Contents

Preface

Java se9 was introduced in 21/09/2019 some features impove bellow:

Modulazation of the JDK under Project Jigsaw

Provided Money and Currency API

Try-with

Annonymous Classes Improvement

Java annotation @SafeVarags

Provided java linker.

Interface Private Methods

HTTP 2 Client

JShell REPL Tool

Platform and JVM Logging

Process API Updates

Collection Factory Of Method

Stream API Improvements

Multi-Release JRA Files

@Deprecated Tag Changes

Start

Modulazation of the JDK under Project Jigsaw

Java Module System is a major change in Java 9 version. Java added this feature to collect java packages and code into a single unit called module. The verions earlier of java, there was no concept of module to create modular java applications, that why size of application increased and difficult to move around. Even JDK itself was too heavy in size.

Modular Jar file is introduced. This Jar contains module-info.class file in its root folder.

To use **a module**, include the jar file into **modulepath** instead of the classPaht. A **module jar** file added to **classPath** is normal jar file and module-infor.class file will be ignored.

A class is container of fields and methods

A packages is a container of classes and interfaces

A module is a container of packages

Detail visit here: https://sharecodefull.blogspot.com/2020/12/java-platform-module-system.html

Provided Money and Currency API

JSR 354 java Specification request Money and Currency API. The JSR did not make its way into JDK 9 but candidate for future JDK releases. It has version (moneta) implementation of version original (javax.money) is used a lot so i will introduced it.

Features:

To provide an API for handing and calculating monetary amounts.

To define classes representing currencies and monetary amounts as well as rounding, add, subtract, mulity, divide, exchange currencies, formating, parsing.

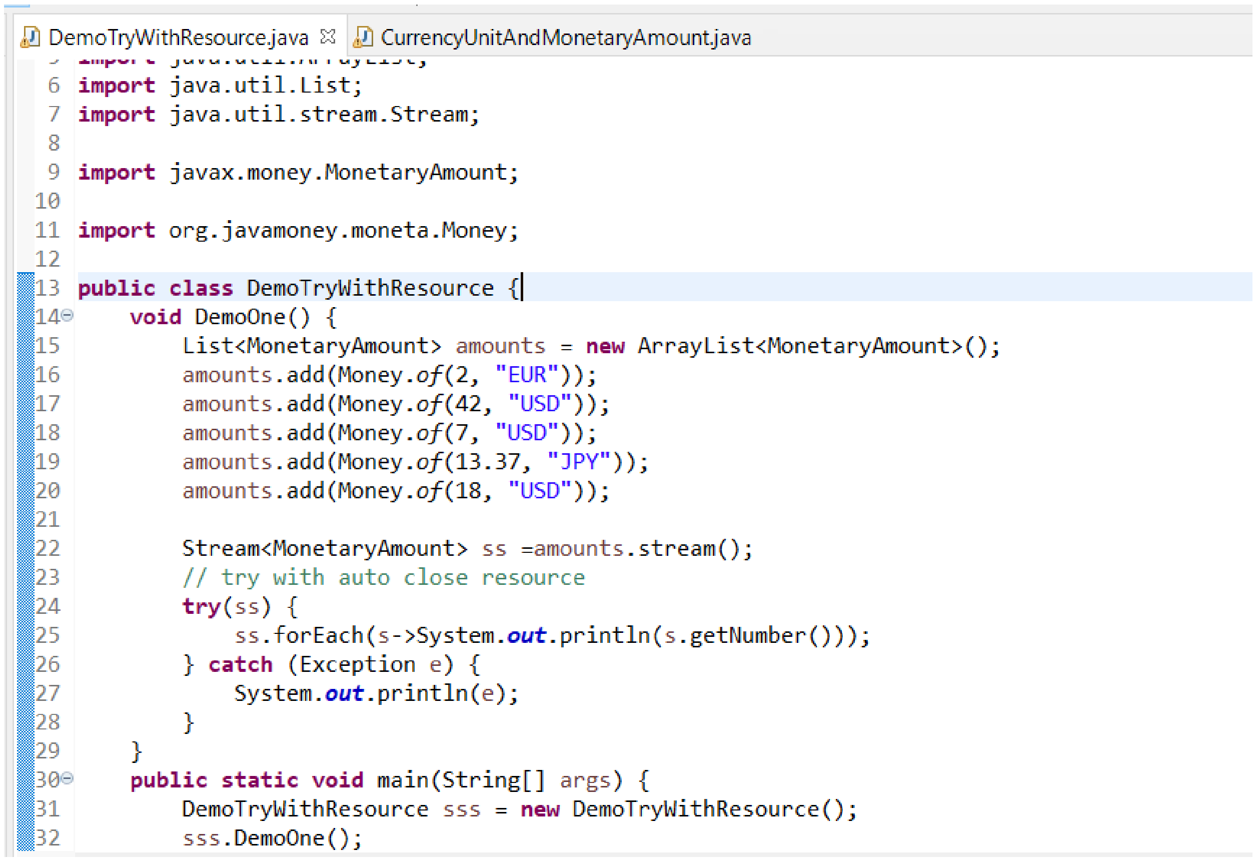
Detail visit here: https://sharecodefull.blogspot.com/2021/01/provided-money-and-currency-api.html

Try-with

Java introduced try-with-resource feature in java 7 that helps to close resource automatically after being used.

In other words, we can say that we don't need to close resource (file, connection, network, etc) explicitly, try-with-resource close that automatically by using AutoClosable interface.

In java 7, try-with-resource has a limitation that requires resource to declare locally within its block.

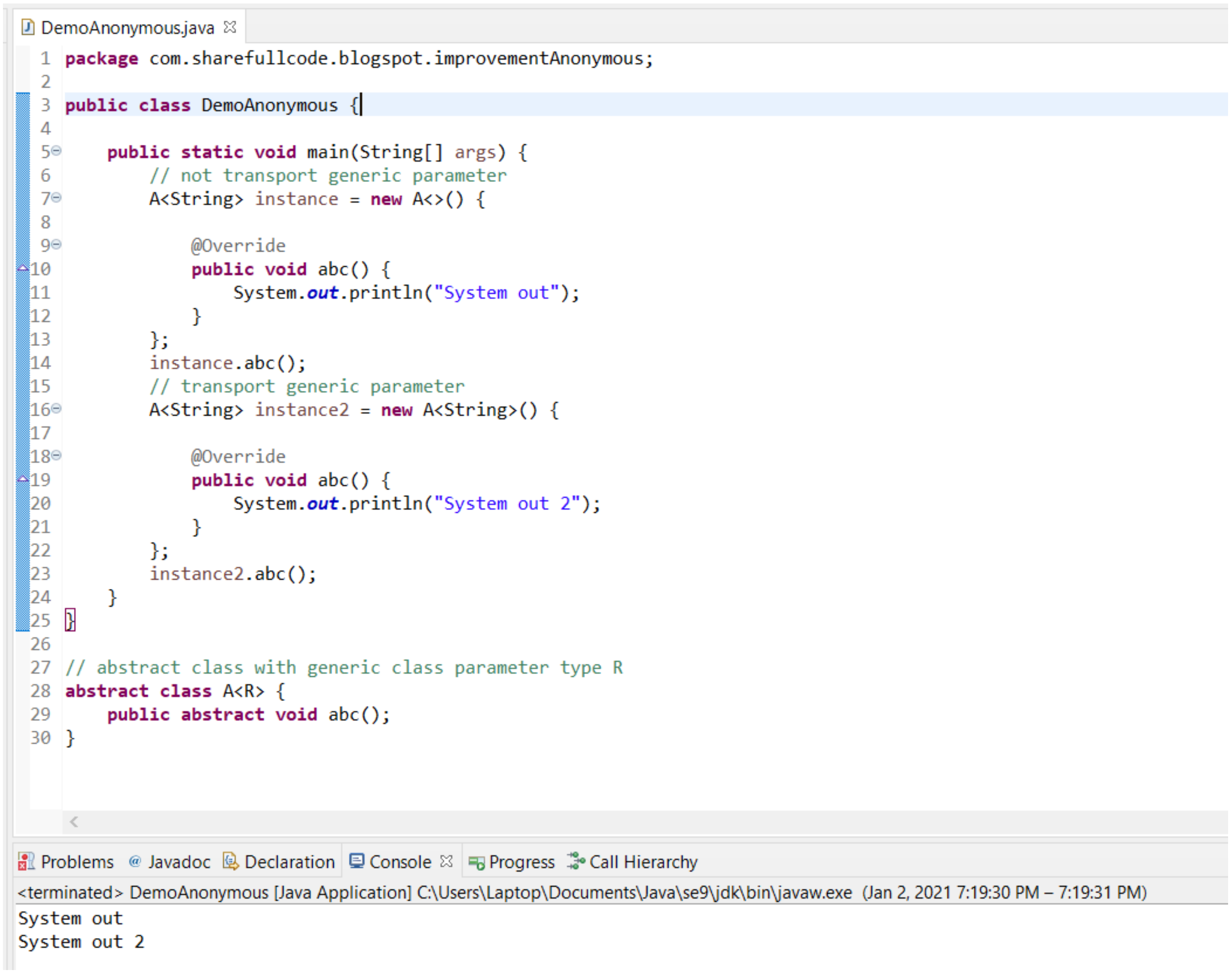

DemoTryWithResource.java ☒ CurrencyUnitAndMonetaryAmount.java

```
import java.util.List;
import java.util.stream.Stream;

import javax.money.MonetaryAmount;

import org.javamoney.moneta.Money;

public class DemoTryWithResource {
    void DemoOne() {
        List<MonetaryAmount> amounts = new ArrayList<MonetaryAmount>();
        amounts.add(Money.of(2, "EUR"));
        amounts.add(Money.of(42, "USD"));
        amounts.add(Money.of(7, "USD"));
        amounts.add(Money.of(13.37, "JPY"));
        amounts.add(Money.of(18, "USD"));

        Stream<MonetaryAmount> ss =amounts.stream();
        // try with auto close resource
        try(ss) {
            ss.forEach(s->System.out.println(s.getNumber()));
        } catch (Exception e) {
            System.out.println(e);
        }
    }
    public static void main(String[] args) {
        DemoTryWithResource sss = new DemoTryWithResource();
        sss.DemoOne();
```

Annonymous Classes Improvement

In java 9 introduced a new feature that allows us to use diamond operator with anonymous classes, In java 9 as long as the inferred type is denotable, we can use the diamond operator when we create an anonymous inner class.

In java 9 not transport parameter with generic still Ok

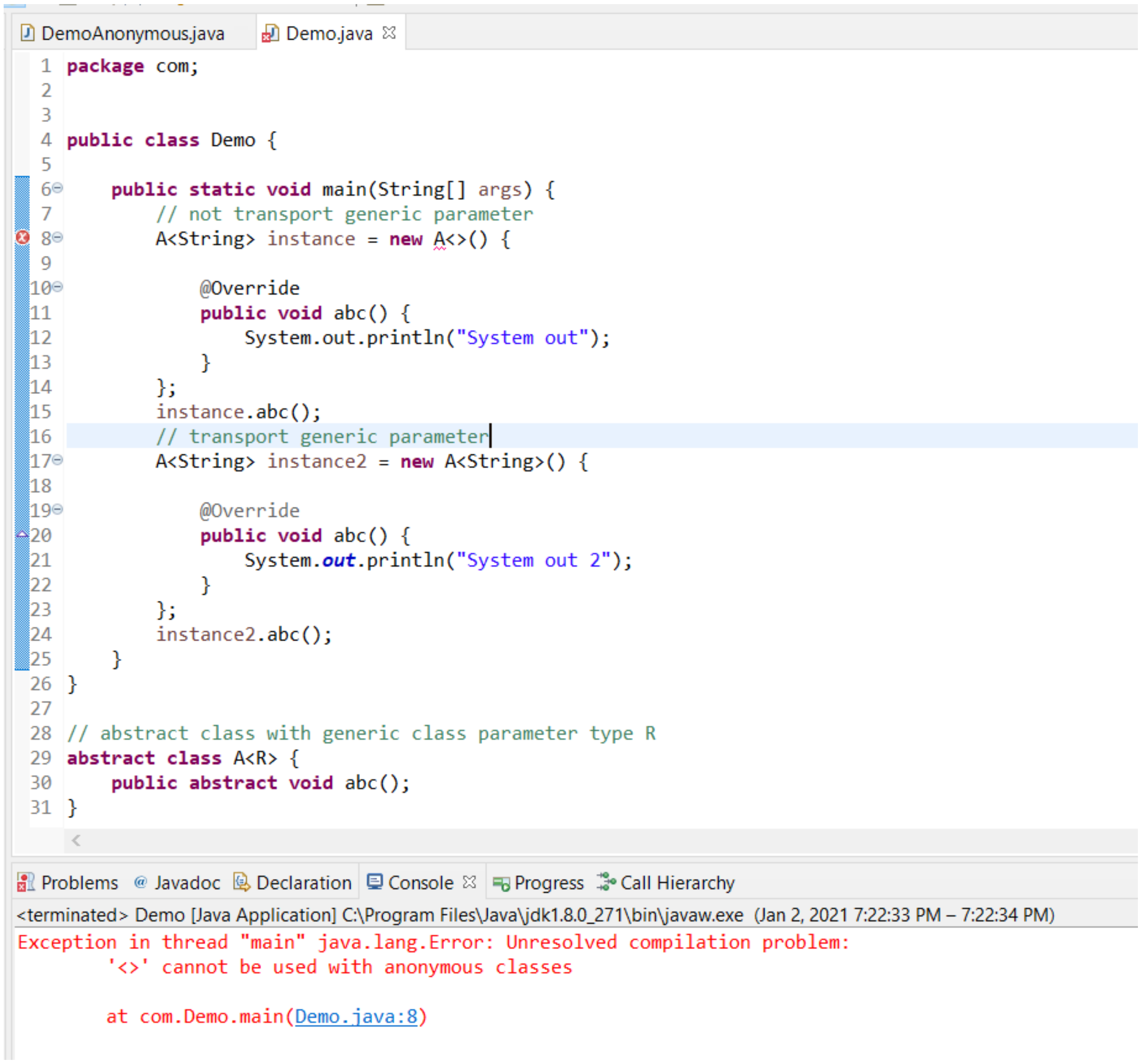


DemoAnonymous.java

```java
package com.sharefullcode.blogspot.improvementAnonymous;

public class DemoAnonymous {

    public static void main(String[] args) {
        // not transport generic parameter
        A<String> instance = new A<>() {

            @Override
            public void abc() {
                System.out.println("System out");
            }
        };
        instance.abc();
        // transport generic parameter
        A<String> instance2 = new A<String>() {

            @Override
            public void abc() {
                System.out.println("System out 2");
            }
        };
        instance2.abc();
    }
}

// abstract class with generic class parameter type R
abstract class A<R> {
    public abstract void abc();
}
```

```
<terminated> DemoAnonymous [Java Application] C:\Users\Laptop\Documents\Java\se9\jdk\bin\javaw.exe  (Jan 2, 2021 7:19:30 PM – 7:19:31 PM)
System out
System out 2
```

With java 8 error complier time



```java
package com;


public class Demo {

    public static void main(String[] args) {
        // not transport generic parameter
        A<String> instance = new A<>() {

            @Override
            public void abc() {
                System.out.println("System out");
            }
        };
        instance.abc();
        // transport generic parameter
        A<String> instance2 = new A<String>() {

            @Override
            public void abc() {
                System.out.println("System out 2");
            }
        };
        instance2.abc();
    }
}

// abstract class with generic class parameter type R
abstract class A<R> {
    public abstract void abc();
}
```

```
<terminated> Demo [Java Application] C:\Program Files\Java\jdk1.8.0_271\bin\javaw.exe  (Jan 2, 2021 7:22:33 PM – 7:22:34 PM)
Exception in thread "main" java.lang.Error: Unresolved compilation problem: 
	'<>' cannot be used with anonymous classes

	at com.Demo.main(Demo.java:8)
```

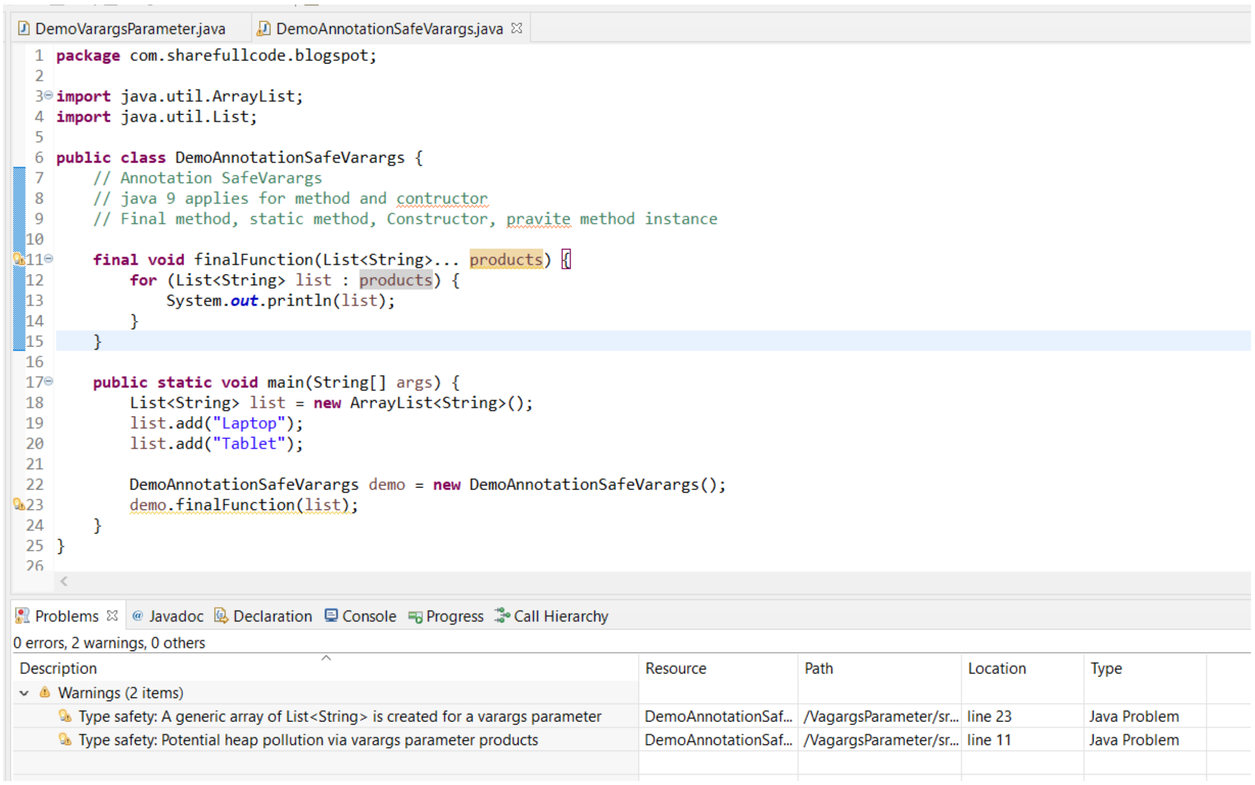
Java annotation @SafeVarags And Variables Argument (Varargs)

Annotation SafeVarargs

It is an annotation which applies on a method or constructor that takes varags parameters. It is used to ensure that the method does not perform unsafe operations on its varags parameters.

It was included in java 7 and can be only applied on : Final method, Static methods, Constructors.

From java 9 it can also be used with private instance methods.



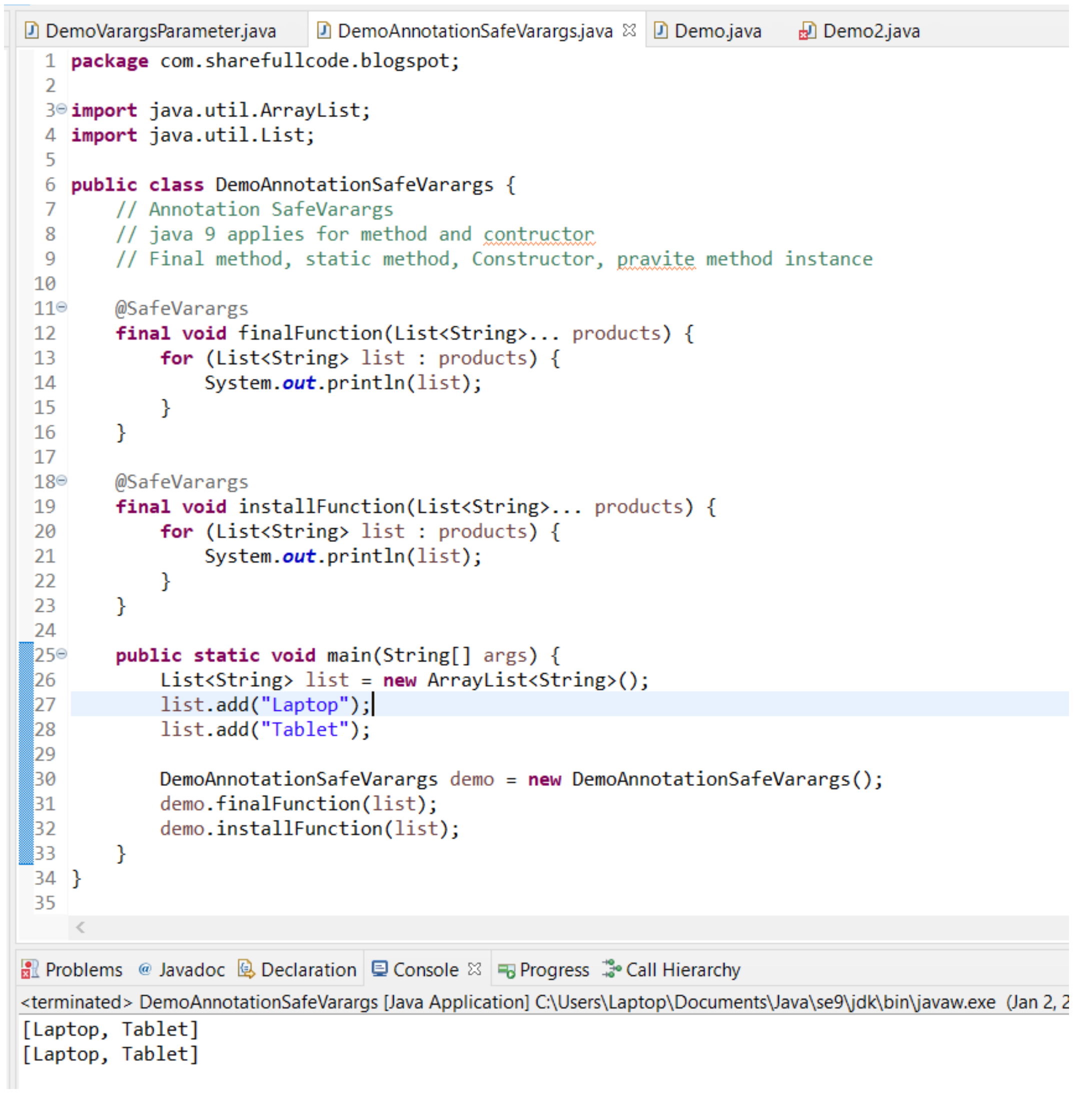
DemoVarargsParameter.java | DemoAnnotationSafeVarargs.java

```java
package com.sharefullcode.blogspot;

import java.util.ArrayList;
import java.util.List;

public class DemoAnnotationSafeVarargs {
    // Annotation SafeVarargs
    // java 9 applies for method and contructor
    // Final method, static method, Constructor, pravite method instance

    final void finalFunction(List<String>... products) {
        for (List<String> list : products) {
            System.out.println(list);
        }
    }

    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("Laptop");
        list.add("Tablet");

        DemoAnnotationSafeVarargs demo = new DemoAnnotationSafeVarargs();
        demo.finalFunction(list);
    }
}
```

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (2 items)				
Type safety: A generic array of List<String> is created for a varargs parameter	DemoAnnotationSaf...	/VagargsParameter/sr...	line 23	Java Problem
Type safety: Potential heap pollution via varargs parameter products	DemoAnnotationSaf...	/VagargsParameter/sr...	line 11	Java Problem

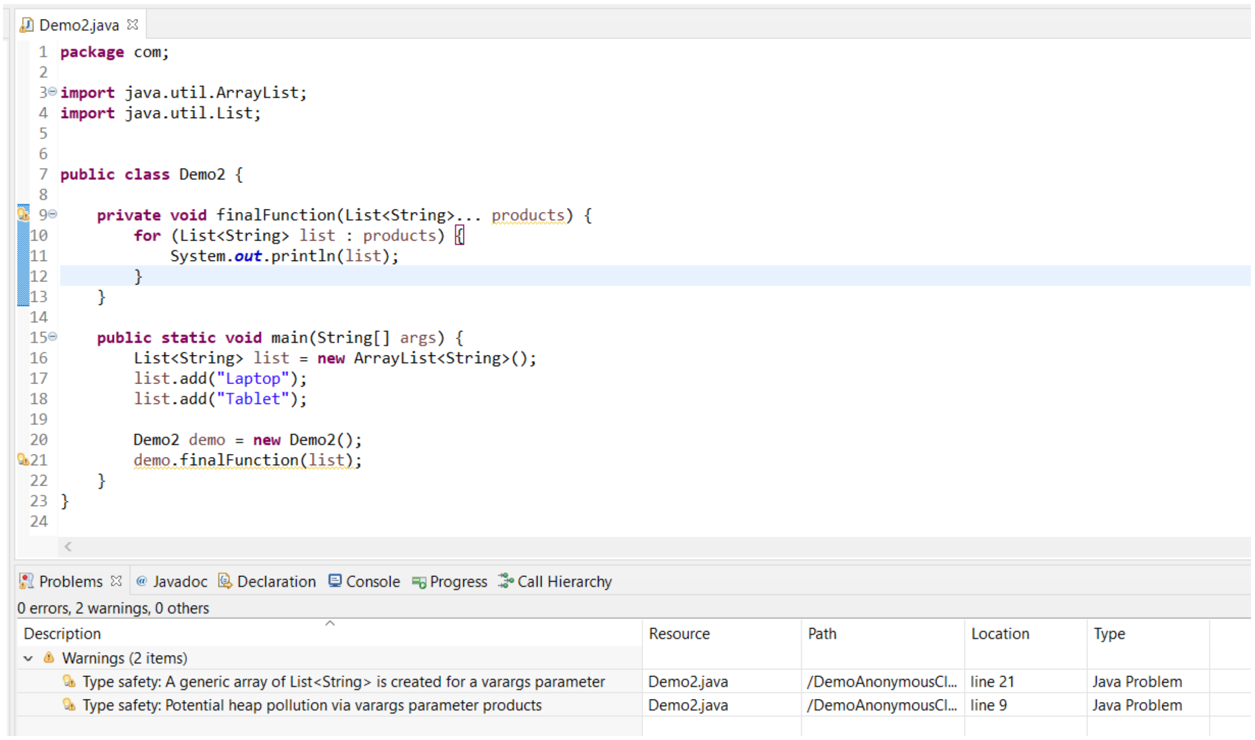
Add Annotation into class then run success



```java
package com.sharefullcode.blogspot;

import java.util.ArrayList;
import java.util.List;

public class DemoAnnotationSafeVarargs {
    // Annotation SafeVarargs
    // java 9 applies for method and contructor
    // Final method, static method, Constructor, pravite method instance

    @SafeVarargs
    final void finalFunction(List<String>... products) {
        for (List<String> list : products) {
            System.out.println(list);
        }
    }

    @SafeVarargs
    final void installFunction(List<String>... products) {
        for (List<String> list : products) {
            System.out.println(list);
        }
    }

    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("Laptop");
        list.add("Tablet");

        DemoAnnotationSafeVarargs demo = new DemoAnnotationSafeVarargs();
        demo.finalFunction(list);
        demo.installFunction(list);
    }
}
```

```
<terminated> DemoAnnotationSafeVarargs [Java Application] C:\Users\Laptop\Documents\Java\se9\jdk\bin\javaw.exe  (Jan 2, 2
[Laptop, Tablet]
[Laptop, Tablet]
```

But run with JDK older compile error with private instance method, when not add annotation SafeVarargs that warning , but when add annotation run error because version JDK older se9 not support for private instance method

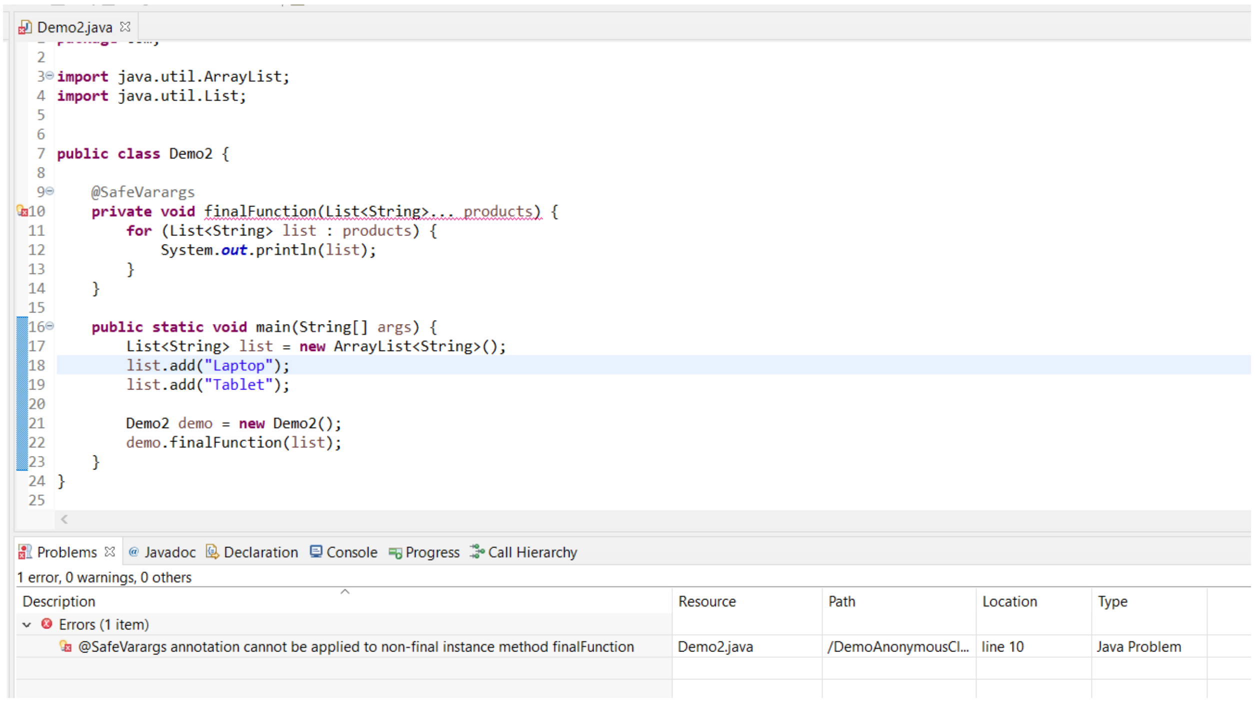


Demo2.java

```java
package com;

import java.util.ArrayList;
import java.util.List;


public class Demo2 {

    private void finalFunction(List<String>... products) {
        for (List<String> list : products) {
            System.out.println(list);
        }
    }

    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("Laptop");
        list.add("Tablet");

        Demo2 demo = new Demo2();
        demo.finalFunction(list);
    }
}
```

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (2 items)				
Type safety: A generic array of List<String> is created for a varargs parameter	Demo2.java	/DemoAnonymousCl...	line 21	Java Problem
Type safety: Potential heap pollution via varargs parameter products	Demo2.java	/DemoAnonymousCl...	line 9	Java Problem



Demo2.java

```java

import java.util.ArrayList;
import java.util.List;


public class Demo2 {

    @SafeVarargs
    private void finalFunction(List<String>... products) {
        for (List<String> list : products) {
            System.out.println(list);
        }
    }

    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("Laptop");
        list.add("Tablet");

        Demo2 demo = new Demo2();
        demo.finalFunction(list);
    }
}
```

1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
@SafeVarargs annotation cannot be applied to non-final instance method finalFunction	Demo2.java	/DemoAnonymousCl...	line 10	Java Problem

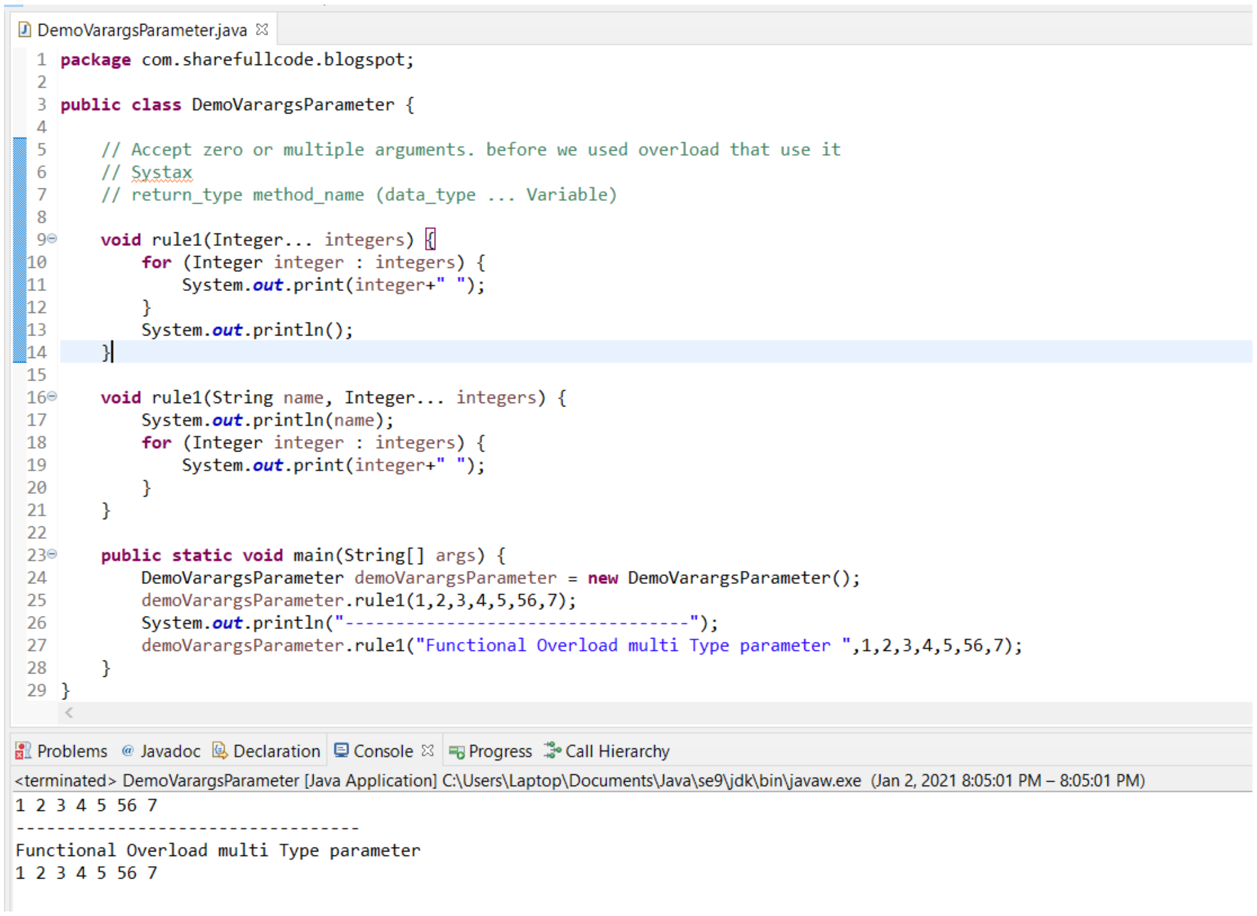
Variables Argument (Varargs)

It replace Overload function when the parameter has same type.

Has Rules: There can only one Varargs in the method, Varargs must be the last argument if has more parameters.

// void method (Int … a, String … b) complie error

// void method (Int … a, String b) complie error



DemoVarargsParameter.java

```java
package com.sharefullcode.blogspot;

public class DemoVarargsParameter {

    // Accept zero or multiple arguments. before we used overload that use it
    // Systax
    // return_type method_name (data_type ... Variable)

    void rule1(Integer... integers) {
        for (Integer integer : integers) {
            System.out.print(integer+" ");
        }
        System.out.println();
    }

    void rule1(String name, Integer... integers) {
        System.out.println(name);
        for (Integer integer : integers) {
            System.out.print(integer+" ");
        }
    }

    public static void main(String[] args) {
        DemoVarargsParameter demoVarargsParameter = new DemoVarargsParameter();
        demoVarargsParameter.rule1(1,2,3,4,5,56,7);
        System.out.println("----------------------------------");
        demoVarargsParameter.rule1("Functional Overload multi Type parameter ",1,2,3,4,5,56,7);
    }
}
```

```
<terminated> DemoVarargsParameter [Java Application] C:\Users\Laptop\Documents\Java\se9\jdk\bin\javaw.exe  (Jan 2, 2021 8:05:01 PM – 8:05:01 PM)
1 2 3 4 5 56 7
----------------------------------
Functional Overload multi Type parameter
1 2 3 4 5 56 7
```

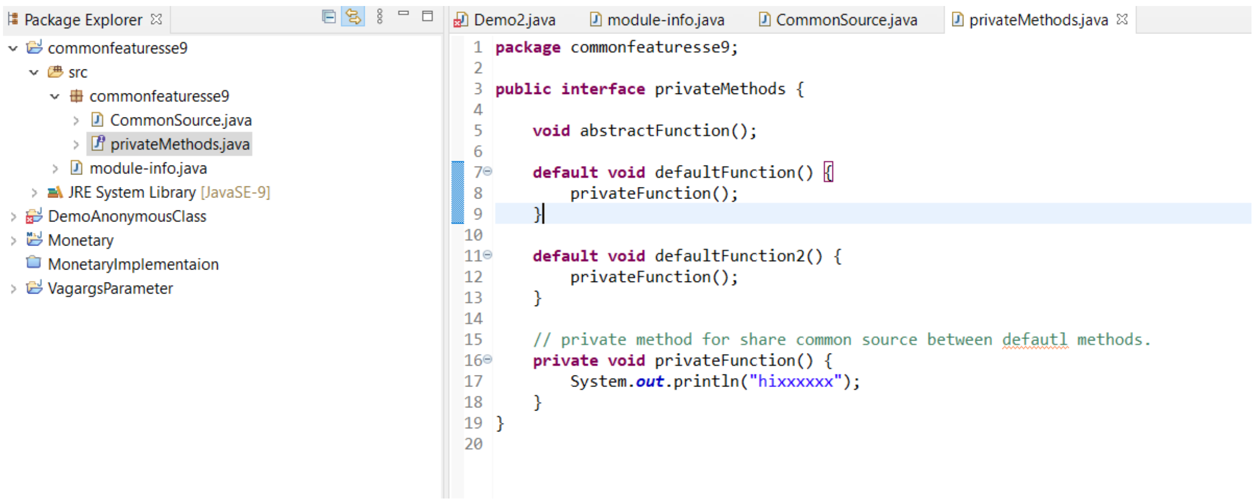
Provided java linker.

Java linker is a tool that can be used to assemble set of modules into a runtime image. It also allow us to assemble module’s dependencies into custom runtime image.

Link time is a phase between the compile and runtime. Jlink works there for linking and assemble modules to runtime image.

Interface Private Methods

In java 9 we can create private methods inside an interface. Interface allows us to declare private methods that help to share common code between non-abstract methods.

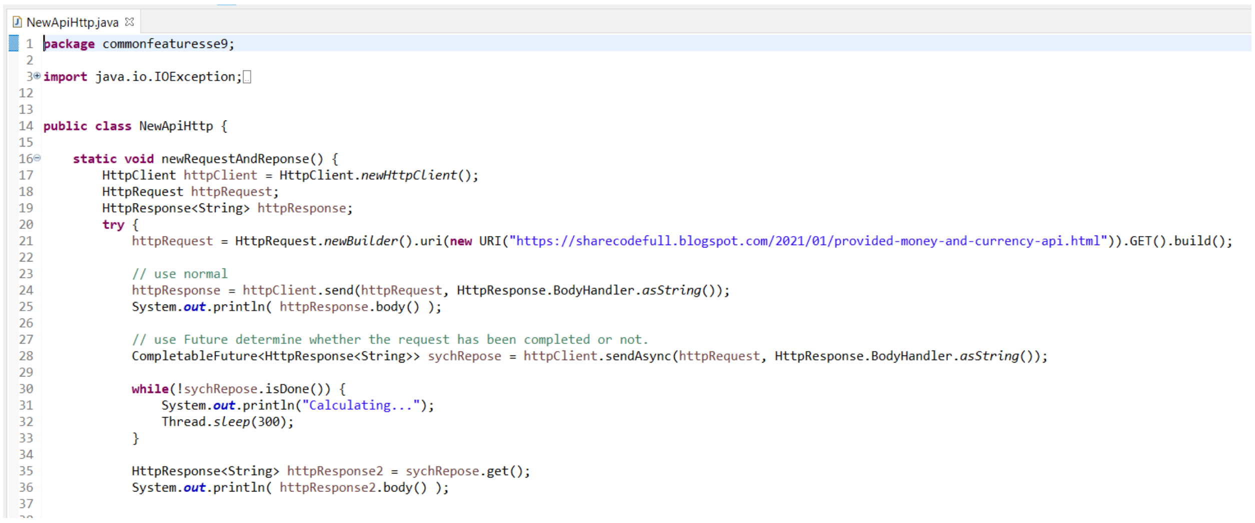


```java
package commonfeaturesse9;

public interface privateMethods {

    void abstractFunction();

    default void defaultFunction() {
        privateFunction();
    }

    default void defaultFunction2() {
        privateFunction();
    }

    // private method for share common source between defautl methods.
    private void privateFunction() {
        System.out.println("hixxxxxx");
    }
}
```

HTTP 2 Client

Http/1.1 client was released on 1997 a lot has changed since. So for Java 9 a new API been introduced that is cleaner and cleaner to use and which also support for http/2. New Api use major classes : HttpClient, HttpRequest, HttpResponse.

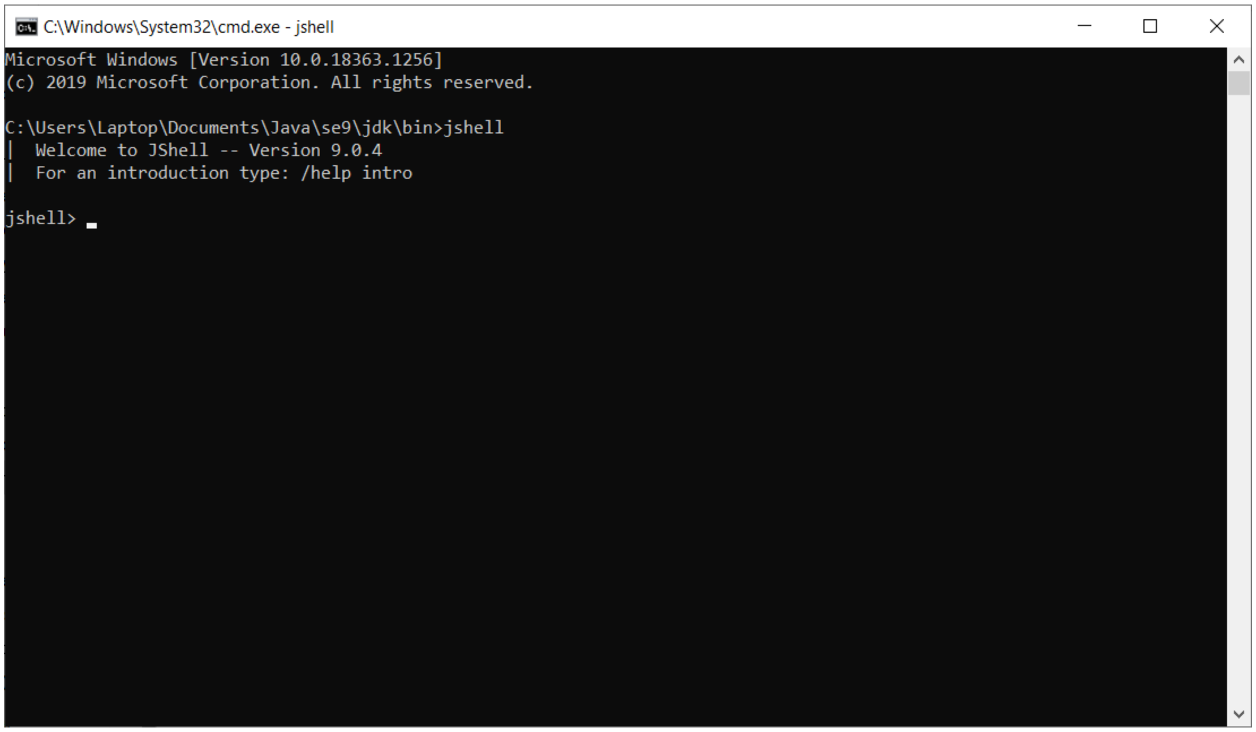
HttpClient support sych response by Future view detail demo:



```java
package commonfeaturesse9;

import java.io.IOException;


public class NewApiHttp {

    static void newRequestAndReponse() {
        HttpClient httpClient = HttpClient.newHttpClient();
        HttpRequest httpRequest;
        HttpResponse<String> httpResponse;
        try {
            httpRequest = HttpRequest.newBuilder().uri(new URI("https://sharecodefull.blogspot.com/2021/01/provided-money-and-currency-api.html")).GET().build();

            // use normal
            httpResponse = httpClient.send(httpRequest, HttpResponse.BodyHandler.asString());
            System.out.println( httpResponse.body() );

            // use Future determine whether the request has been completed or not.
            CompletableFuture<HttpResponse<String>> sychRepose = httpClient.sendAsync(httpRequest, HttpResponse.BodyHandler.asString());

            while(!sychRepose.isDone()) {
                System.out.println("Calculating...");
                Thread.sleep(300);
            }

            HttpResponse<String> httpResponse2 = sychRepose.get();
            System.out.println( httpResponse2.body() );

```

JShell REPL Tool

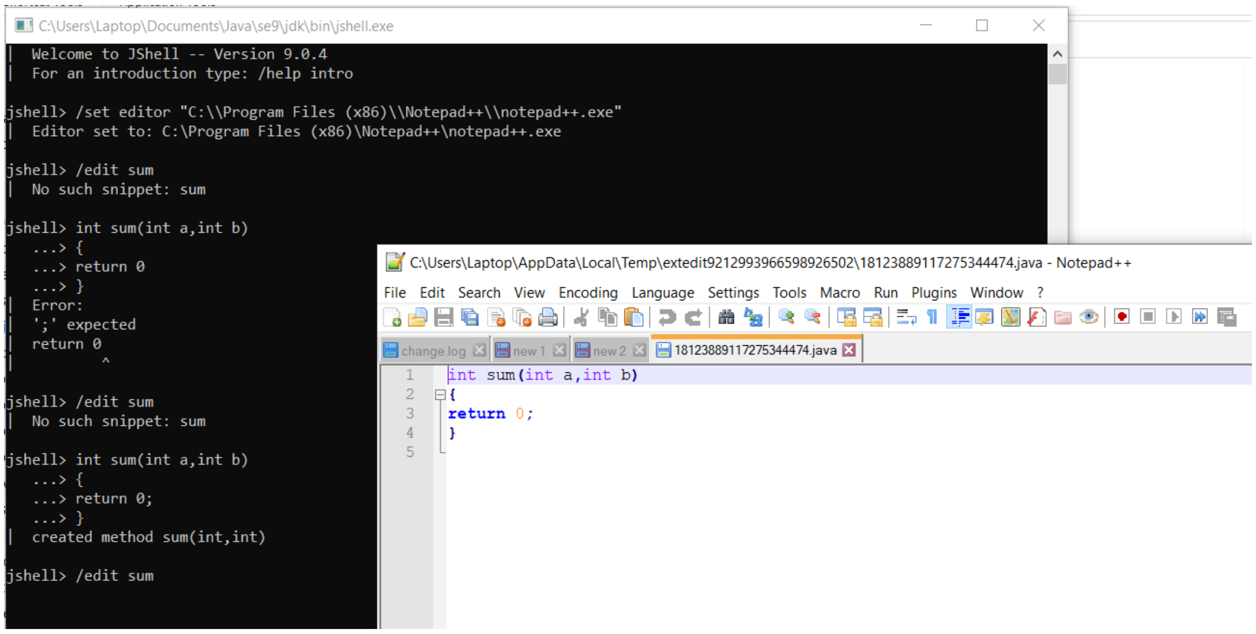
Jshell tool is a new command line interactive tool shipped with java 9 distribution to evaluate declearations, statements and expression written in java. JShell allows us to excute jav code snippet and get immediate results without having to create a solution or project.



C:\Windows\System32\cmd.exe - jshell

```
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Laptop\Documents\Java\se9\jdk\bin>jshell
|  Welcome to JShell -- Version 9.0.4
|  For an introduction type: /help intro

jshell> _
```

You can set editor external when save it run in jShell



C:\Users\Laptop\Documents\Java\se9\jdk\bin\jshell.exe

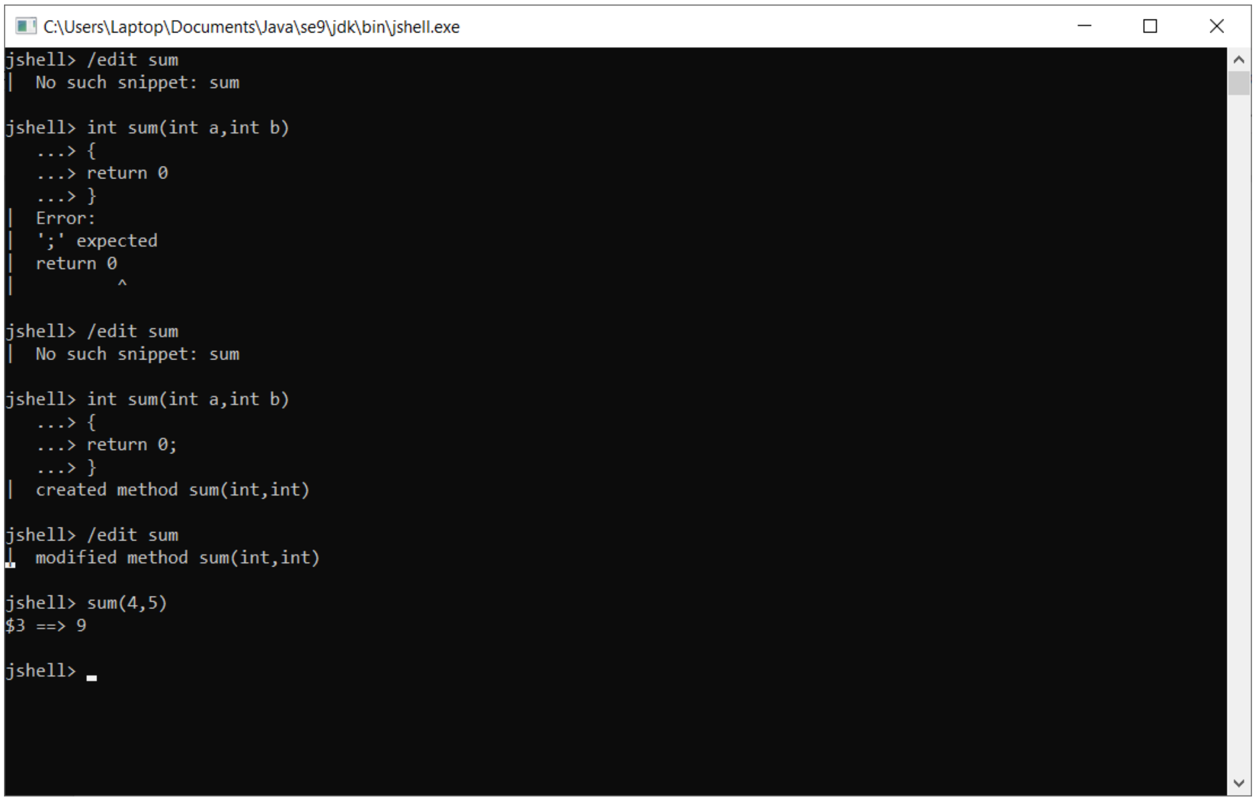
```
|  Welcome to JShell -- Version 9.0.4
|  For an introduction type: /help intro

jshell> /set editor "C:\\Program Files (x86)\\Notepad++\\notepad++.exe"
|  Editor set to: C:\Program Files (x86)\Notepad++\notepad++.exe

jshell> /edit sum
|  No such snippet: sum

jshell> int sum(int a,int b)
   ...> {
   ...> return 0
   ...> }
|  Error:
|  ';' expected
|  return 0
|          ^

jshell> /edit sum
|  No such snippet: sum

jshell> int sum(int a,int b)
   ...> {
   ...> return 0;
   ...> }
|  created method sum(int,int)

jshell> /edit sum
```

C:\Users\Laptop\AppData\Local\Temp\extedit9212993966598926502\18123889117275344474.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
change.log
new 1
new 2
18123889117275344474.java

```
int sum(int a,int b)
{
return 0;
}
```

Edit -> save -> turn off editor extenal -> goi ham Sum



```
jshell> /edit sum
|  No such snippet: sum

jshell> int sum(int a,int b)
   ...> {
   ...> return 0
   ...> }
|  Error:
|  ';' expected
|  return 0
|          ^

jshell> /edit sum
|  No such snippet: sum

jshell> int sum(int a,int b)
   ...> {
   ...> return 0;
   ...> }
|  created method sum(int,int)

jshell> /edit sum
|  modified method sum(int,int)

jshell> sum(4,5)
$3 ==> 9

jshell>
```

Platform and JVM Logging

JDK has improved logging in platform classes JDK Class and JVM components, through new API.

The API is meant to be used by the classes in the JDK, not by Application.

For your application code, you will continue using other logging Apis as before. It automatic

Use java.lang.System.Logger, which provides the logging API you can implement it. Then create class LoggerFinder from extends java.lang.System.LoggerFinder to autocreate instance Logger. View detail here:

https://sharecodefull.blogspot.com/2021/01/api-logging.html

Process API Updates

Prior to java 5, the only way to spawn a new process was to use the Runtime.getRuntime().exec(). Then in java 5, ProcessBuilder API was introduced which supported a cleaner way to of spawning new processes. Now java 9 is adding a new way of getting information about current and any spawned process.

To get information of any process, now you should use java.lang.ProcessHandle.Info interface.

Also use ProcessHandle.allProcesses() to get a stream of ProcessHandle of all processes available in system.

Collection Factory Of Method

Since java 9, you can create immutable collections such as immutable list, immutable set and immutable map using new factory methods **OF**.

Create immutable List

Static <E> List<E> of(E ... elements);

List immutable **cannot** Add, removed, replaced, sort. If call these methods exception UnsupportedOperationException. They do **not allow null** element. If given null appearance NullPointerException. They are **serializable** if all elements are serializable. The order of element in the list is the same as the **order of the provided** arguments, or of the elements in the provided array

Create immutable Set

Static <E> Set<E> .of(E… elements);

Set do not allow duplicate elements as well. The interation order of set element is unspecified and is subject to change.

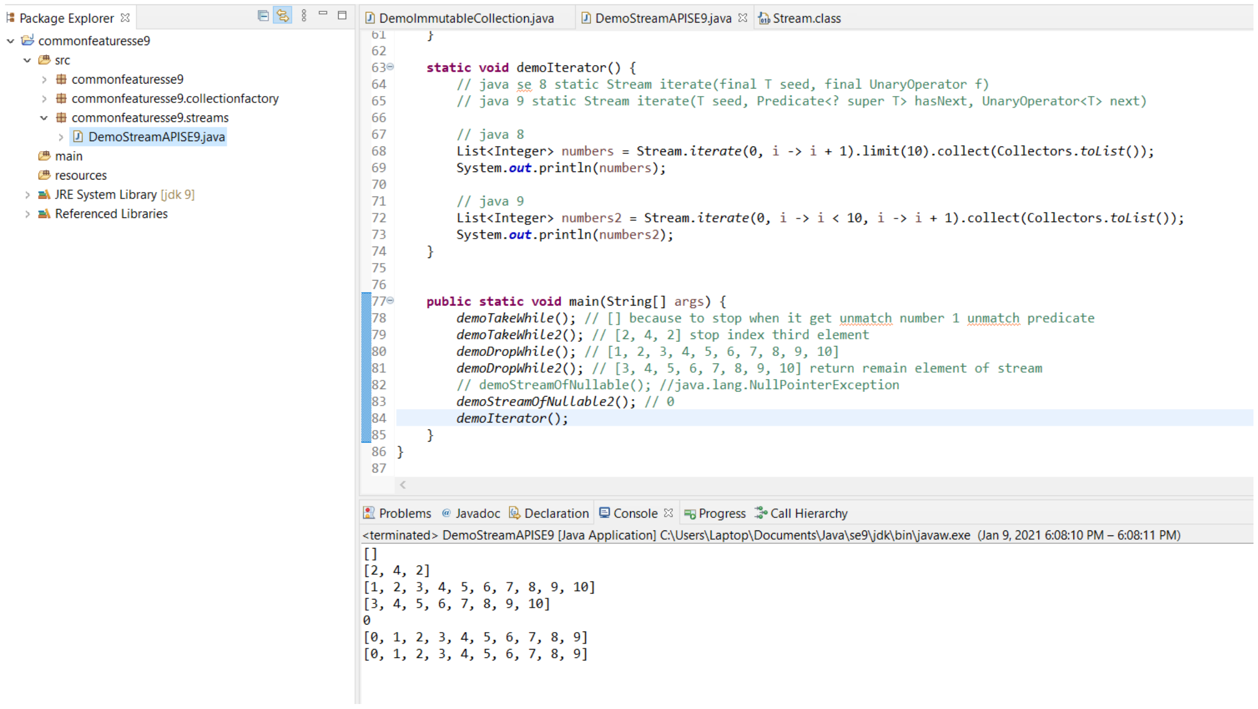
Create immutable Map

Static <K,V> Map<K, V>.of (K1,V2, k2,v2 , …)

static <K, V> Map<K, V> ofEntries(Entry<? extends K, ? extends V>... entries)

Stream API Improvements

In java 9, Stream API has improved and new methods added to the Stream interface. TakeWhile, dropWhile and ofNullable, and Overloaded iterate method are added to perform operations on stream elements



```java
    }

    static void demoIterator() {
        // java se 8 static Stream iterate(final T seed, final UnaryOperator f)
        // java 9 static Stream iterate(T seed, Predicate<? super T> hasNext, UnaryOperator<T> next)

        // java 8
        List<Integer> numbers = Stream.iterate(0, i -> i + 1).limit(10).collect(Collectors.toList());
        System.out.println(numbers);

        // java 9
        List<Integer> numbers2 = Stream.iterate(0, i -> i < 10, i -> i + 1).collect(Collectors.toList());
        System.out.println(numbers2);
    }


    public static void main(String[] args) {
        demoTakeWhile(); // [] because to stop when it get unmatch number 1 unmatch predicate
        demoTakeWhile2(); // [2, 4, 2] stop index third element
        demoDropWhile(); // [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
        demoDropWhile2(); // [3, 4, 5, 6, 7, 8, 9, 10] return remain element of stream
        // demoStreamOfNullable(); //java.lang.NullPointerException
        demoStreamOfNullable2(); // 0
        demoIterator();
    }
}
```

```
<terminated> DemoStreamAPISE9 [Java Application] C:\Users\Laptop\Documents\Java\se9\jdk\bin\javaw.exe  (Jan 9, 2021 6:08:10 PM – 6:08:11 PM)
[]
[2, 4, 2]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[3, 4, 5, 6, 7, 8, 9, 10]
0
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Multi-Release JRA Files

This enhancement is related to how you package application classes in jar files. Perviously, you had to package all classes into a jar file and drop in the classpath of the another application , which wish to use it.

Using multi-release feature, now a jar can contains different versions of a class- compatiable to different JDK releases. The information regarding diffrent versions of a class, and in which jdk version which class shall be picked up by class loaded is stored in MANIFEST.MF file. In this case, MANIFEST.MF file includes the entry Multi-Release : True in its main section.

Ex

```
JAR content root
  A.class
  B.class
  C.class
  D.class
  META-INF
     MANIFEST.MF
     versions
        9
           A.class
           B.class
        10
           A.class
```

Let’s assume that in JDK 10, A.class isupdated to leverage some java 10 features, then this jar file can be updated like this above example.

@Deprecated Tag Changes

From java 9, @Deprecated annotation will have to attributes: forRemoval and since.

forRemoval: indicates whether the annotated element is subject to removal in a future version.

Since: it returns the version which the annotated element became deprecated.

Reference

My Github source for java 9:

https://github.com/nguyenthinhit996/sharefullcode/tree/java/Learn%20Java/NewFeatureSE9

This section is non commercial mainly sharing and advance knowlage for java.This tutorials has referenced document from the list below if you has complain for license, i will remove all from internet. Thank you all everything.

https://www.javatpoint.com/java-versions

https://howtodoinjava.com/java9/java9-new-features-enhancements

https://www.baeldung.com/java-money-and-currency

https://dzone.com/articles/looking-java-9-money-and