AG2411 – GIS Architecture and Algorithms
Takeshi Shirabe
Geoinformatics, KTH
2022-10-31

Exercise 0a: Introduction to Programming in Java

This exercise is designed for those who have no or very little experience with computer programming and those who have some experience with programming but in a language other than Java. If you are already familiar to Java programming, you may skip the exercise.

We have chosen Java as the programming language of this course for two reasons. First, Java is one of the most well-designed object-oriented languages. Object orientation is a paradigm that is employed by many modern programming languages. The other reason is a more practical one: our teaching staff can help you learn programming in Java. However, if you do not feel comfortable with this choice, you may use another GENERAL-PURPOSE COMPLIED language such as C++. **But before doing so, please contact the instructor.**

In the first half of the exercise, you will learn how to **write**, **compile**, and **run** a computer program in Java using a sample program of the most celebrated kind, that is, the “Hello World!” program. It has only one function, which is to print out "Hello World!" on a display.

1. First Java code

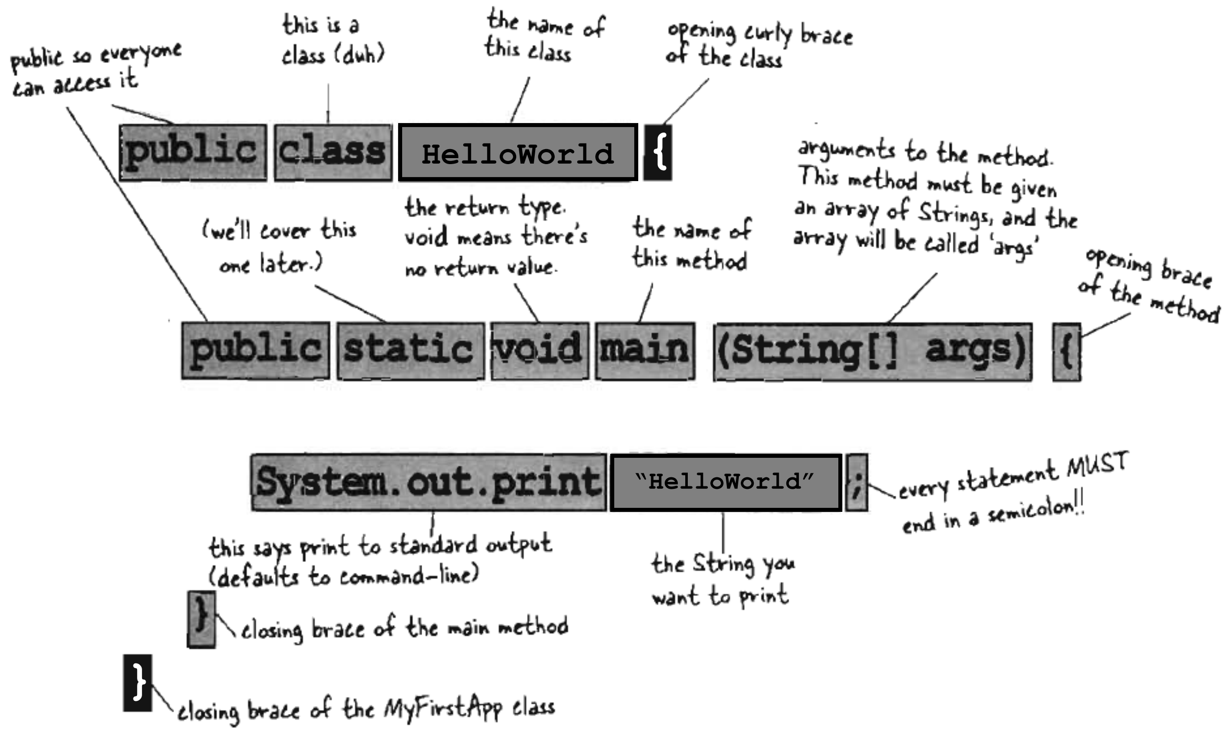
So the “Hello World!” program is going to be your first computer program. But wait. Do you know what a **computer program** is? It is a sequence of statements that instructs a computer how to do a specific task. It is like a navigation instruction “Go straight and turn right at the first intersection. Turn left at the next intersection. Stay on the current street and you will reach the destination.” But a computer program needs to be much better structured to avoid any ambiguity.

The best way to communicate a program is to write it down. In doing this, you have to follow the format and syntax of a programming language. A program written in a programming language is called **source code**. You have chosen Java as your programming language. So you will see lots of Java source code in this course. The first one represents the “Hello World” program as follows.

```
public class HelloWorld {
   public static void main(String[] args) {
         System.out.print("Hello World!");
   }
}
```

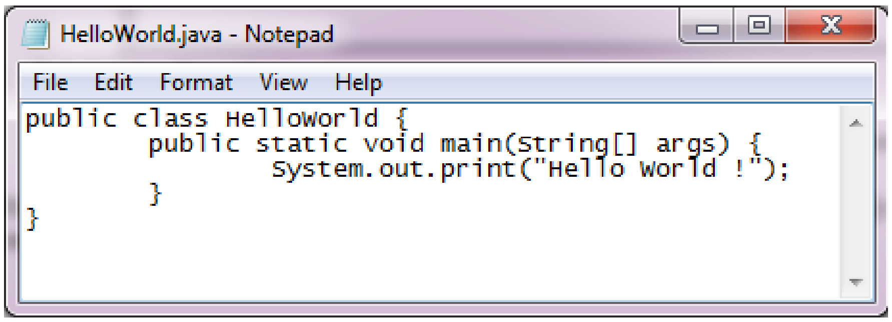
1. Write Java source code

If you want to make a jump start now, here is a quick analysis of the above code. Of course, you can skip this.


public so everyone can access it
this is a class (duh)
the name of this class
opening curly brace of the class
public class HelloWorld {
arguments to the method. This method must be given an array of Strings, and the array will be called 'args'
(we'll cover this one later.)
the return type. void means there's no return value.
the name of this method
opening brace of the method
public static void main (String[] args) {
System.out.print "HelloWorld" ;
every statement MUST end in a semicolon!!
this says print to standard output (defaults to command-line)
the String you want to print
}
closing brace of the main method
}
closing brace of the MyFirstApp class

Let's write this code on a text editor, e.g., Microsoft's Notepad.

1. To open Nodepad, select
   Start > Windows Accessories > Notepad
2. Write the code in a blank Notepad file. It should be look like this.



```
public class HelloWorld {
        public static void main(String[] args) {
                System.out.print("Hello World !");
        }
}
```

3. To save the file, select
   File > Save As…
   Then save it "***HelloWorld.java***" in a folder of your choice (most likely one in your Z drive).

Note that a text (or ASCII) file that stores source code is referred to as a (Java) **source file**, whose name must end with ".java".

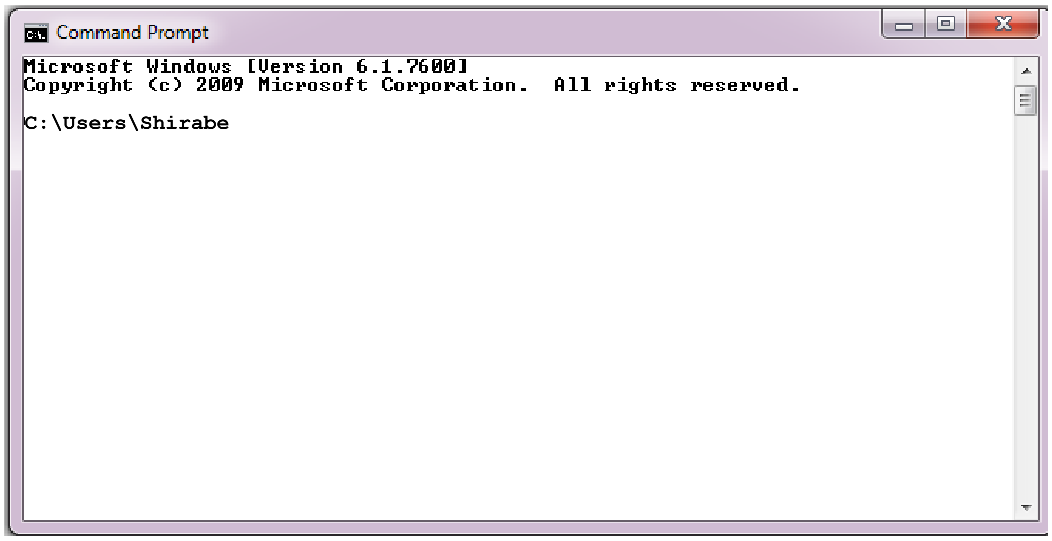
2. Compile Java source code

Although you do not understand what exactly your first java program says, you may have some idea what it means. It is because the syntax of Java (and other computer languages you may have heard of) is designed for humans to write and read code. Such a language is called a high-level language.

However, source code makes no sense to computers. Thus, source code has to be translated into a form that computers understand in order to run their corresponding programs. The process of this translation is called **compilation**, which is triggered by the command **javac**. Code written in the computer-friendly form is called (java) **byte code**, which is stored in a file called a "**class file**." The name of a class file ends with "**.class**".

Let's compile HelloWorld.java.

1. Open the Command Prompt application (in Windows) by selecting
   Start > Windows System > Command Prompt



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Shirabe
```

2. Go to your working directory (i.e. the directory in which you store HelloWorld.java file). Assuming that it is **Z:\AG2411\Lab0**, type

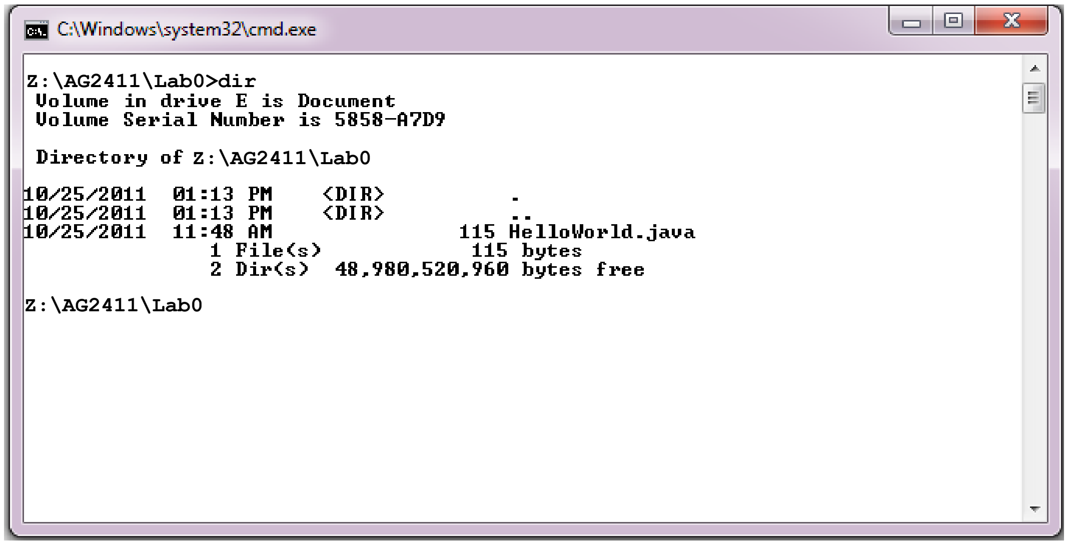
   Z:

   and press the Enter key, and the type the following line and press the Enter key:

   cd AG2411\Lab0

3. In this directory there should be still only one file, HelloWorld.java. To see this, type the following line and press the Enter key:

   dir



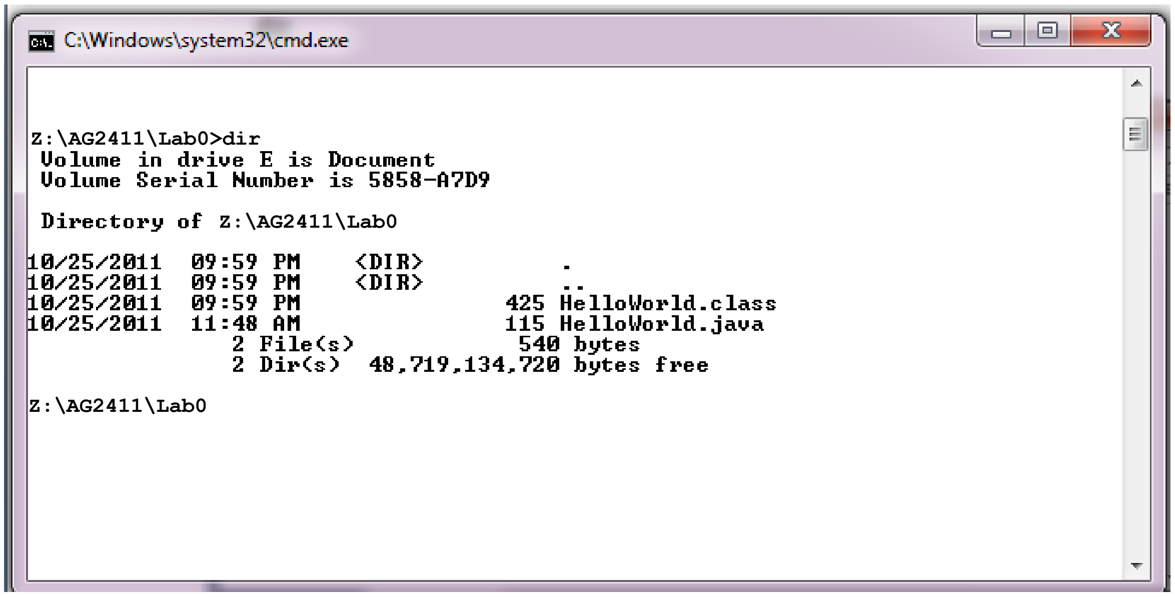
```
Z:\AG2411\Lab0>dir
 Volume in drive E is Document
 Volume Serial Number is 5858-A7D9

 Directory of Z:\AG2411\Lab0

10/25/2011  01:13 PM    <DIR>          .
10/25/2011  01:13 PM    <DIR>          ..
10/25/2011  11:48 AM               115 HelloWorld.java
               1 File(s)            115 bytes
               2 Dir(s)  48,980,520,960 bytes free

Z:\AG2411\Lab0
```

4. To compile HelloWorld.java, type the following line and press the Enter key:

   javac HelloWorld.java

5. The file called **HelloWorld.class** has been automatically created in the working directory. To see this, type the following line and hit the Enter key:

   dir



```
Z:\AG2411\Lab0>dir
 Volume in drive E is Document
 Volume Serial Number is 5858-A7D9

 Directory of Z:\AG2411\Lab0

10/25/2011  09:59 PM    <DIR>          .
10/25/2011  09:59 PM    <DIR>          ..
10/25/2011  09:59 PM               425 HelloWorld.class
10/25/2011  11:48 AM               115 HelloWorld.java
               2 File(s)            540 bytes
               2 Dir(s)  48,719,134,720 bytes free

Z:\AG2411\Lab0
```

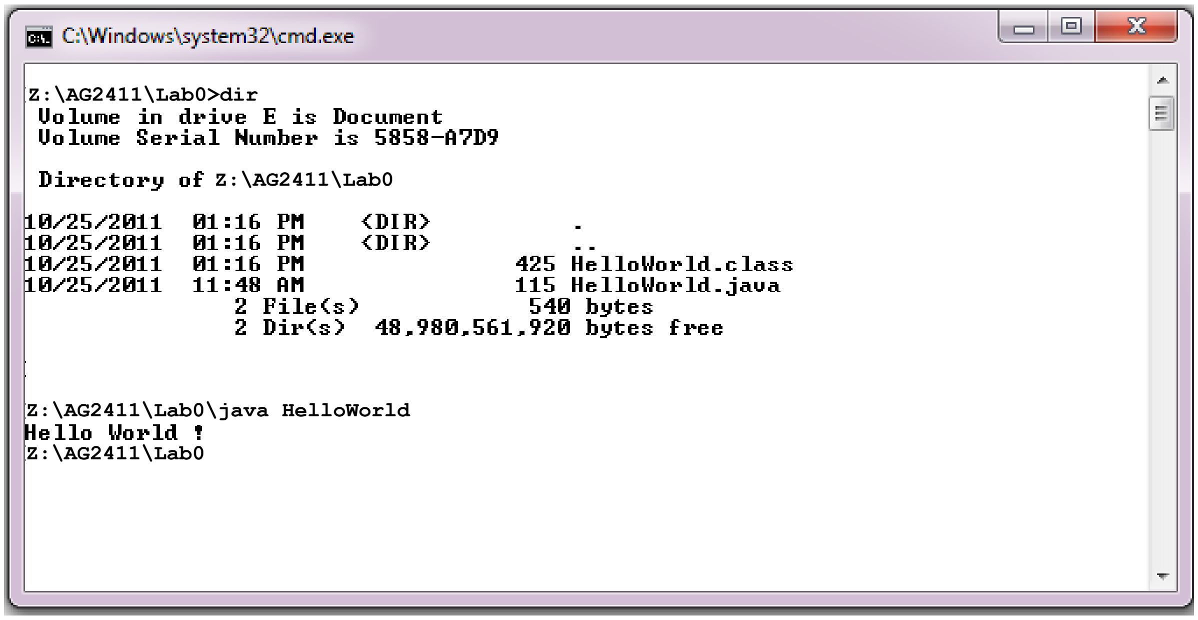
2. Run a Java class file

As mentioned already, the computer understands java class files, which means that it can run them. The command to run a java class file is **java**.

1. To run the HelloWorld.class file, type the following line and press the Enter key:

   java HelloWorld

   Note that there is no ".class" after "HelloWorld."



```
Z:\AG2411\Lab0>dir
 Volume in drive E is Document
 Volume Serial Number is 5858-A7D9

 Directory of Z:\AG2411\Lab0

10/25/2011  01:16 PM    <DIR>          .
10/25/2011  01:16 PM    <DIR>          ..
10/25/2011  01:16 PM               425 HelloWorld.class
10/25/2011  11:48 AM               115 HelloWorld.java
               2 File(s)            540 bytes
               2 Dir(s)  48,980,561,920 bytes free


Z:\AG2411\Lab0\java HelloWorld
Hello World !
Z:\AG2411\Lab0
```

2. Key Concepts and Keywords

In the previous section, all you did was to blindly type lines of characters comprising some code without knowing what they really meant. This section will introduce you to some concepts and terms essential for computer programming (particularly in Java), which will prepare you to do a lot more complex tasks.

1. Variables

Almost all computer programs (including those used in GIS) process data, and data are often stored somewhere before being processed. In programming, a variable refers to a location in a computer memory where a piece of data is stored.

There are a variety of types of data that Java can handle, which include:
- **boolean**: true or false
- **int**: integer
- **String**: string of characters

The syntax for declaring a variable is:

data type name variable name;

For example,
```
boolean b;
int i;
String s;
```

Once variables are declared, you can assign values to them. The syntax for assigning a value to a variable is:

variable name = value;

For example,
```
b = true;
i = 0;
s = "Hello World!";
```

Let’s use these variables and modify the HelloWorld code.

```
public class HelloWorld {
   public static void main(String[] args) {
         // Declare variables
         boolean b;
         int i;
         String s;
         // Assign values to variables
         b = true;
         i = 0;
         s = “Hello World!”;
         // Print on the screen
         System.out.println(b);
         System.out.println(i);
         System.out.println(s);
   }
}
```

Write, compile, and run the above code, and see what this program does.

Each variable used in the above code contains only one value. If you want a single variable to store two or more values, however, you can declare it as **array** (of variables, each of which stores one value). The syntax for declaring an array variable is:

<u>data type</u>[] <u>variable name</u>;

The syntax for assigning a specified number of *default* values to an array variable is:

<u>variable name</u> = new <u>data type</u>[length];

The syntax for assigning a value to the element at a specified position (or index) of an array variable is:

<u>variable name</u>[index] = <u>value</u>;

Note that the first index is 0.

Here is how to declare and assign arrays of booleans, integers, and strings.

```
String[] s;
s = new String[4];
s[0] = “Hello”;
s[1] = “ ”;
s[2] = “World”;
s[3] = “!”;
```

The syntax for getting access to the value stored in the element at a specified position (or index) of an array variable is:

<u>variable name</u>[index];

The following program uses an array variable and does exactly the same thing as the original "Hello World!" program.

```
public class HelloWorld {
   public static void main(String[] args) {
         String[] s;
         s = new String[4];
         s[0] = "Hello"; // the first index is 0!
         s[1] = " ";
         s[2] = "World";
         s[3] = "!";
         System.out.print(s[0] + s[1] + s[2]+ s[3]);
   }
}
```

2. Arguments

What data your program needs to process may depend on the context. Sometimes you may want to say hello to the world but other times you may want to say goodbye to it. To respond to the need for more dynamic data processing, you can use **arguments**. Arguments are an array of values that are passed to a program from the outside of the program (often by the user of the program).

The current version of the "Hello World!" program does not use arguments. But it does have a variable to store arguments. It is called **args**. Note that this variable takes the form of an array of Strings.

The following version of the "Hello World!" program takes three arguments as input, and prints them on the screen in return.

```
public class HelloWorld {
   public static void main(String[] args) {
         String s1;
         String s2;
         String s3;
         s1 = args[0];    // the first index is 0!
         s2 = args[1];
         s3 = args[2];
         System.out.print (s1 + " " + s2 + " " +s3);
   }
}
```

Write and compile the above program, and run it with three arguments, "Goodbye", "World", and "!", that is:

java HelloWorld Goodbye World !

You can certainly use a different set of arguments. Try!

3. Control Statements

You are already familiar to statements expressed in Java. For example,

```
      int[] i;
      s1 = args[0];
      System.out.print (s1+s2+s3);
```

are all Java statements.

In this section, you will learn special types of statements. They are special in that they have special purposes and follow special syntactic rules.

<if-statements>

An if-statement is a statement used to execute a specified piece of code if a specified condition is true. The syntax for expressing an if-statement is:

```
      if (condition) {
            do something
      }
```

Here is an example of an if-statement.

```
   public class HelloWorld {
      public static void main(String[] args) {
            if (args.length >= 3) {
                  String s1;
                  String s2;
                  String s3;
                  s1 = args[0];    // the first index is 0!
                  s2 = args[1];
                  s3 = args[2];
                  System.out.println(s1+s2+s3);
            }
      }
   }
```

An if-statement may be followed by an else-statement. The syntax for expressing this variation of if-statement is:

```
if (condition) {
      do something
}
else {
      do something
}
```

Here is an example of an if-statement followed by an else-statement.

```
public class HelloWorld {
   public static void main(String[] args) {
         if (args.length >= 3) {
               String s1;
               String s2;
               String s3;
               s1 = args[0];    // the first index is 0!
               s2 = args[1];
               s3 = args[2];
               System.out.println(s1+s2+s3);
         }
         else {
               System.out.print("Too few arguments!");
               System.exit(0);
         }
   }
}
```

An if-statement may be followed by one or more else-if-statements and an else-statement. The syntax for expressing this variation of if-statement is:

```
if (condition) {
      do something
}
else if (condition) {
      do something
}
.
.
.
else if (condition) {
      do something
}
else (condition) {
      do something
}
```

Here is an example of an if-statement followed by an else-if-statement and an else-statement.

```
public class HelloWorld {
   public static void main(String[] args) {
         if (args.length < 3) {
               System.out.print("Too few arguments!");
               System.exit(0);
         }
         else if (args.length == 3) {
               String s1;
               String s2;
               String s3;
               s1 = args[0];    // the first index is 0!
               s2 = args[1];
               s3 = args[2];
               System.out.println(s1+s2+s3);
         }
         else{
               System.out.print("Too many arguments!");
               System.exit(0);
         }
   }
}
```

<for-loop statements>
A for-loop statement is a statement used to repeat a piece of code for a specified number of times. The syntax for expressing a for-loop statement is:

```
for(int i=0; i<some_integer; i++){
      do something
}
```

Note that there are a variety of ways to change the value of the variable in the first line (i.e. `i`). You will learn more about this later.

Here is an example of a for-loop.

```
public class HelloWorld {
      public static void main(String[] args) {
            String s;
            s = args[0];
            for(int i=0; i<9; i++){
                  System.out.println(i + " " + s);
            }
      }
}
```

<while-loop statements>
A while-loop statement is a statement used to repeat a piece of code while a specified condition is true. The syntax for expressing a while-loop statement is:

```
while(condition){
      do something
}
```

Here is an example of a **while-loop** statement, which actually generates the same output as the for-loop statement illustrated above does, but differently.

```
public class HelloWorld {
      public static void main(String[] args) {
            String s;
            int i;
            s = args[0];
            i = 0;
            while(i<9){
                  System.out.println(i + " " + s);
                  i = i + 1;
            }
      }
}
```

There may be cases where you want to get out the while-loop even while the condition is true when some other condition is true. The syntax for expressing such a while-loop is:

```
while(condition1){
      if (condition2) {
            break;
      }
      do something
}
```

Here is an example of a **while-loop** statement, which stops printing and counting when i is found to be 5.

```
public class HelloWorld {
      public static void main(String[] args) {
            String s;
            int i;
            s = args[0];
            i = 0;
            while(i<9){
                  if (i == 5) {
                        break;
                  }
                  System.out.println(i + " " + s);
                  i = i + 1;
            }
      }
}
```

In other cases, you don't want to get out the while-loop while the condition is true but want to skip the current iteration when some other condition is true. The syntax for expressing such a while-loop is:

```
while(condition1){
      if (condition2) {
            continue;
      }
      do something
}
```

Here is an example of a **while-loop** statement, which does not print (but continues counting) when i is found to be 5 and resumes printing (and counting) when i is found to be 6.

```
public class HelloWorld {
      public static void main(String[] args) {
            String s;
            int i;
            s = args[0];
            i = 0;
            while(i<9){
                  if (i == 5) {
                        i = i + 1; // still need to count!
                        continue;
                  }
                  System.out.println(i + " " + s);
                  i = i + 1;
            }
      }
}
```

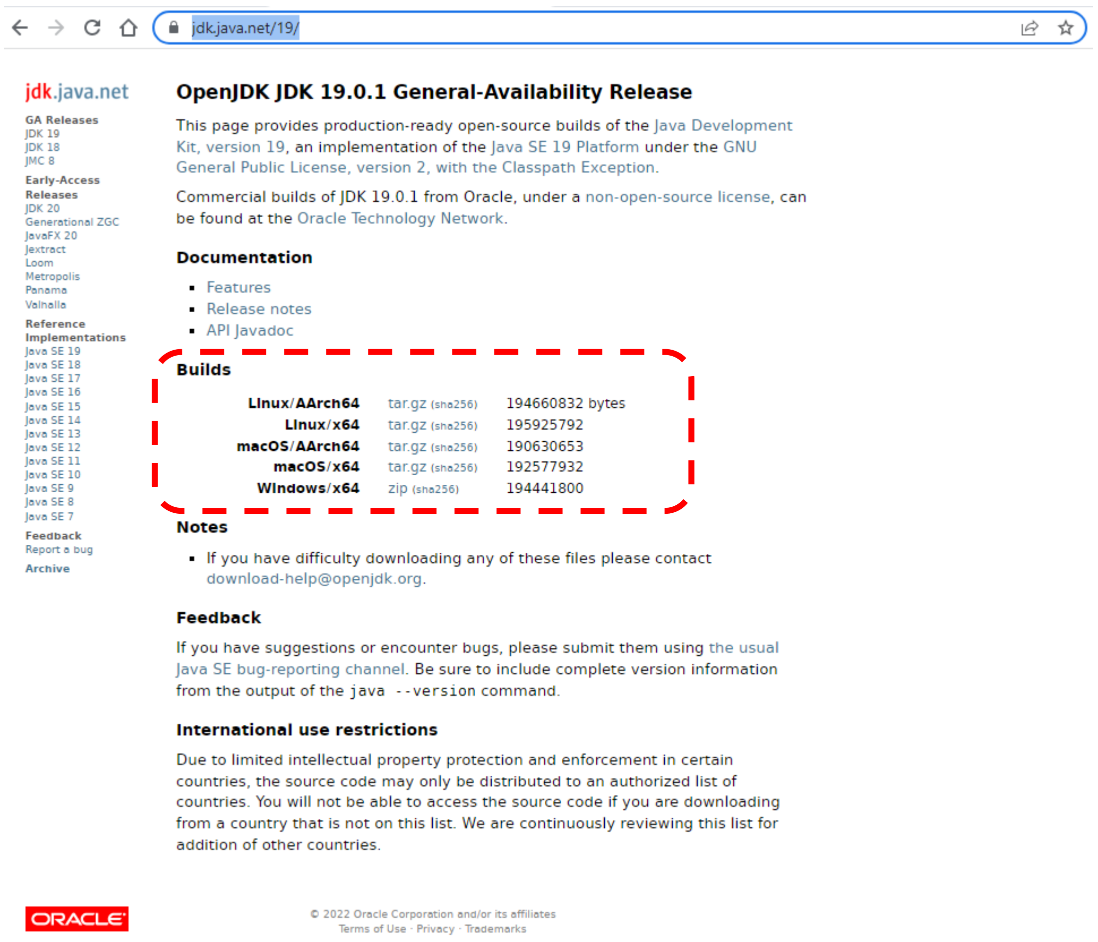
Reference:

There is a good java tutorial published by Oracle. You can get access to it or download its complete package (as a zipped archive) at http://download.oracle.com/javase/tutorial/index.html

Appendix A: Installing Java to your own computer (OpenJDK)

To compile java source code (stored in .java files) and run java bytecode (stored in .class files), you need a software application consisting of some important components including a java complier. Oracle offers one such application called "Open Java Development Kit (OpenJDK)." Follow the instruction below to install it.

1. Find an installation program of OpenJDK at: https://jdk.java.net/19/ (or a version of your choice)
2. Choose an operating system (Windows, Mac or Linux), and download. See below. If you are not certain about your OS, ask the teaching assistant.



OpenJDK JDK 19.0.1 General-Availability Release

This page provides production-ready open-source builds of the Java Development Kit, version 19, an implementation of the Java SE 19 Platform under the GNU General Public License, version 2, with the Classpath Exception.

Commercial builds of JDK 19.0.1 from Oracle, under a non-open-source license, can be found at the Oracle Technology Network.

Documentation

- Features
- Release notes
- API Javadoc

Builds

Linux/AArch64	tar.gz (sha256)	194660832 bytes
Linux/x64	tar.gz (sha256)	195925792
macOS/AArch64	tar.gz (sha256)	190630653
macOS/x64	tar.gz (sha256)	192577932
Windows/x64	zip (sha256)	194441800

Notes

- If you have difficulty downloading any of these files please contact download-help@openjdk.org.

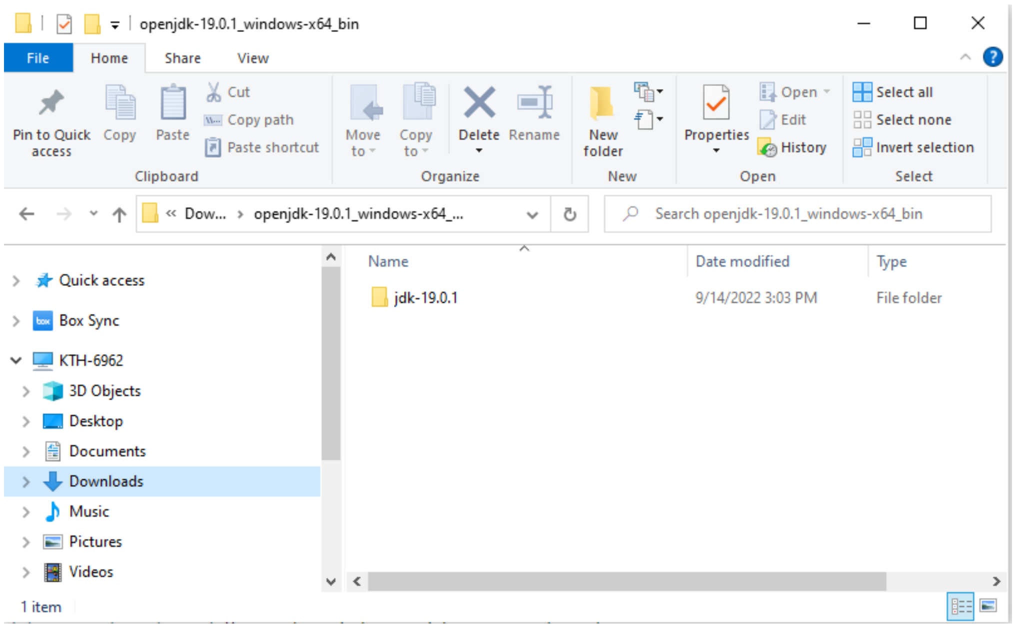
Feedback

If you have suggestions or encounter bugs, please submit them using the usual Java SE bug-reporting channel. Be sure to include complete version information from the output of the `java --version` command.

International use restrictions

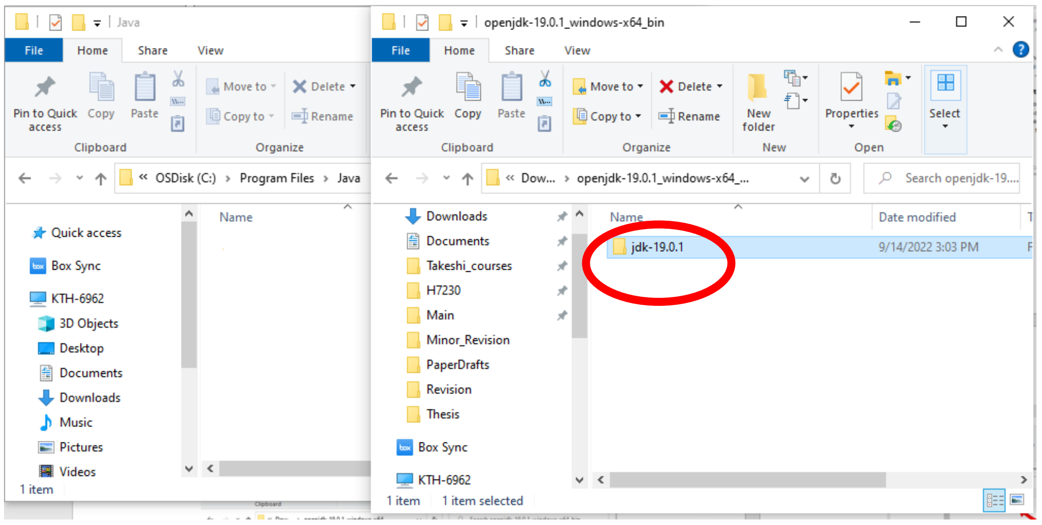
Due to limited intellectual property protection and enforcement in certain countries, the source code may only be distributed to an authorized list of countries. You will not be able to access the source code if you are downloading from a country that is not on this list. We are continuously reviewing this list for addition of other countries.

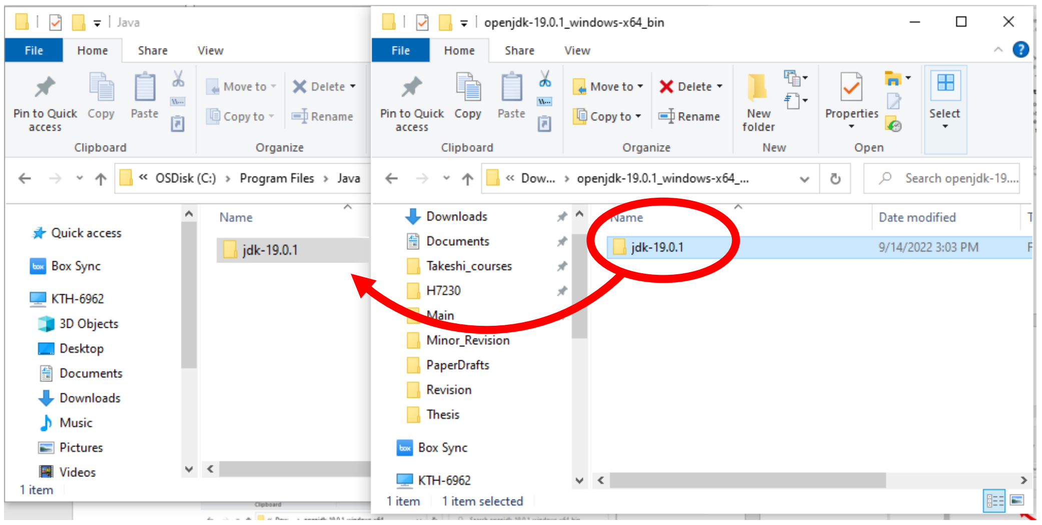
3. Extract the downloaded zip file. You will get a folder, which should look like this.



Name	Date modified	Type
jdk-19.0.1	9/14/2022 3:03 PM	File folder

4. Copy this folder into the Java folder in the Program Files on your computer. See below.



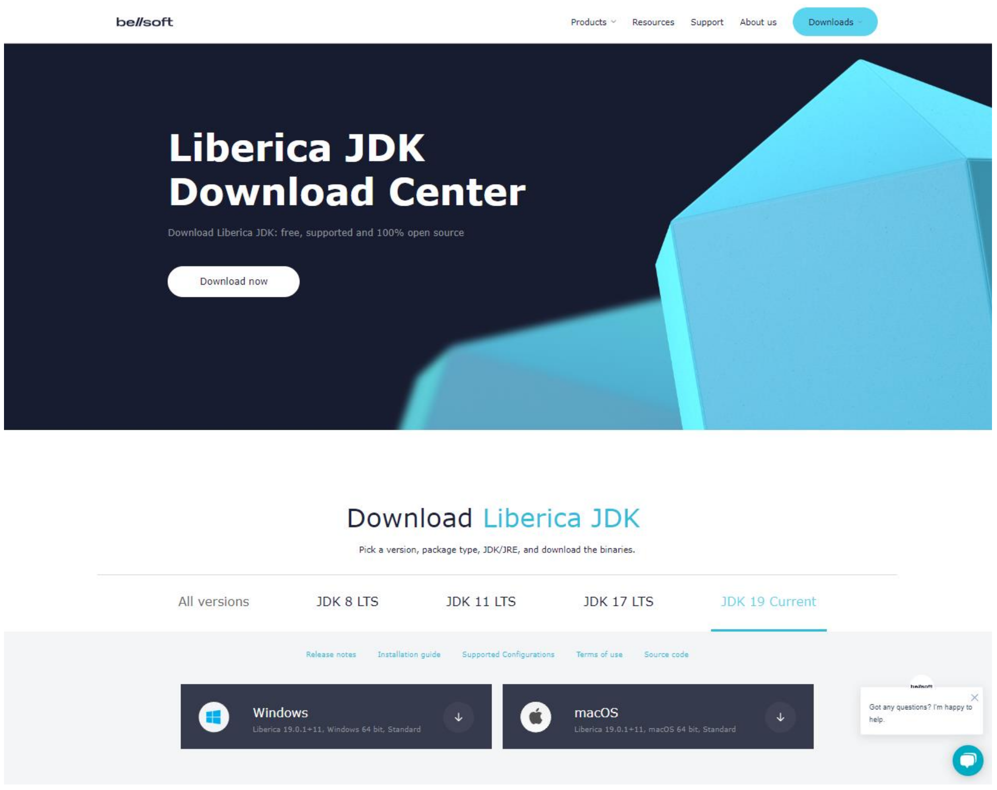


Java
File
Home
Share
View
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
Clipboard
Organize
« OSDisk (C:) › Program Files › Java
Quick access
Box Sync
KTH-6962
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Name
jdk-19.0.1
1 item
openjdk-19.0.1_windows-x64_bin
New folder
Properties
Select
New
Open
« Dow... › openjdk-19.0.1_windows-x64_...
Search openjdk-19....
Downloads
Documents
Takeshi_courses
H7230
Main
Minor_Revision
PaperDrafts
Revision
Thesis
Box Sync
KTH-6962
Name
Date modified
jdk-19.0.1
9/14/2022 3:03 PM
1 item
1 item selected

Appendix B: Installing Java to your own computer (Liberica JDK)

There are many alternatives to Oracle's OpenJDK. "Liberica JDK" is one of them developed by BellSoft. Follow the instruction below to install it.

1. Find an installation program of Liberica JDK at: https://bell-sw.com/pages/downloads/#downloads (or a version of your choice)
2. Choose an operating system (Windows, Mac or Linux), and download. See below. If you are not certain about your OS, ask the teaching assistant.



Liberica JDK Download Center

Download Liberica JDK: free, supported and 100% open source

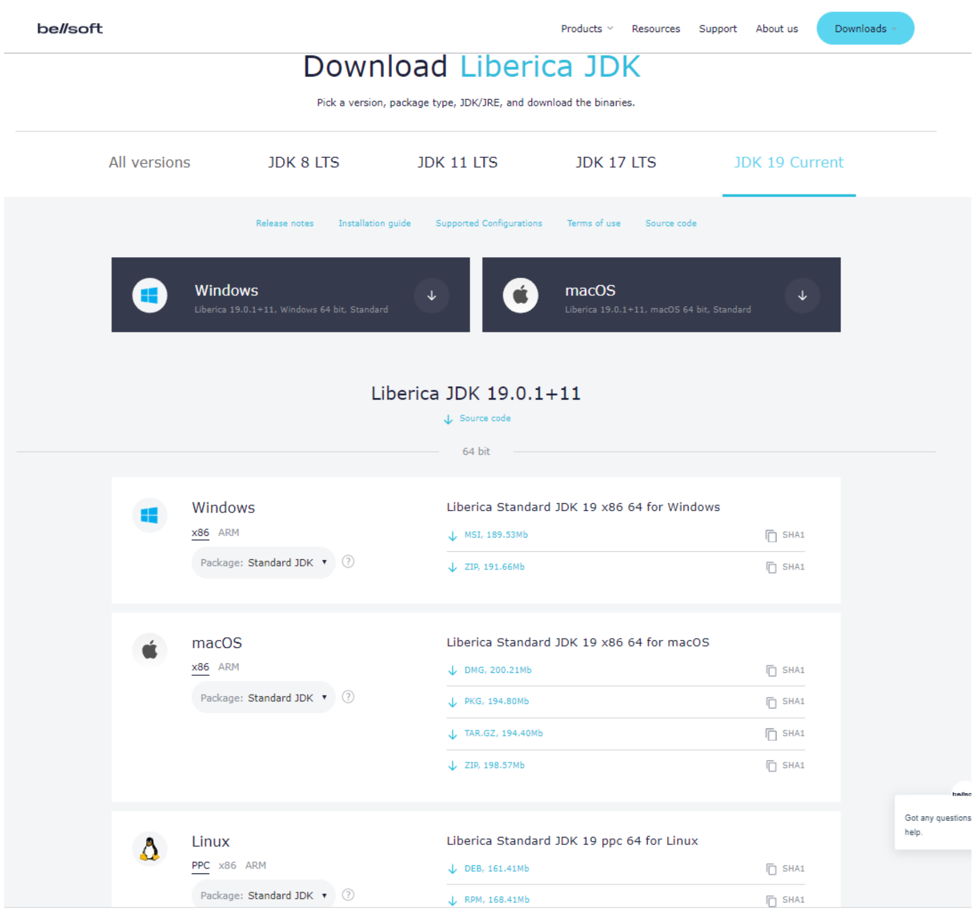
Download Liberica JDK

Pick a version, package type, JDK/JRE, and download the binaries.

Liberica 19.0.1+11, Windows 64 bit, Standard

Liberica 19.0.1+11, macOS 64 bit, Standard

Got any questions? I'm happy to help.



Download Liberica JDK

Pick a version, package type, JDK/JRE, and download the binaries.

Windows — Liberica 19.0.1+11, Windows 64 bit, Standard

macOS — Liberica 19.0.1+11, macOS 64 bit, Standard

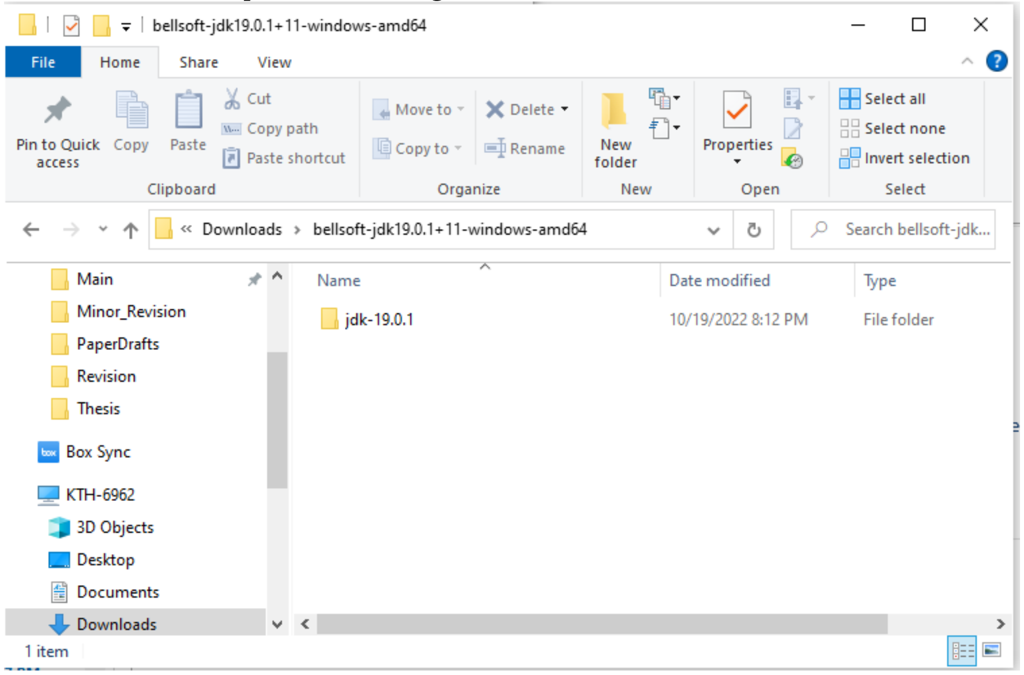
Liberica JDK 19.0.1+11

Source code

64 bit

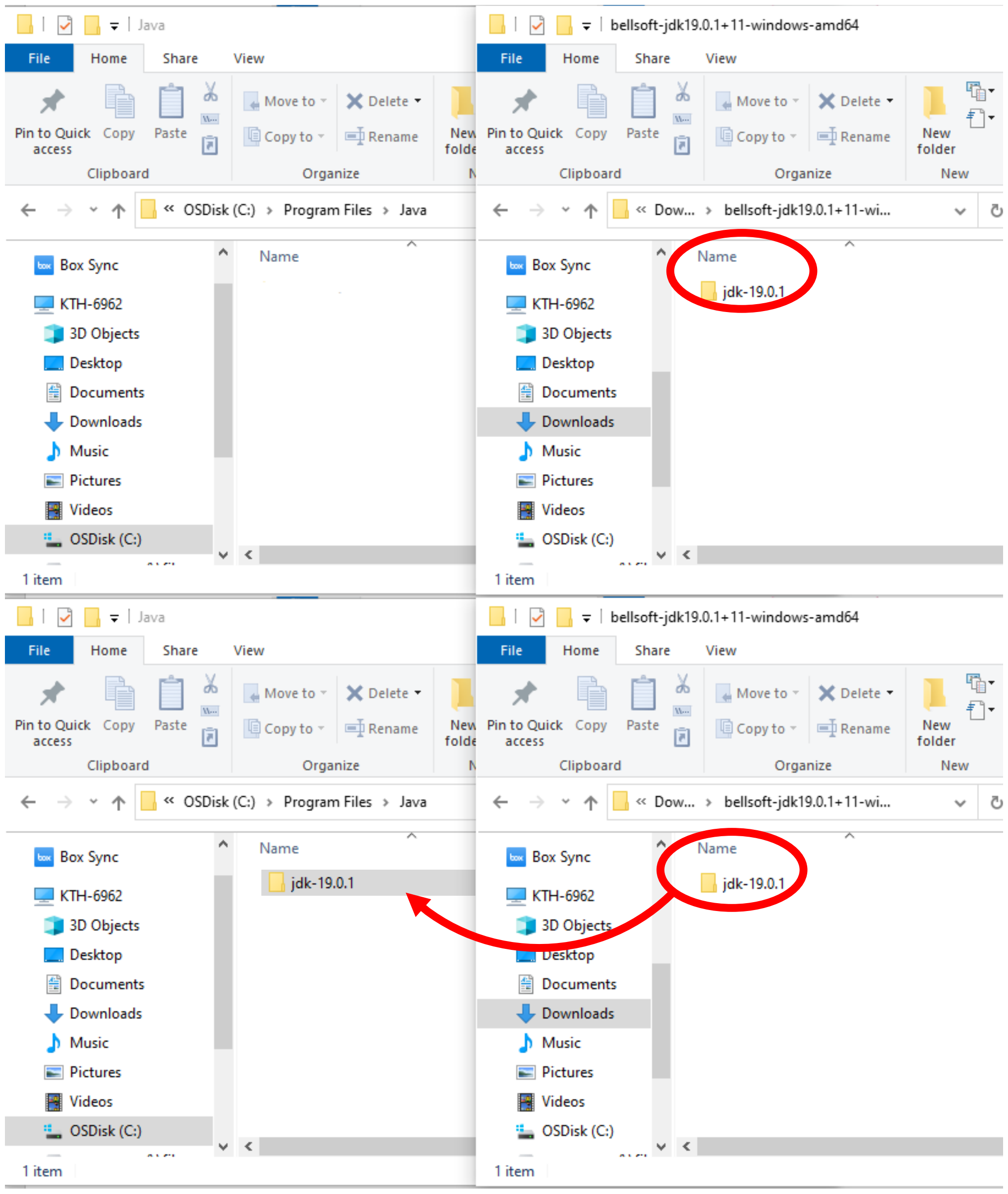
Platform	Download	
Windows (x86, ARM; Package: Standard JDK) — Liberica Standard JDK 19 x86 64 for Windows	MSI, 189.53Mb	SHA1
	ZIP, 191.66Mb	SHA1
macOS (x86, ARM; Package: Standard JDK) — Liberica Standard JDK 19 x86 64 for macOS	DMG, 200.21Mb	SHA1
	PKG, 194.80Mb	SHA1
	TAR.GZ, 194.40Mb	SHA1
	ZIP, 198.57Mb	SHA1
Linux (PPC, x86, ARM; Package: Standard JDK) — Liberica Standard JDK 19 ppc 64 for Linux	DEB, 161.41Mb	SHA1
	RPM, 168.41Mb	SHA1

3. Extract the downloaded zip file. You will get a folder, which should look like this.



Name	Date modified	Type
jdk-19.0.1	10/19/2022 8:12 PM	File folder

4. Copy this folder into the Java folder in the Program Files on your computer. See below.


Java
File
Home
Share
View
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
New folder
Clipboard
Organize
New
OSDisk (C:) › Program Files › Java
bellsoft-jdk19.0.1+11-windows-amd64
Dow... › bellsoft-jdk19.0.1+11-wi...
Name
jdk-19.0.1
Box Sync
KTH-6962
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OSDisk (C:)
1 item