Intro to Java (Page 9)

JUMP June 2019

Draft: 07/26/2019

Table of Contents

1. JAR files
2. Collections
3. Difference Between Arrays and Collections in Java

JAR Files

A **JAR** (**Java** ARchive) is a package **file** format typically used to aggregate many **Java** class **files** and associated metadata and resources (text, images, etc.) into one **file** for distribution. ... They are built on the ZIP format and typically have a .**jar file** extension.

JAR Files

Creating a Jar File

In Java, it is common to combine several classes in one .jar ("java archive") file. Library classes are stored that way. Larger projects use jar files. You can create your own jar file combining several classes, too.

jar files are created using the jar.exe utility program from the JDK.

Most IDEs like STS allows you to export your project as a JAR file as well.

Steps to Creating a Java Project, and a simple Class

1. Create a Java Project “JARProjectToExport”
2. Create a package name: “jardemopackage”
3. Create a class name: “Dice” inside the “jardemopackage”
4. Add the following content to the Dice Class

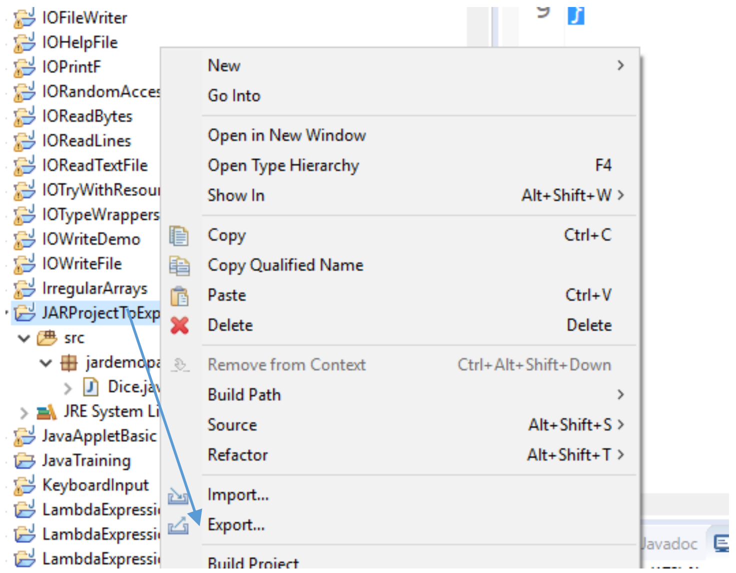
Note: If you don’t make the class and the method “public”. The package will be useless

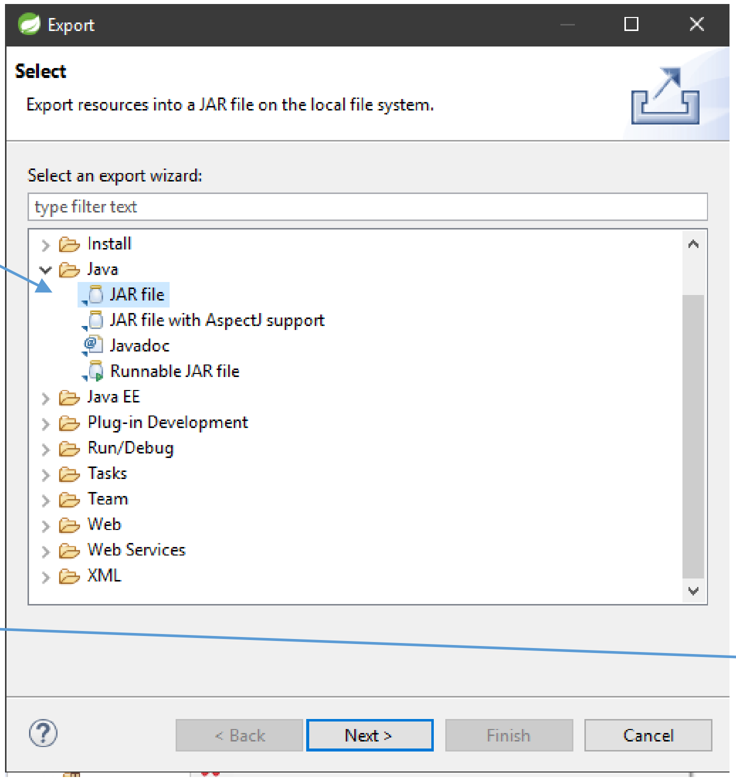
```
package jardemopackage;
public class Dice {
  public Dice() {

  }
  public int getDice() {
    return (int) (Math.random() * 6) + 1;
  }
}
```

JAR Files

Exporting the Project as a JAR file using STS

1. Use the context-menu on the project and select "Export"
2. Select "JAR file" from the JAVA folder
3. Click Next >
4. For our scenario, the options preselected are correct.
   Notice the "Select the resources to export", in theory, if you had dependencies between projects, you would be able to export them all as one single JAR file.
5. ***Pay attention to the "Select the export destination" because this is where the JAR file will be created and stored. Provide a folder/name which makes sense.***
6. Click "Finish"




Export

Select

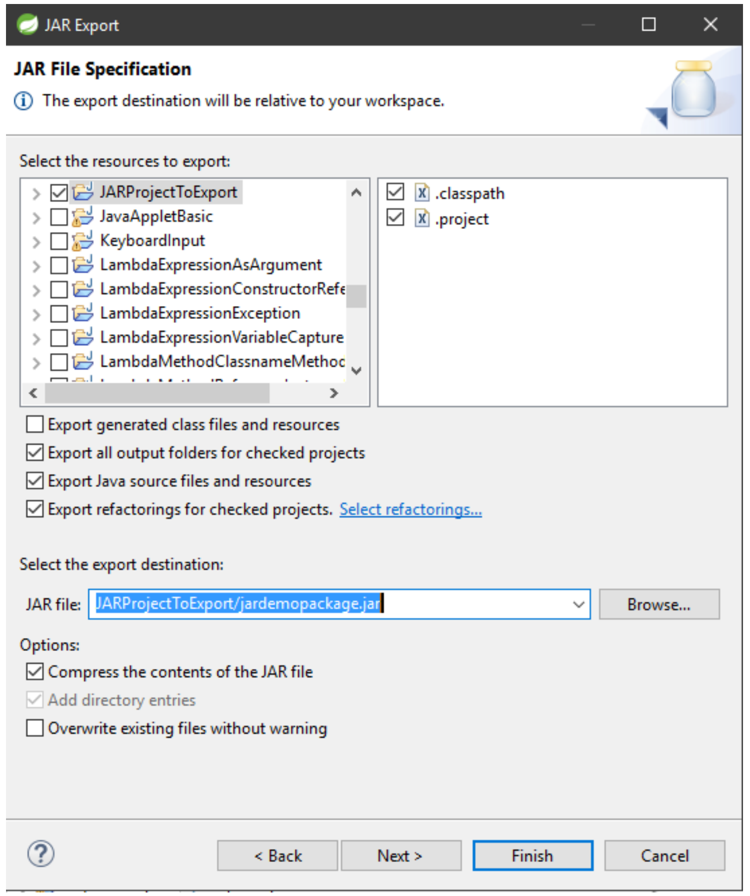
Export resources into a JAR file on the local file system.

Select an export wizard:

type filter text

- Install
- Java
  - JAR file
  - JAR file with AspectJ support
  - Javadoc
  - Runnable JAR file
- Java EE
- Plug-in Development
- Run/Debug
- Tasks
- Team
- Web
- Web Services
- XML

< Back | Next > | Finish | Cancel


JAR Export

JAR File Specification

The export destination will be relative to your workspace.

Select the resources to export:

- ☑ JARProjectToExport
- ☐ JavaAppletBasic
- ☐ KeyboardInput
- ☐ LambdaExpressionAsArgument
- ☐ LambdaExpressionConstructorRefe
- ☐ LambdaExpressionException
- ☐ LambdaExpressionVariableCapture
- ☐ LambdaMethodClassnameMethoc

- ☑ .classpath
- ☑ .project

☐ Export generated class files and resources
☑ Export all output folders for checked projects
☑ Export Java source files and resources
☑ Export refactorings for checked projects. Select refactorings...

Select the export destination:

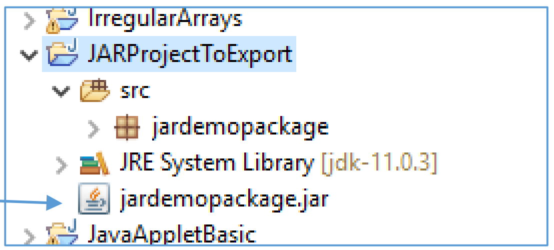
JAR file: JARProjectToExport/jardemopackage.jar | Browse...

Options:

☑ Compress the contents of the JAR file
☑ Add directory entries
☐ Overwrite existing files without warning

< Back | Next > | Finish | Cancel

Should have a JAR file



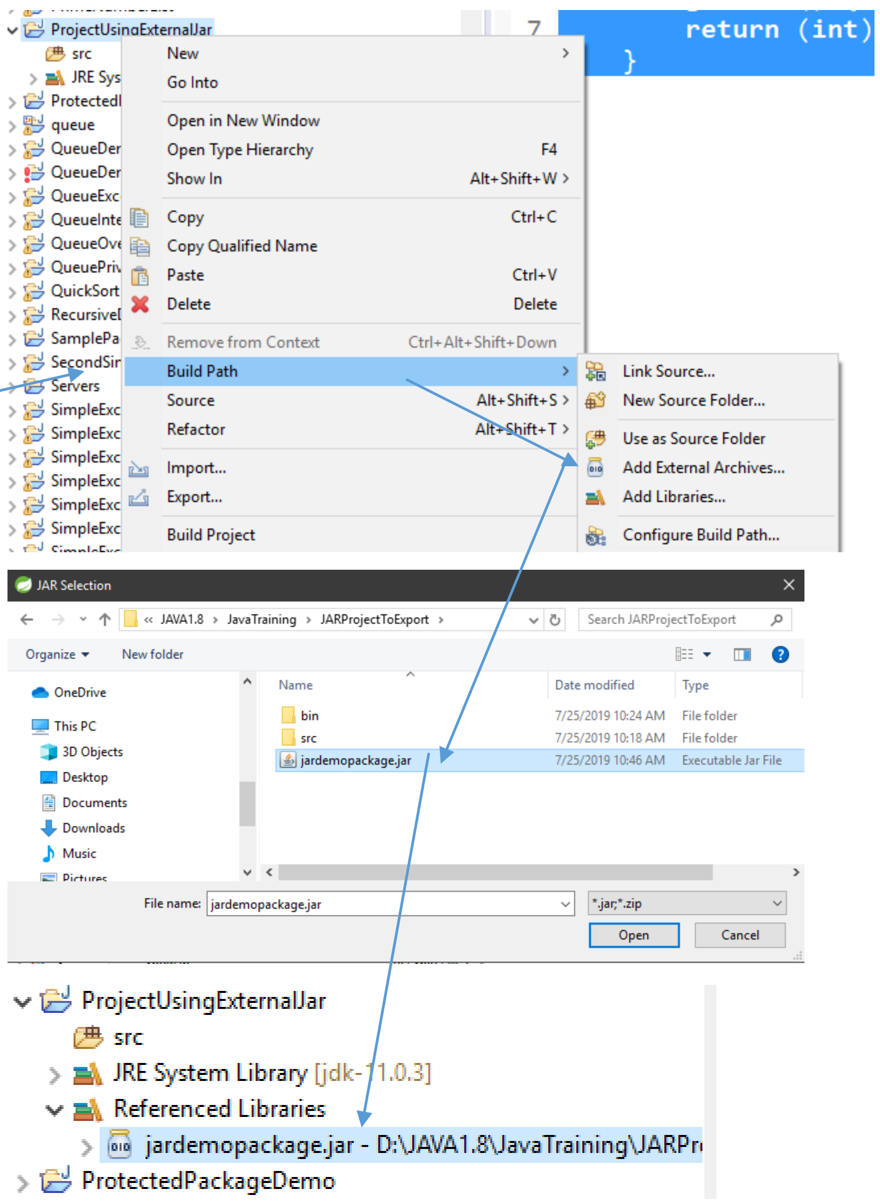
JAR Files

Using a JAR file into a project

1. Create a Java Project “ProjectUsingExternalJar”
2. Using the context-menu on the project, select Build Path > Add External Archives
3. Find the JAR file that was exported for this demo and click “Open”
4. Create a Main class in your src (it will create a default package)

Main.java

```
import jardemopackage.*;
import jardemopackage.Dice.*;
public class Main {
   public static void main(String[] args) {
      Dice d = new Dice();
      System.out.println("First Throw:  " + d.getDice());
      System.out.println("Second Throw:  " + d.getDice());
   }
}
```



Name	Date modified	Type
bin	7/25/2019 10:24 AM	File folder
src	7/25/2019 10:18 AM	File folder
jardemopackage.jar	7/25/2019 10:46 AM	Executable Jar File

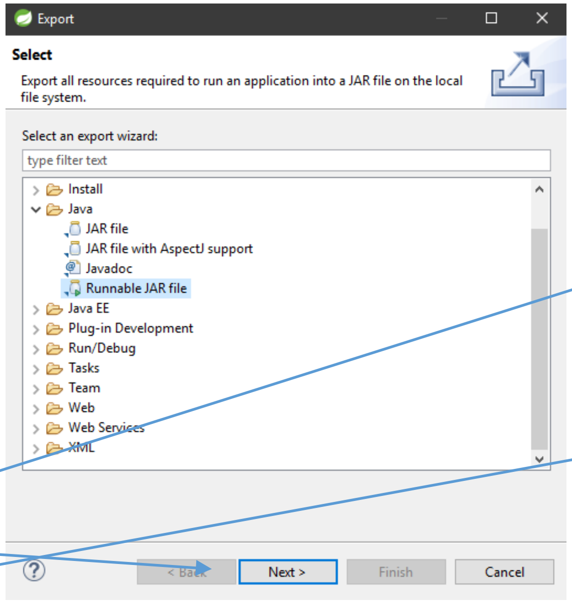
File name: jardemopackage.jar
.jar;.zip

JAR Files

Create a Self-Executing Jar

1. Using the Java Project “ProjectUsingExternalJar”
2. Use the context-menu on the project and select “Export”
3. Select “Runnable JAR file” from the JAVA folder
4. Click Next >
5. Ensure your “Launch Configuration” selection is based on your Main class for your project
6. Click the “Browse” button and ensure you type as a file name “selfexecutingjar” and ensure your path is in your ProjectUsingExternalJar folder
7. Click Save
8. This brings you back to the Runnable JAR File Export dialog box
9. Keep all other options as is and “Finish”
10. Click OK for this warning

Note: if you haven’t executed the file, your Launch Configuration will not provide the proper class/project info.


Export

Select

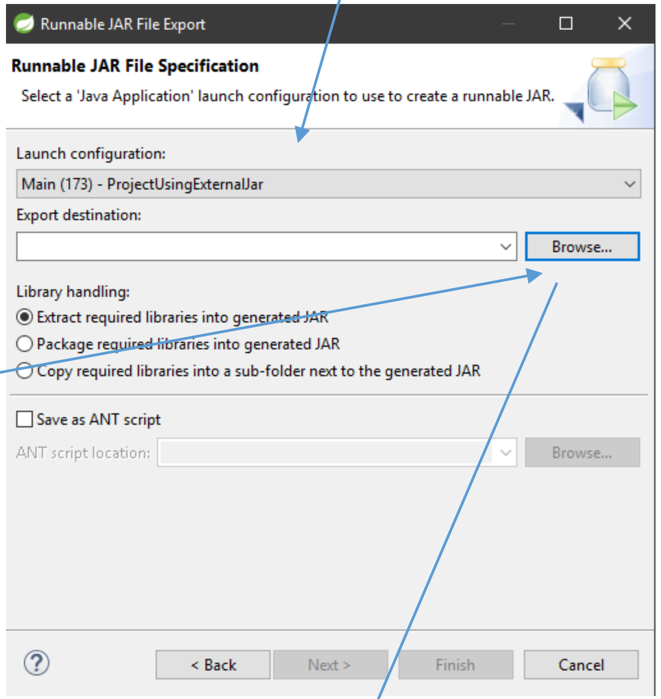
Export all resources required to run an application into a JAR file on the local file system.

Select an export wizard:

type filter text

- Install
- Java
  - JAR file
  - JAR file with AspectJ support
  - Javadoc
  - Runnable JAR file
- Java EE
- Plug-in Development
- Run/Debug
- Tasks
- Team
- Web
- Web Services
- XML

< Back | Next > | Finish | Cancel


Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: Main (173) - ProjectUsingExternalJar

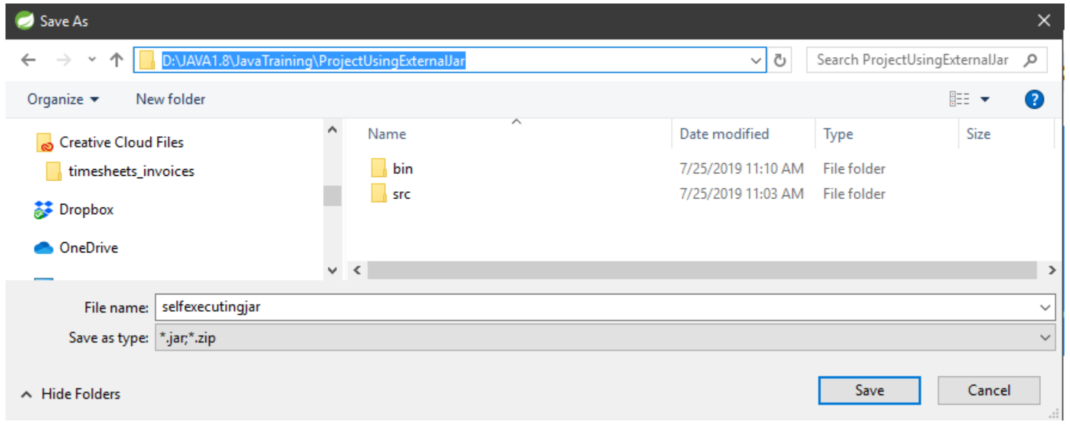
Export destination: [blank] Browse...

Library handling:
- [✓] Extract required libraries into generated JAR
- Package required libraries into generated JAR
- Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location: Browse...

< Back | Next > | Finish | Cancel


Save As

D:\JAVA1.8\JavaTraining\ProjectUsingExternalJar

Search ProjectUsingExternalJar

Organize ▾ New folder

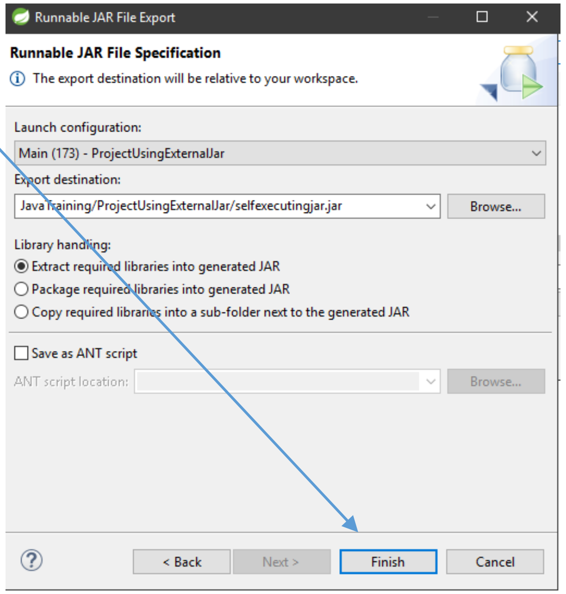
Creative Cloud Files
timesheets_invoices
Dropbox
OneDrive

Name	Date modified	Type	Size
bin	7/25/2019 11:10 AM	File folder	
src	7/25/2019 11:03 AM	File folder	

File name: selfexecutingjar

Save as type: *.jar;*.zip

Hide Folders | Save | Cancel


Runnable JAR File Export

Runnable JAR File Specification

ⓘ The export destination will be relative to your workspace.

Launch configuration: Main (173) - ProjectUsingExternalJar

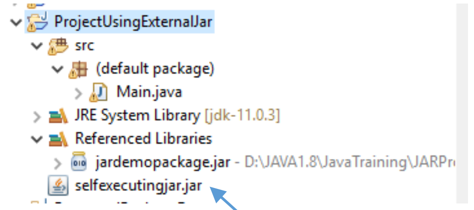
Export destination: JavaTraining/ProjectUsingExternalJar/selfexecutingjar.jar Browse...

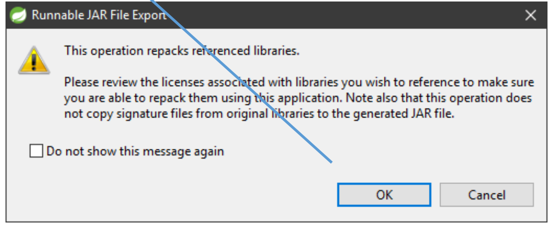
Library handling:
- [✓] Extract required libraries into generated JAR
- Package required libraries into generated JAR
- Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location: Browse...

< Back | Next > | Finish | Cancel


- ProjectUsingExternalJar
  - src
    - (default package)
      - Main.java
  - JRE System Library [jdk-11.0.3]
  - Referenced Libraries
    - jardemopackage.jar - D:\JAVA1.8\JavaTraining\JARPr
  - selfexecutingjar.jar


Runnable JAR File Export

This operation repacks referenced libraries.

Please review the licenses associated with libraries you wish to reference to make sure you are able to repack them using this application. Note also that this operation does not copy signature files from original libraries to the generated JAR file.

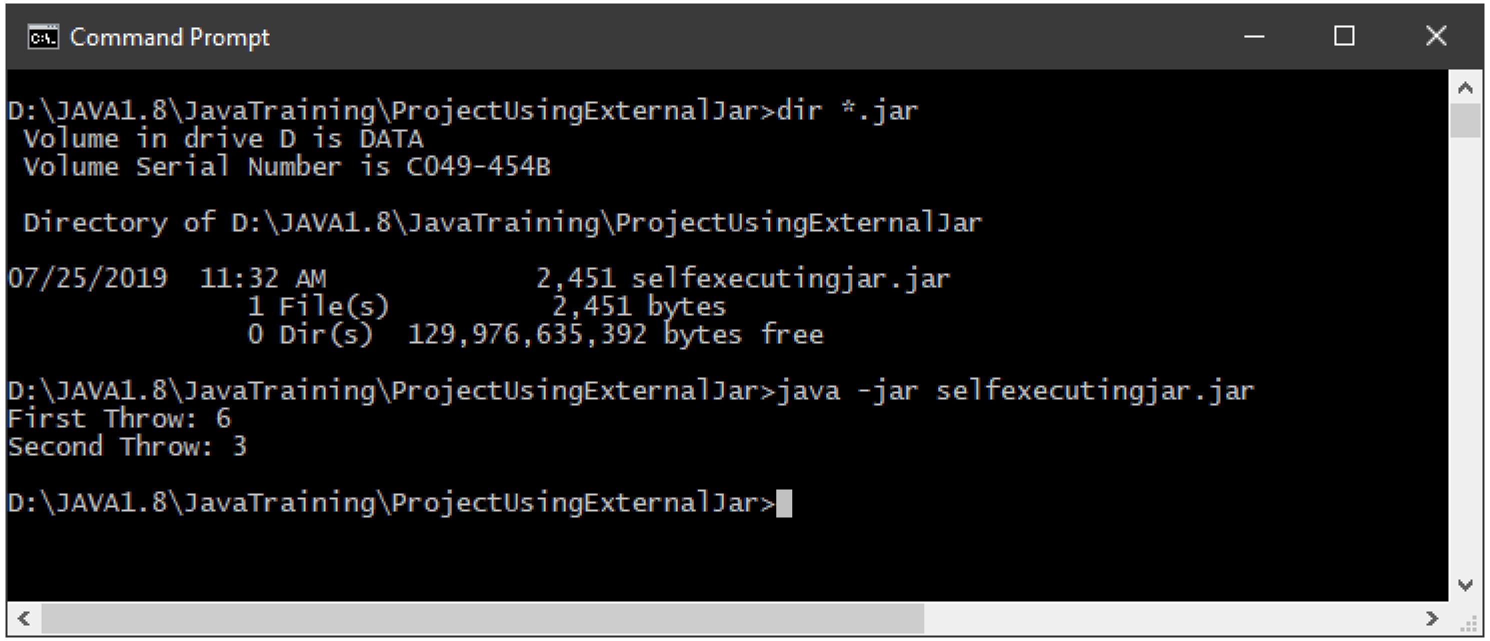
☐ Do not show this message again

OK | Cancel

JAR Files

Running a Self-Executing JAR file

1. Locate the directory of your jar file to execute using a cmd / terminal window.
2. **`>java –jar name_of_jar_file`**
   Note: on mac/linux, may need to use a "sudo" prefix



```
D:\JAVA1.8\JavaTraining\ProjectUsingExternalJar>dir *.jar
 Volume in drive D is DATA
 Volume Serial Number is C049-454B

 Directory of D:\JAVA1.8\JavaTraining\ProjectUsingExternalJar

07/25/2019  11:32 AM             2,451 selfexecutingjar.jar
               1 File(s)          2,451 bytes
               0 Dir(s)  129,976,635,392 bytes free

D:\JAVA1.8\JavaTraining\ProjectUsingExternalJar>java -jar selfexecutingjar.jar
First Throw: 6
Second Throw: 3

D:\JAVA1.8\JavaTraining\ProjectUsingExternalJar>
```

Collections

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

What is Collection in Java
A Collection represents a single unit of objects, i.e., a group.

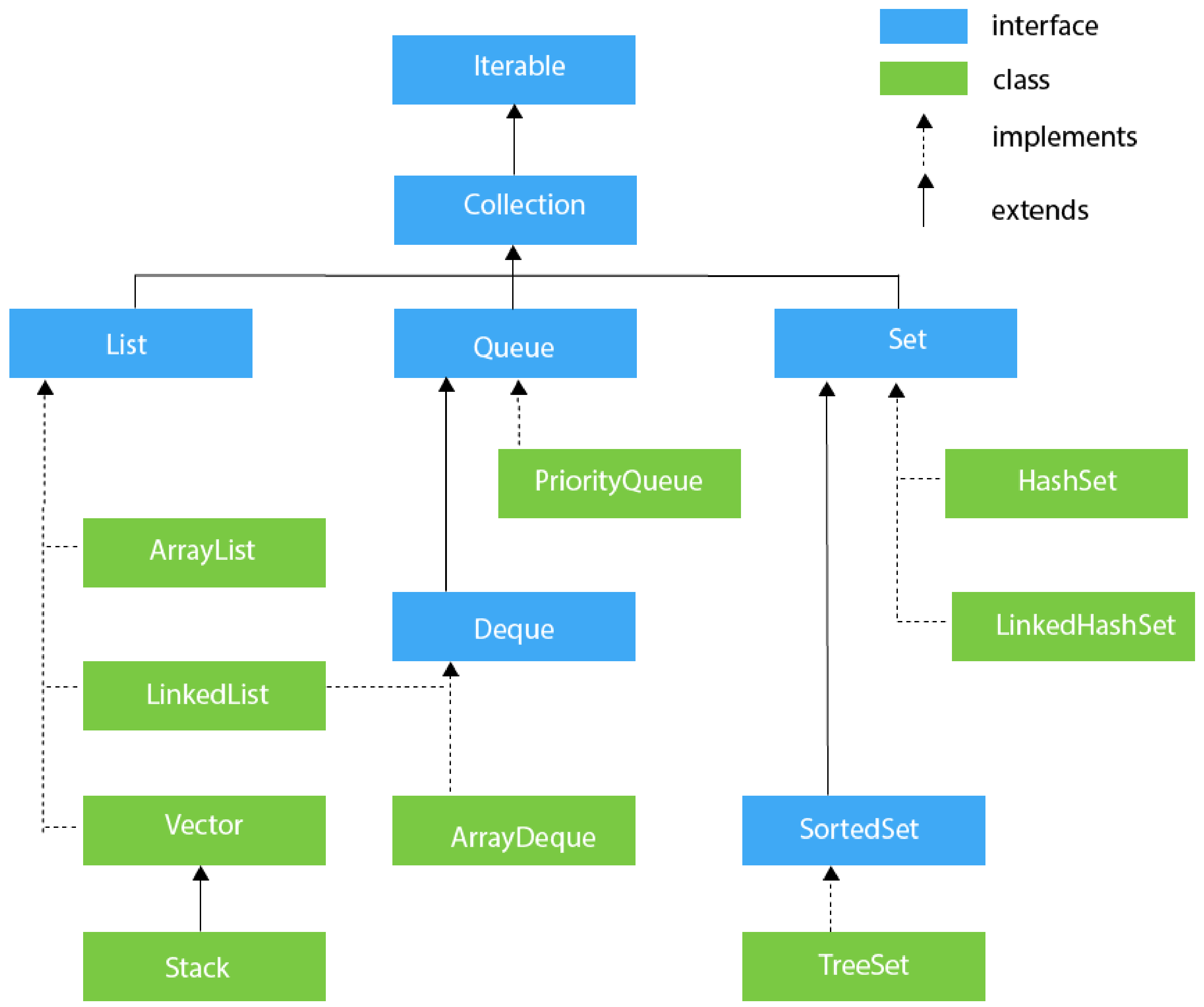
What is Collection framework
The Collection framework represents a unified architecture for storing and manipulating a group of objects. It has:

- Interfaces and its implementations, i.e., classes
- Algorithm

Collections

The hierarchy of Collection framework.

The **java.util** package contains all the classes and interfaces for the Collection framework.



https://www.javatpoint.com/collections-in-java

Collections

Projects List and Links

- CollectionArrayList - https://beginnersbook.com/2013/12/java-arraylist/
- CollectionLinkedList - https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/
- CollectionVector - https://beginnersbook.com/2013/12/vector-in-java/
- CollectionStack - https://www.callicoder.com/java-stack/
- CollectionPriorityQueue - https://www.callicoder.com/java-priority-queue/
- CollectionPriorityQueueComparator - https://www.callicoder.com/java-priority-queue/
- CollectionArrayDeque - https://www.geeksforgeeks.org/arraydeque-in-java/
- CollectionHashSet - https://www.callicoder.com/java-hashset/
- CollectionHashSetFromAnotherCollection - https://www.callicoder.com/java-hashset/
- CollectionHashSetSimpleOperations - https://www.callicoder.com/java-hashset/
- CollectionHashSetRemoveElements - https://www.callicoder.com/java-hashset/
- CollectionHashSetIteration - https://www.callicoder.com/java-hashset/
- CollectionHashSetUserDefinedObjects - https://www.callicoder.com/java-hashset/
- CollectionLinkedHashSet - https://www.geeksforgeeks.org/linkedhashset-in-java-with-examples/
- CollectionTreeSet - https://www.callicoder.com/java-treeset/
- CollectionTreeSetCustomComparator - https://www.callicoder.com/java-treeset/
- CollectionTreeSetElementsAccess - https://www.callicoder.com/java-treeset/
- CollectionTreeSetUserDefinedObjects - https://www.callicoder.com/java-treeset/

Difference Between Arrays and Collections in Java

The main advantage of Collection Framework are Collections are grow-able in nature, hence based on our requirement we can increase or decrease the size. Some others advantage are given below;

- Collection is re-sizable or dynamically draw-able memory.
  (can be garbage collected)
- Provides useful data structures in the form of predefined classes that reduces programming affords.
- It support to store heterogeneous elements or object.
- It provides higher performance.
- It provides Extendibility (depends on incoming flow of data, if the size of collection framework variable is increasing than the collection framework variable is containing Extendibility feature).
- It provides adaptability facility
  (The process of adding the content of one collection framework variable to another collection framework either in the beginning or in the ending or in the middle in known as adaptability).
- It is one of the algorithmic oriented.
- It provides in-built sorting technique.
- It provides in-built searching technique.
- It provides higher preliminary concepts of Data Structure such as:- Stack, Queue, LinkedList, Trees ..etc.

Difference Between Arrays and Collections in Java

Array	Collection
Arrays are fixed in size and hence once we created an array we are not allowed to increase or decrease the size based on our requirement.	Collections are grow-able in nature and hence based on our requirement we can increase or decrease the size.
Arrays can hold both primitives as well as objects.	Collections can hold only objects but not primitive.
Performance point of view arrays faster than collection.	Performance point of view collections are slower than array.
Arrays can hold only homogeneous elements.	Collections can hold both homogeneous and heterogeneous elements.