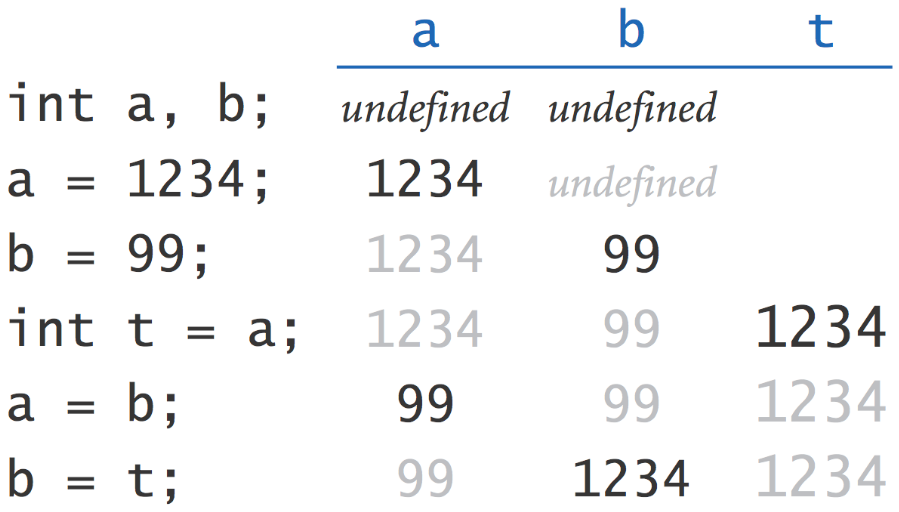
Assignment statement. Associates a value with a variable.



```
int a, b;
a = 1234 ;
b = 99;
int c = a + b;
```

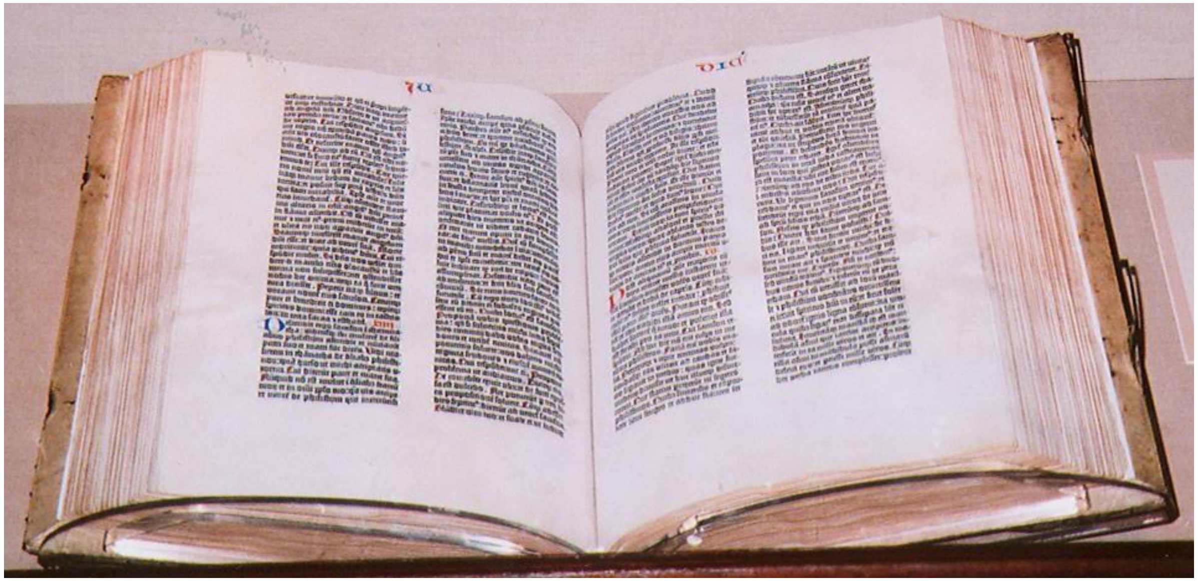
Trace

Trace. Table of variable values after each statement.



	a	b	t
`int a, b;`	*undefined*	*undefined*	
`a = 1234;`	1234	*undefined*	
`b = 99;`	1234	99	
`int t = a;`	1234	99	1234
`a = b;`	99	99	1234
`b = t;`	99	1234	1234

Text



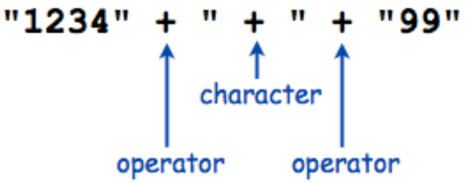
Text

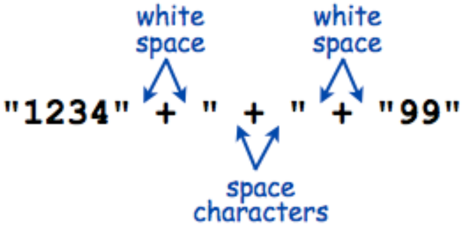
`String` data type. Useful for program input and output.

values	sequences of characters
typical literals	`"Hello," "1 " " * "`
operation	concatenate
operator	`+`

expression	*value*
`"Hi, " + "Bob"`	`"Hi, Bob"`
`"1" + " 2 " + "1"`	`"1 2 1"`
`"1234" + " + " + "99"`	`"1234 + 99"`
`"1234" + "99"`	`"123499"`

Caveat. Meaning of characters depends on context.



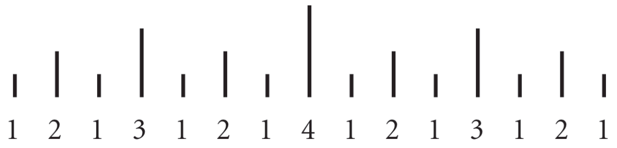


Subdivisions of a Ruler

```
public class Ruler {
   public static void main(String[] args) {
      String ruler1 = "1";
      String ruler2 = ruler1 + " 2 " + ruler1;
      String ruler3 = ruler2 + " 3 " + ruler2;
      String ruler4 = ruler3 + " 4 " + ruler3;
      System.out.println(ruler4);
   }
}
```

```
      "1"
    "1 2 1"
"1 2 1 3 1 2 1"
```

string concatenation



Integers

…, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, …

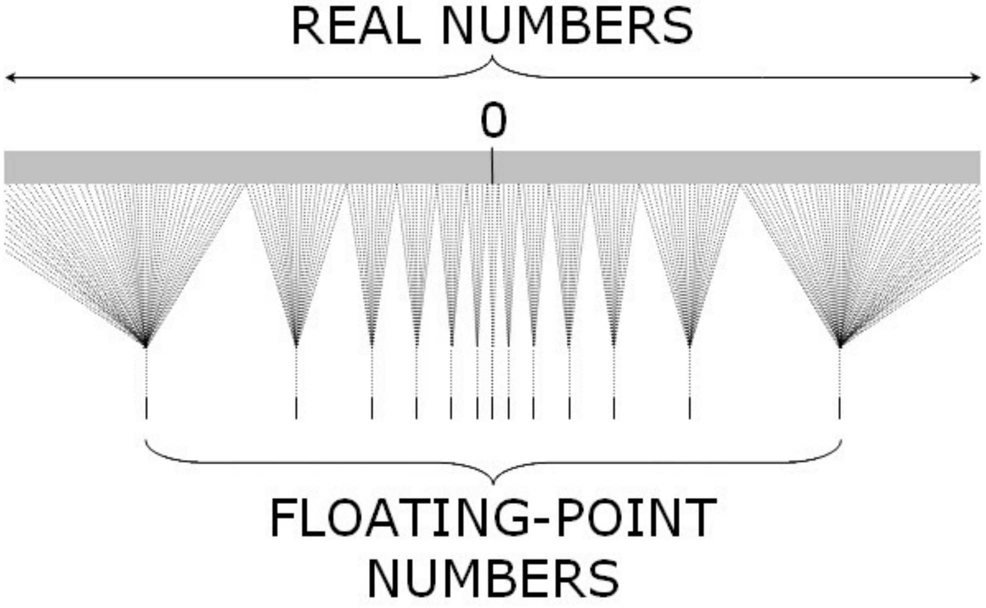
Integers

`int` data type. Useful for expressing algorithms.

values	integers between -2^{31} and $+2^{31}-1$				
typical literals	`1234` `99` `-99` `0` `1000000`				
operations	add	subtract	multiply	divide	remainder
operators	`+`	`-`	`*`	`/`	`%`

expression	*value*	*comment*
`5 + 3`	`8`	
`5 - 3`	`2`	
`5 * 3`	`15`	
`5 / 3`	`1`	no fractional part
`5 % 3`	`2`	remainder
`1 / 0`		run-time error
`3 * 5 - 2`	`13`	`*` has precedence
`3 + 5 / 2`	`5`	`/` has precedence
`3 - 5 - 2`	`-4`	left associative
`( 3 - 5 ) - 2`	`-4`	better style
`3 - ( 5 - 2 )`	`0`	unambiguous

Floating-Point Numbers



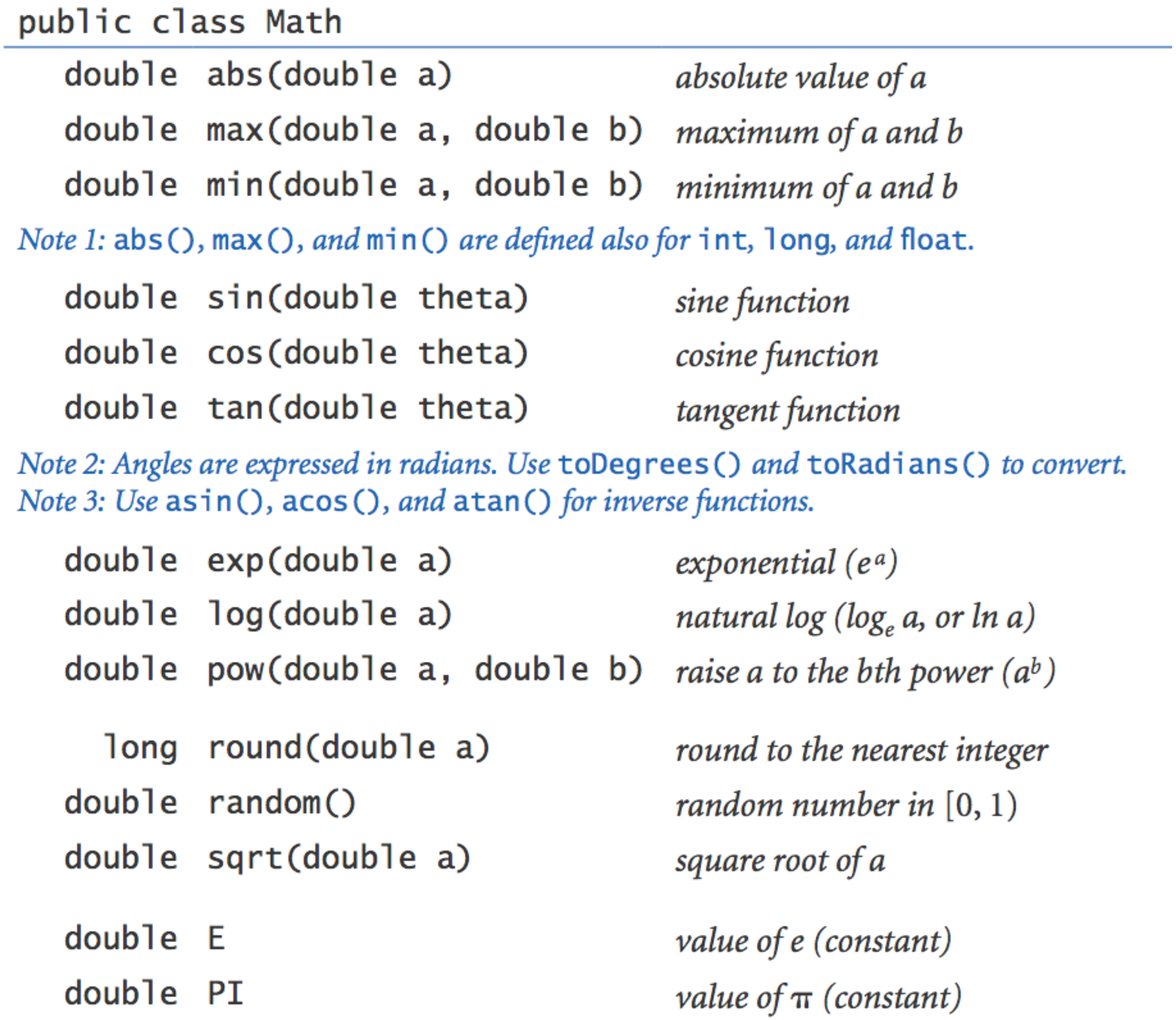
Floating-Point Numbers

`double` data type. Useful in scientific applications.

values	real numbers (specified by IEEE 754 standard)			
typical literals	`3.14159` `6.022e23` `-3.0` `2.0` `1.414213562373095`			
operations	add	subtract	multiply	divide
operators	`+`	`-`	`*`	`/`

expression	*value*
`3.141 + .03`	`3.171`
`3.141 - .03`	`3.111`
`6.02e23 / 2.0`	`3.01e23`
`5.0 / 3.0`	`1.6666666666666667`
`10.0 % 3.141`	`0.577`
`1.0 / 0.0`	`Infinity`
`Math.sqrt(2.0)`	`1.4142135623730951`
`Math.sqrt(-1.0)`	`NaN`

Excerpts from Java’s Math Library



`public class Math`

`double abs(double a)`	*absolute value of a*
`double max(double a, double b)`	*maximum of a and b*
`double min(double a, double b)`	*minimum of a and b*

Note 1: `abs()`, `max()`, *and* `min()` *are defined also for* `int`, `long`, *and* `float`.

`double sin(double theta)`	*sine function*
`double cos(double theta)`	*cosine function*
`double tan(double theta)`	*tangent function*

Note 2: Angles are expressed in radians. Use `toDegrees()` *and* `toRadians()` *to convert.*
Note 3: Use `asin()`, `acos()`, *and* `atan()` *for inverse functions.*

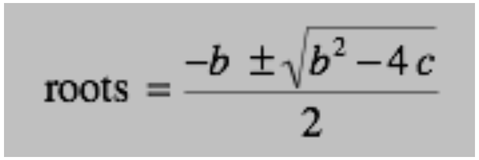
`double exp(double a)`	*exponential* (e^a)
`double log(double a)`	*natural log* ($\log_e a$, *or* $\ln a$)
`double pow(double a, double b)`	*raise a to the bth power* (a^b)
`long round(double a)`	*round to the nearest integer*
`double random()`	*random number in* $[0, 1)$
`double sqrt(double a)`	*square root of a*
`double E`	*value of e (constant)*
`double PI`	*value of* π *(constant)*

http://download.oracle.com/javase/6/docs/api/java/lang/Math.html

Quadratic Equation

Ex. Solve quadratic equation $x^2 + bx + c = 0$

See Quadratic.java code



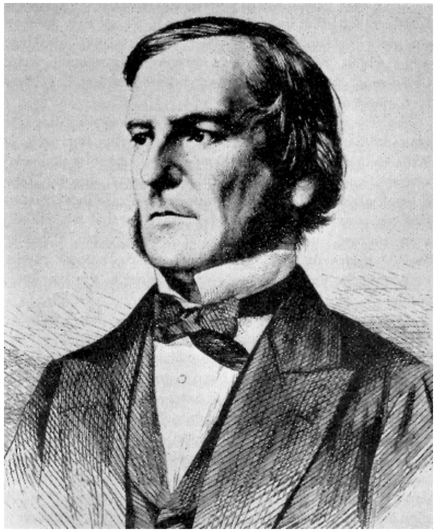
$$\text{roots} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

Testing

$x^2 – 3x + 2$ $x^2 + x + 1$

$x^2 – x - 1$

Booleans



Booleans

`boolean` data type. Useful to control logic and flow of a program.

values	true or false		
literals	`true` `false`		
operations	and	or	not
operators	&&	||	!

a	!a
true	false
false	true

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Comparisons

Comparisons. Take two operands of one type (e.g., `int`) and produce a result of type `boolean`.

op	*meaning*	`true`	`false`
`==`	*equal*	`2 == 2`	`2 == 3`
`!=`	*not equal*	`3 != 2`	`2 != 2`
`<`	*less than*	`2 < 13`	`2 < 2`
`<=`	*less than or equal*	`2 <= 2`	`3 <= 2`
`>`	*greater than*	`13 > 2`	`2 > 13`
`>=`	*greater than or equal*	`3 >= 2`	`2 >= 3`

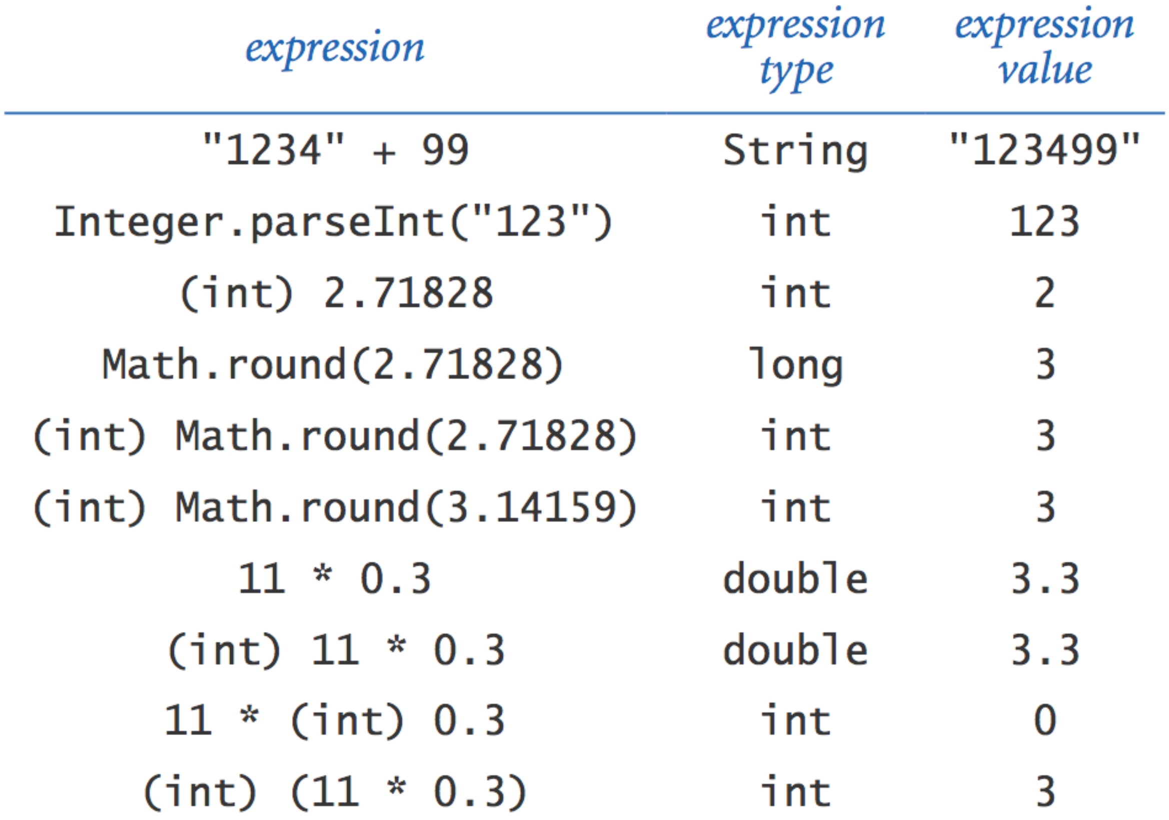
non-negative discriminant?	`(b*b - 4.0*a*c) >= 0.0`
beginning of a century?	`(year % 100) == 0`
legal month?	`(month >= 1) && (month <= 12)`

Type Conversion

Type Conversion

Type conversion. Convert value from one data type to another.

- Automatic: no loss of precision; or with string
- Expli



expression	*expression type*	*expression value*
`"1234" + 99`	`String`	`"123499"`
`Integer.parseInt("123")`	`int`	`123`
`(int) 2.71828`	`int`	`2`
`Math.round(2.71828)`	`long`	`3`
`(int) Math.round(2.71828)`	`int`	`3`
`(int) Math.round(3.14159)`	`int`	`3`
`11 * 0.3`	`double`	`3.3`
`(int) 11 * 0.3`	`double`	`3.3`
`11 * (int) 0.3`	`int`	`0`
`(int) (11 * 0.3)`	`int`	`3`

Random Integer

Ex. Generate a pseudo-random number between `0` and `N-1`.

RandomInt.java

Summary

A data type is a set of values and operations on those values.

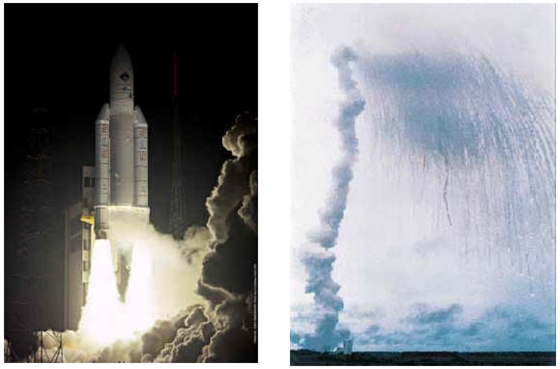
- `String` text processing.
- `double, int` mathematical calculation.
- `boolean` decision making.

In Java, you must:

- Declare type of values.
- Convert between types when necessary.

Why do we need types?

- Type conversion must be done at some level.
- Compiler can help do it correctly.
- Ex 1: in 1996, Ariane 5 rocket exploded after takeoff because of bad type conversion.
- Ex 2: `i = 0` in Matlab redefines $\sqrt{-1}$.



example of bad type conversion