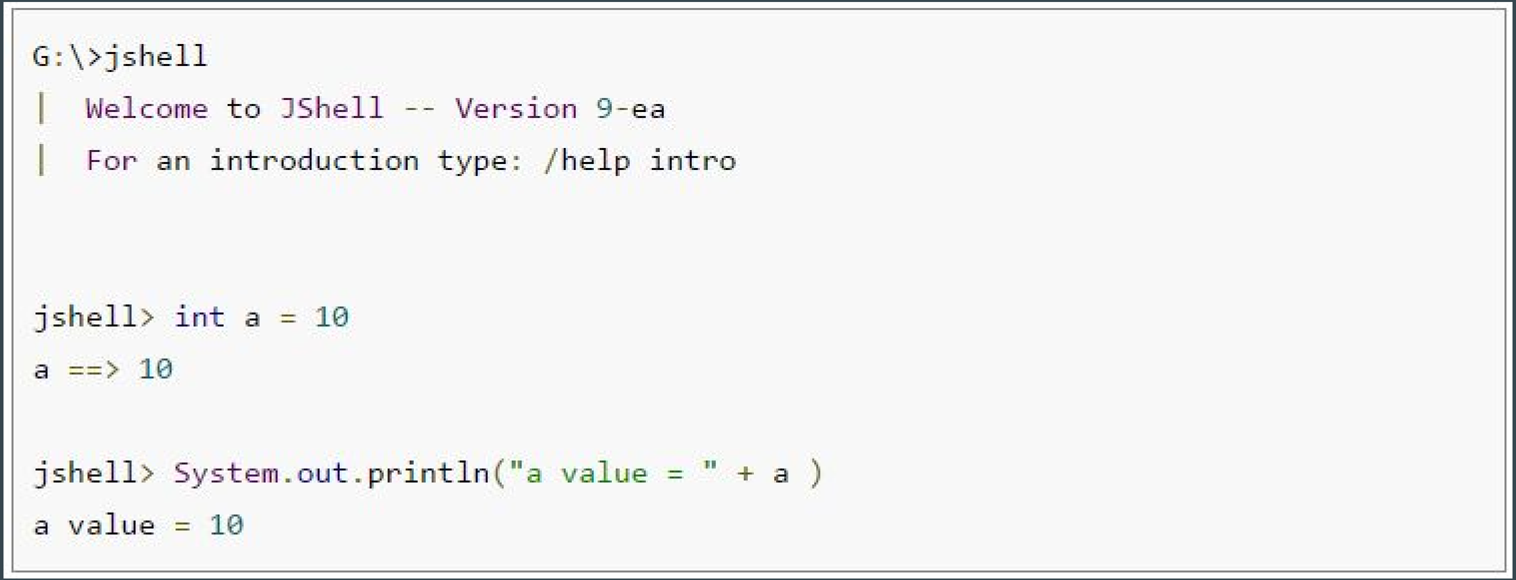
Java 9 Features Overview

- Java REPL
- Java Module System (Jigsaw Project)
- Misc

Java 9 REPL (Jshell)



```
G:\>jshell
|  Welcome to JShell -- Version 9-ea
|  For an introduction type: /help intro



jshell> int a = 10
a ==> 10

jshell> System.out.println("a value = " + a )
a value = 10
```

Java Modules System

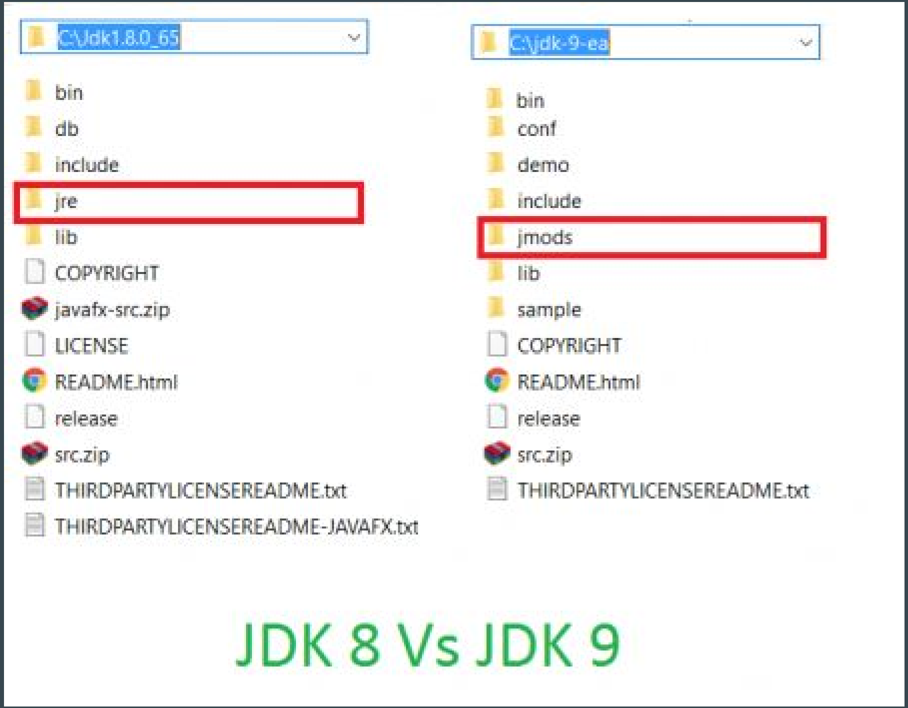
Problems (older versions)

- JDK is too big
- JAR files like rt.jar etc are too big to use in small devices and applications.
- Not able to support better Performance.
- Current Java System is too open. Everyone can access it. No Encapsulation
- Hard to Test and Maintain applications. Coupling between components
- As User can access Internal APIs too, Security is also big issue.

Advantajes

- Smaller modules, we can use whatever modules we want. Easier to scale Small devices.
- Ease of Testing and Maintainability.
- Supports better Performance.
- Strong Encapsulation. We cannot access Internal Non-Critical APIs anymore.
- Modules can hide unwanted and internal details. Better Security.
- Less Coupling between components.
- Its easy to support Single Responsibility Principle (SRP).

Java Module System (cont)



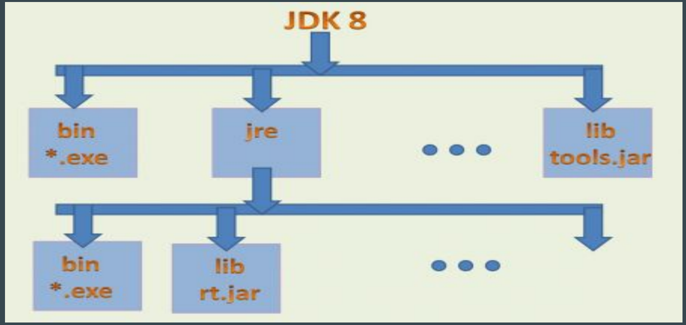
C:\Jdk1.8.0_65

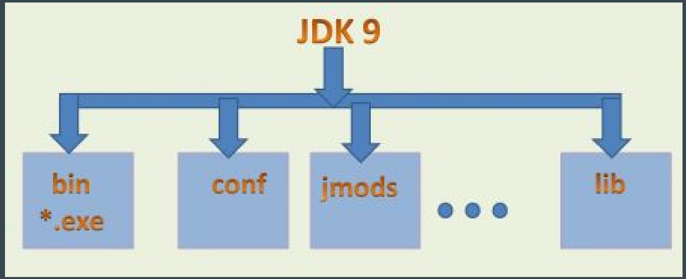
bin
db
include
jre
lib
COPYRIGHT
javafx-src.zip
LICENSE
README.html
release
src.zip
THIRDPARTYLICENSEREADME.txt
THIRDPARTYLICENSEREADME-JAVAFX.txt

C:\jdk-9-ea

bin
conf
demo
include
jmods
lib
sample
COPYRIGHT
README.html
release
src.zip
THIRDPARTYLICENSEREADME.txt

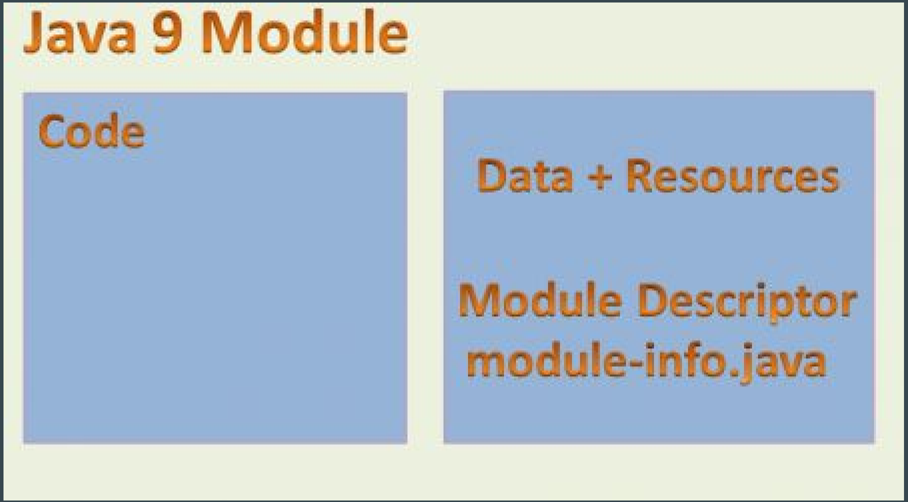
JDK 8 Vs JDK 9





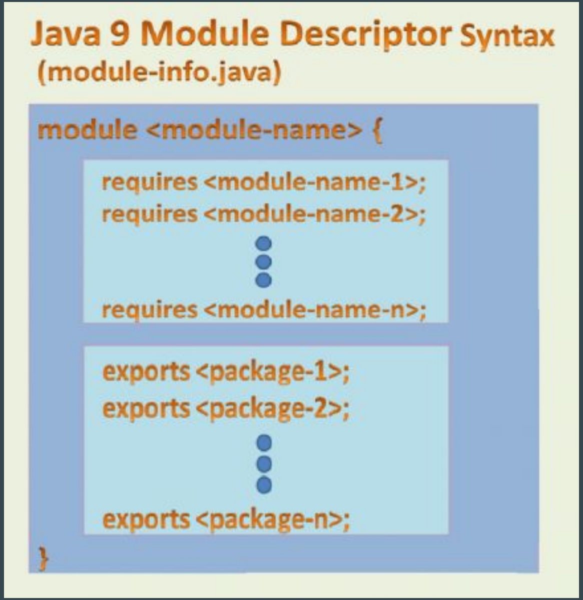
Java 9 Module

- Unique Module Name
- One Module Descriptor (module-info.java)
- Set of Packages
- Set of Types and Resources



- All JDK Mdoules starts with “jdk.*”
- All Java SE Specifications Modules starts with “java.*”

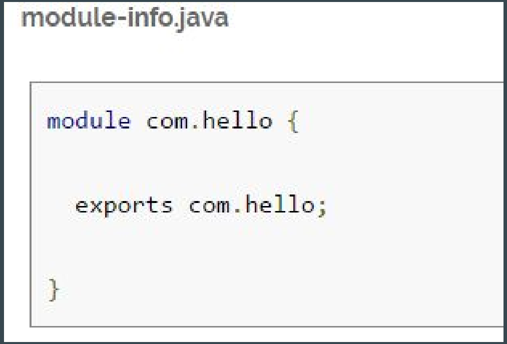
Java 9 Module (Cont)



Java 9 Module Descriptor Syntax (module-info.java)

```
module <module-name> {
    requires <module-name-1>;
    requires <module-name-2>;
    ⋮
    requires <module-name-n>;

    exports <package-1>;
    exports <package-2>;
    ⋮
    exports <package-n>;
}
```

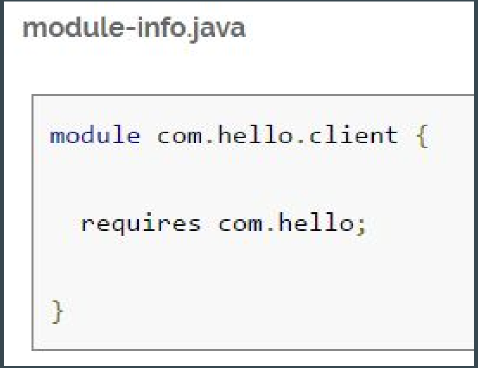


module-info.java

```
module com.hello {

  exports com.hello;

}
```



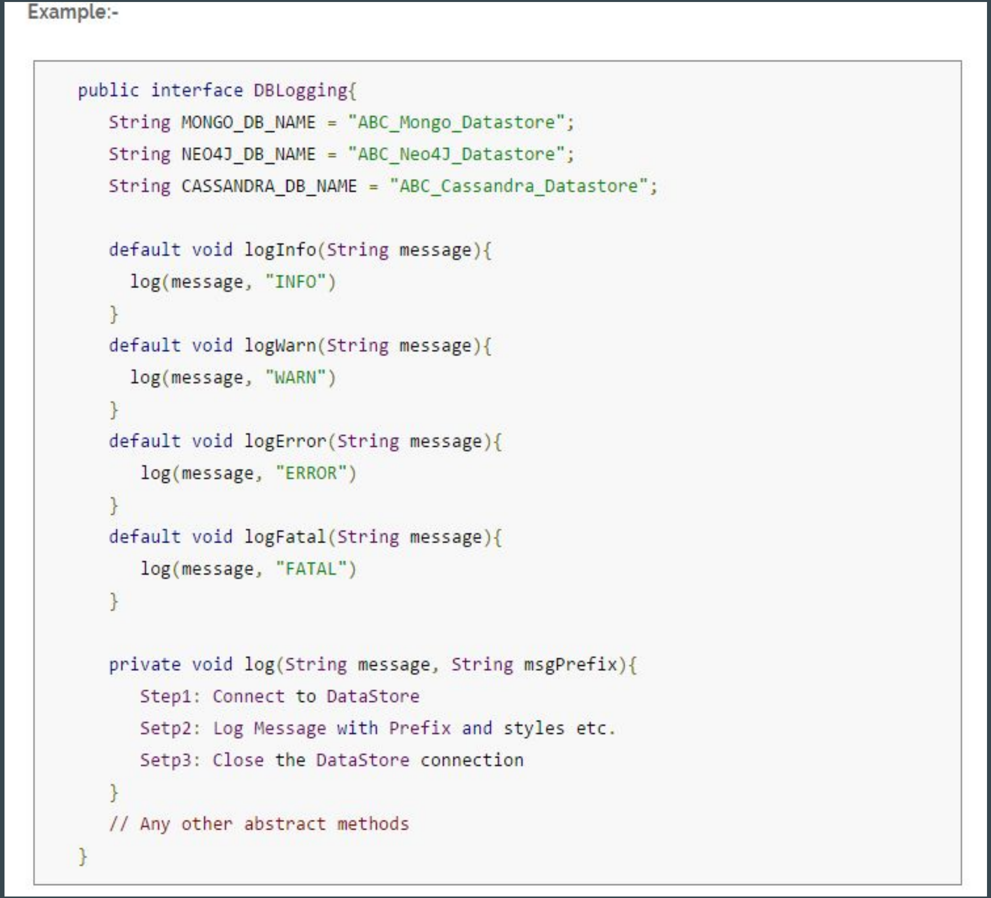
module-info.java

```
module com.hello.client {

  requires com.hello;

}
```

Private Methods on interfaces



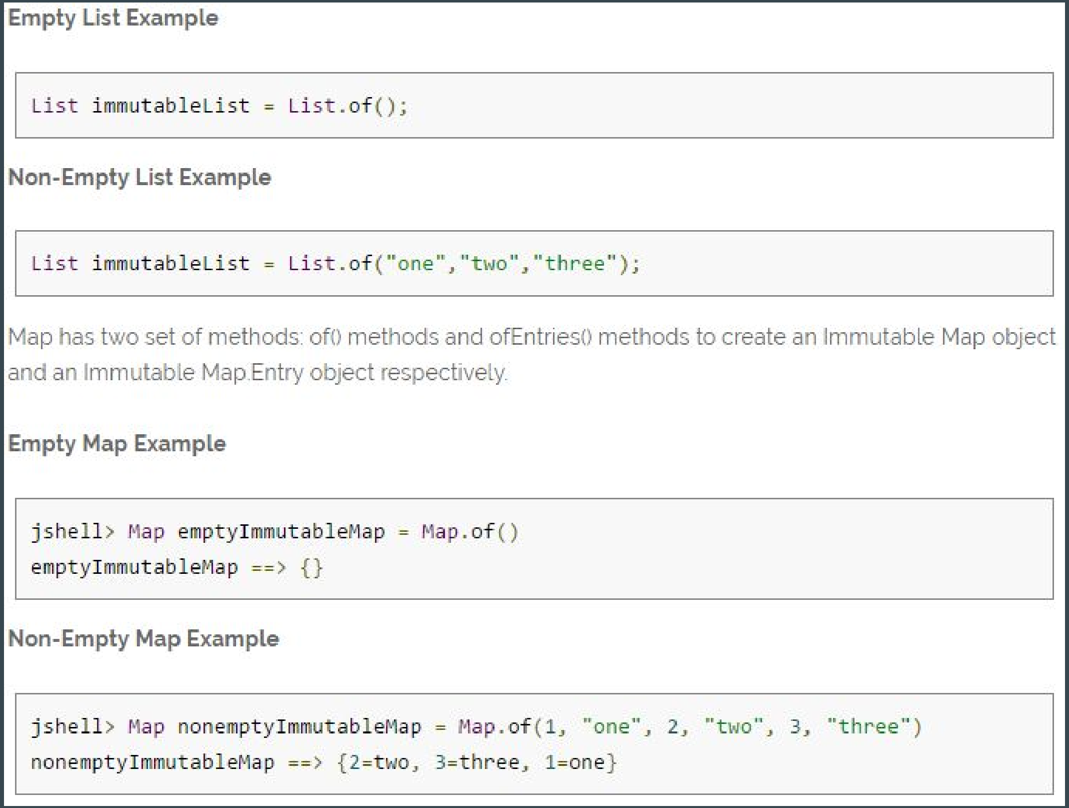
Example:-

```
public interface DBLogging{
   String MONGO_DB_NAME = "ABC_Mongo_Datastore";
   String NEO4J_DB_NAME = "ABC_Neo4J_Datastore";
   String CASSANDRA_DB_NAME = "ABC_Cassandra_Datastore";

   default void logInfo(String message){
     log(message, "INFO")
   }
   default void logWarn(String message){
     log(message, "WARN")
   }
   default void logError(String message){
      log(message, "ERROR")
   }
   default void logFatal(String message){
      log(message, "FATAL")
   }

   private void log(String message, String msgPrefix){
      Step1: Connect to DataStore
      Setp2: Log Message with Prefix and styles etc.
      Setp3: Close the DataStore connection
   }
   // Any other abstract methods
}
```

Factory Methods for Immutable List, Set, Map and Map.Entry



Empty List Example

```
List immutableList = List.of();
```

Non-Empty List Example

```
List immutableList = List.of("one","two","three");
```

Map has two set of methods: of() methods and ofEntries() methods to create an Immutable Map object and an Immutable Map.Entry object respectively.

Empty Map Example

```
jshell> Map emptyImmutableMap = Map.of()
emptyImmutableMap ==> {}
```

Non-Empty Map Example

```
jshell> Map nonemptyImmutableMap = Map.of(1, "one", 2, "two", 3, "three")
nonemptyImmutableMap ==> {2=two, 3=three, 1=one}
```

Reactive Streams

Java SE 9 Reactive Streams API is a Publish/Subscribe Framework to implement Asynchronous, Scalable and Parallel applications very easily using Java language.

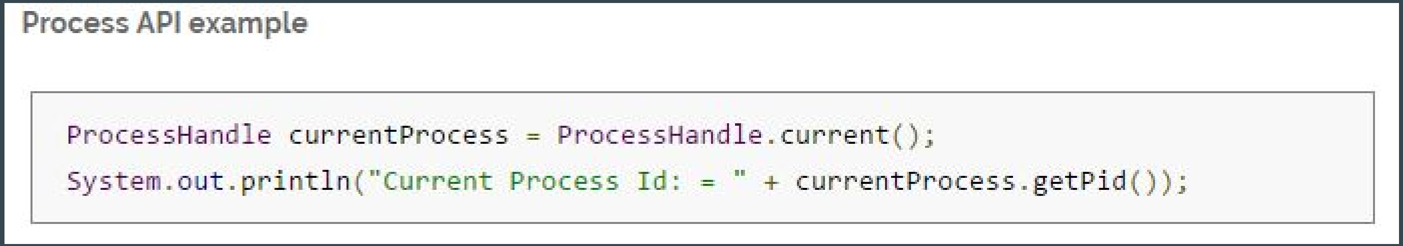
Java SE 9 has introduced the following API to develop Reactive Streams in Java-based applications.

- java.util.concurrent.Flow
- java.util.concurrent.Flow.Publisher
- java.util.concurrent.Flow.Subscriber
- java.util.concurrent.Flow.Processor

Process API improvemets

Two new interfaces

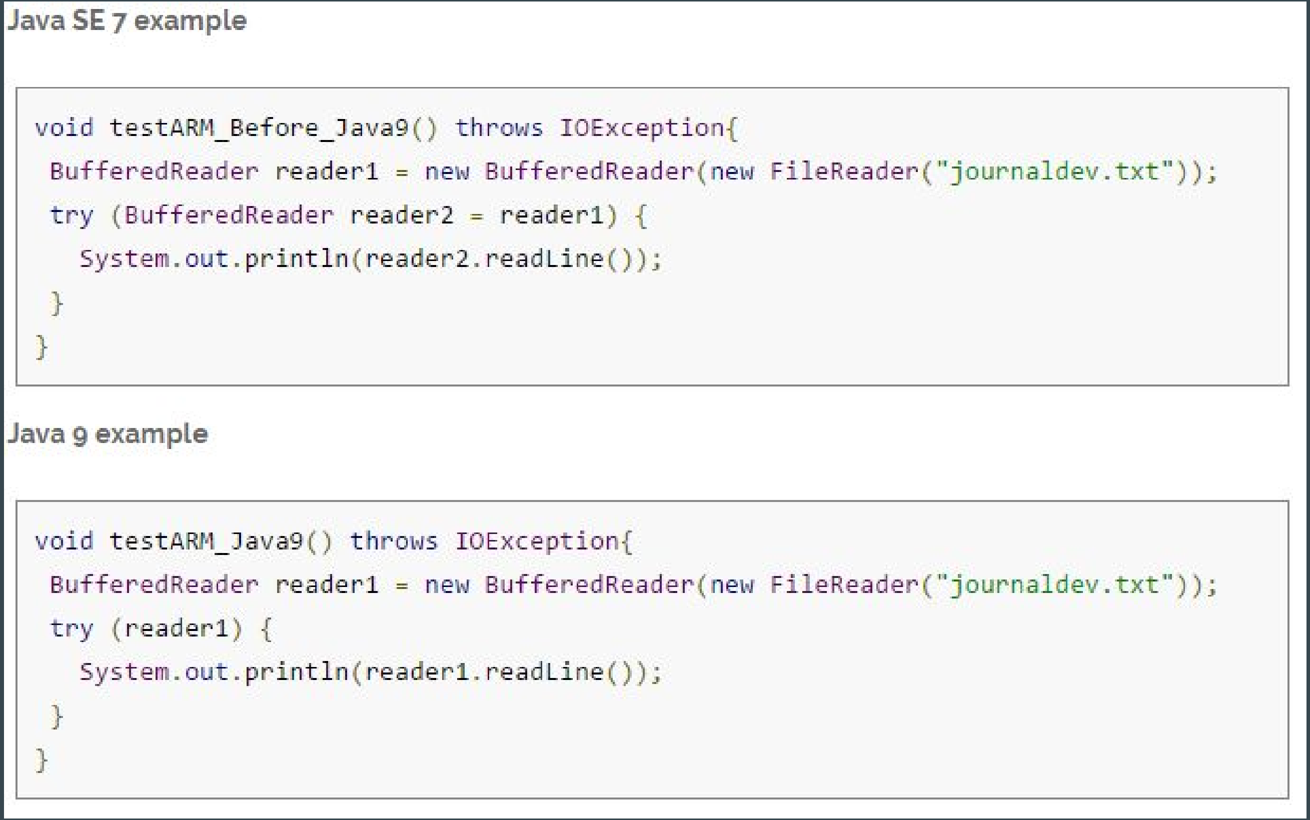
- java.lang.ProcessHandle
- java.lang.ProcessHandle.Info



Process API example

```
ProcessHandle currentProcess = ProcessHandle.current();
System.out.println("Current Process Id: = " + currentProcess.getPid());
```

Try with resources improvement



Java SE 7 example

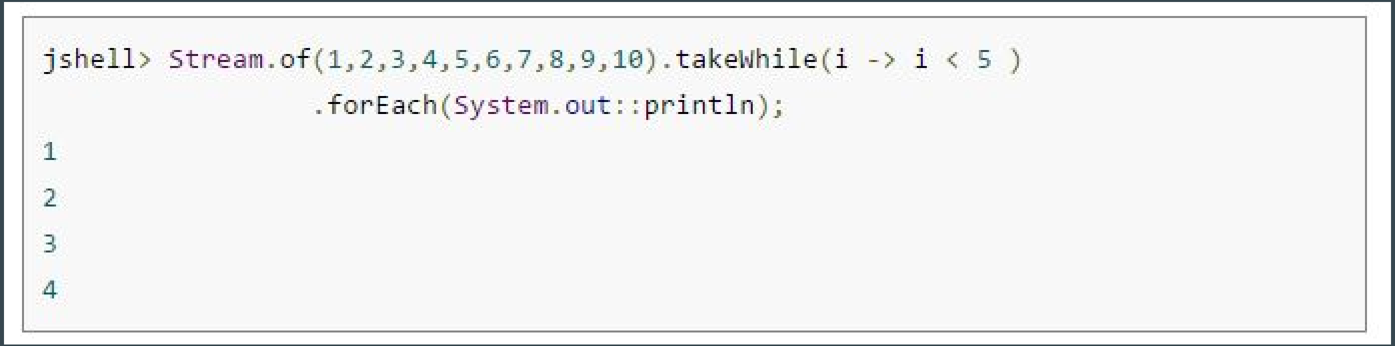
```
void testARM_Before_Java9() throws IOException{
 BufferedReader reader1 = new BufferedReader(new FileReader("journaldev.txt"));
 try (BufferedReader reader2 = reader1) {
   System.out.println(reader2.readLine());
 }
}
```

Java 9 example

```
void testARM_Java9() throws IOException{
 BufferedReader reader1 = new BufferedReader(new FileReader("journaldev.txt"));
 try (reader1) {
   System.out.println(reader1.readLine());
 }
}
```

Streams API improvements

Included more methods like: takeWhile and dropWhile



```
jshell> Stream.of(1,2,3,4,5,6,7,8,9,10).takeWhile(i -> i < 5 )
                  .forEach(System.out::println);
1
2
3
4
```

More features ...

- GC (Garbage Collector) Improvements
- Enhaced @Deprecated annotation
- Http 2 Client
- Optional class improvements
- Multi resolution Image API
- Stack-Walking API
- Deprecate the Applet API
- Java Platform Logging API and Service
- Compact Strings
- Javadoc Search , HTML5 Javadoc
- More ....

Questions?