KU LEUVEN

TECHNOLOGIECAMPUS DIEPENBEEK



New Features of C# 7.0 and Java 9

By Thomas Machiels

New features of C#7.0 and Java 9

- C# 7.0:
  - Release date: March 7, 2017
  - Data consumption
  - Code simplification
  - Performance

- Java 9:
  - Release date: September 21, 2017
  - Collection factory methods
  - Stream API improvements
  - Reactive Streams

New Features of C#7.0

- What is new?
  - Tuples
  - Out variables
  - Pattern Matching
  - Local functions
  - Literal improvements
  - Return references

C#7.0: Tuples

- In 2010: Tuple Class introduced
  - Not a good performance
  - Tuple elements have no name

```
private static Tuple<int, int> Divide(int x, int y)
        {
            int result = x / y;
            int reminder = x % y;
            return Tuple.Create(result, reminder);
        }
…

var division = Divide(120, 25);
int result = division.Item1;
int reminder = division.Item2;
```

C#7.0: Tuples

- In C# 7.0:
  - Inline declarations
  - Named tuple elements
  - Better perfomance
  - Max 8 elements

```
private static (int result, int reminder) Divide2(int x, int y)
{
        int result = x / y;
        int reminder = x % y;
        return (result, reminder);
}
…
var division2 = Divide2(120, 25);
int result2 = division2.result;
int reminder2 = division2.reminder;
```

C#7.0: Tuples

- Example:

```
(string first, string middle, string last) LookupName(long id)
{
    ... // retrieve first, middle and last from data storage
    return (first, middle, last); // tuple literal
}

…

var names = LookupName(id);
WriteLine($"found {names.first} {names.last}.");

var name = (first: "Jan", middle:"Piet", last:"Joris");
```

C#7.0: Tuples

- Example:

```
public (double lat, double lng) GetLatLng(string address) { ... }

var ll = GetLatLng("some address");
Console.WriteLine($"Lat: {ll.lat}, Long: {ll.lng}");


var ll = (lat: 0, lng: 0);
```

C#7.0: Tuples

- Deconstruction

```
(string first, string middle, string last) = LookupName(id1);

(var first, var middle, var last) = LookupName(id1);

var (first, middle, last) = LookupName(id1);

(first, middle, last) = LookupName(id1);
```

C#7.0: Out Variables

- Previous version:

```
public void PrintCoordinates(Point p)
{
    int x, y; // have to "predeclare"
    p.GetCoordinates(out x, out y);
    WriteLine($"({x}, {y})");
}
```

- In C#7.0:

```
public void PrintCoordinates(Point p)
{
    p.GetCoordinates(out int x, out int y);
    WriteLine($"({x}, {y})");
}
```

C#7.0: Out Variables

- Using var:

```
public void PrintCoordinates(Point p)
{
    p.GetCoordinates(out var x, out var y);
    WriteLine($"({x}, {y})");
}
```

- In the future: WildCards Allowed with “ * ”

```
public void PrintCoordinates(Point p)
{
    p.GetCoordinates(out int x, out *); // I only care about x
    WriteLine($"({x})");
}
```

C#7.0: Pattern Matching

- From functional languages
- Enhancing two existing language constructs:
  - Is expression: can now have patterns
  - Switch statements: can match on patterns

- Examples
  - Constant Patterns
  - Type Patterns
  - Var Patterns
  - Tuple patterns not yet supported!

C#7.0: Pattern Matching

- Is-Expression with patterns

```
public void PrintStars(object o)
{
        if (o is null) return;
        if (o is int i) {
            Console.WriteLine(new string('*', i));
         }
}

if (o is int i || (o is string s && int.TryParse(s, out i)) {
    /* use i */
}


string filePath = size is int x &&
                        x > 10 ? "C:\Archive" : "C:\TinyFiles";
```

C#7.0: Pattern Matching

- Difference “is” and “as” keyword:
  - “is” check returns a boolean
  - “as” check returns a null value

```
if(obj is string)
{
      ...
}

string str = obj as string;
if(str != null)
{
      ...
}
```

C#7.0: Pattern Matching

- Switch statements with patterns

- Switch on:
  - Any Type
  - Patterns in case clauses
  - Additional conditions in case clauses

C#7.0: Pattern Matching

```
switch (shape)
{
    case Circle c:
        WriteLine($"circle with radius {c.Radius}");
        break;
    case Rectangle s when(s.Length == s.Height):
        WriteLine($"{s.Length} x {s.Height} square");
        break;
    case Rectangle r:
        WriteLine($"{r.Length} x {r.Height} rectangle");
        break;
    default:
        WriteLine("<unknown shape>");
        break;
    case null:
        throw new ArgumentNullException(nameof(shape));
}
```

C#7.0: Pattern Matching

```
public void PrintShape(Shape shape)
{
    Circle c = shape as Circle;
    if (c != null)
    {
        WriteLine($"circle with radius {c.Radius}");
        return;
    }

    Rectangle s = shape as Rectangle;
    if (s != null)
    {
        if (s.Length == s.Height)
        {
            WriteLine($"{s.Length} x {s.Height} square");
            return;
        }
        else
        {
            WriteLine($"{r.Length} x {r.Height} rectangle");
            return;
        }
    }
    WriteLine("<unknown shape>");
}
```

C#7.0: Local Functions

- Helper functions inside functions
- Can use variables declared in their scope
- Also use variables declared in encompassing scope

```
static void ReduceMagicalEffects(this IWeapon weapon, int timePassed)
{
    double decayRate = CalculateDecayRate();
    double GetRemainingEffect(double currentEffect) =>
        currentEffect * Math.Pow(decayRate, timePassed);

    weapon.FireEffect = GetRemainingEffect(weapon.FireEffect);
    weapon.IceEffect = GetRemainingEffect(weapon.IceEffect);
    weapon.LightningEffect = GetRemainingEffect(weapon.LightningEffect);
}
```

C#7.0: Local Functions

```
public int Fibonacci(int x)
{
    if (x < 0) throw new ArgumentException();
    return Fib(x).current;

    (int current, int previous) Fib(int i)
    {
        if (i == 0) return (1, 0);
        var (p, pp) = Fib(i - 1);
        return (p + pp, p);
    }
}
```

C#7.0: Literal Improvements

- “ _ “ allowed as digit seperator:

```
var d = 100_000_000_000;
var x = 0xAB_CD_EF;
```

- Or as binary literal:

```
var b = 0b1010_1011_1100_1101_1110_1111;
```

C#7.0: Ref Return

- Preventing unnecessary copying of data

```
public ref int Find(int number, int[] numbers)
{
    for (int i = 0; i < numbers.Length; i++)
    {
        if (numbers[i] == number)
        {
            return ref numbers[i]; // return the storage location
        }
    }
}

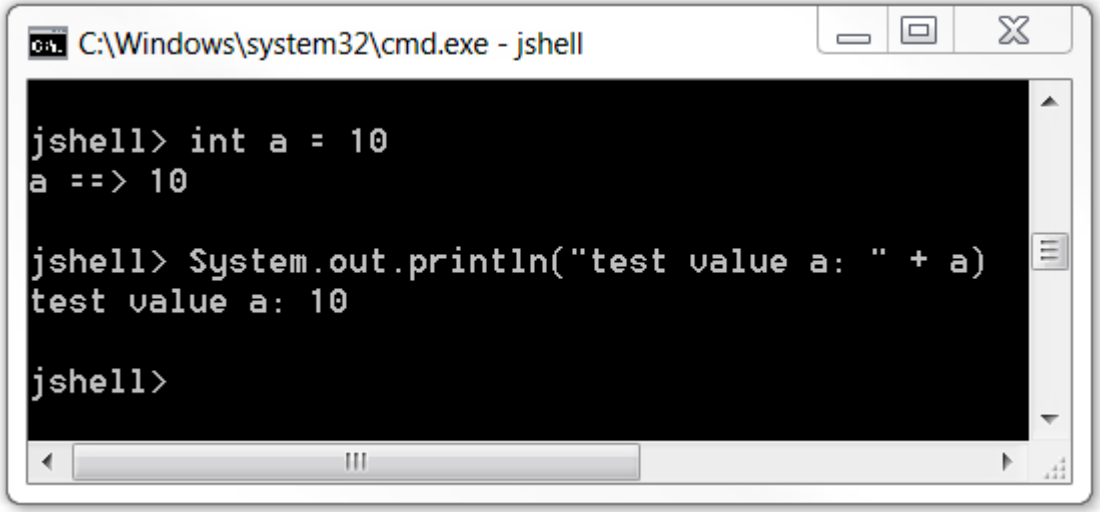
int[] array = { 1, 15, -39, 0, 7, 14, -12 };
ref int place = ref Find(7, array); //aliases place 7 in the array
place = 9; // replaces 7 with 9 in the array
WriteLine(array[4]); // prints 9
```

New features of Java 9

- Java 9 JShell
- Factory methods for immutable collections
- Private method interfaces
- Java 9 Module System
- Try with resources improvements
- Completable Future API improvements
- Reactive Streams
- Diamond operator for anonymous inner class
- Optional Class improvements
- Stream API improvements

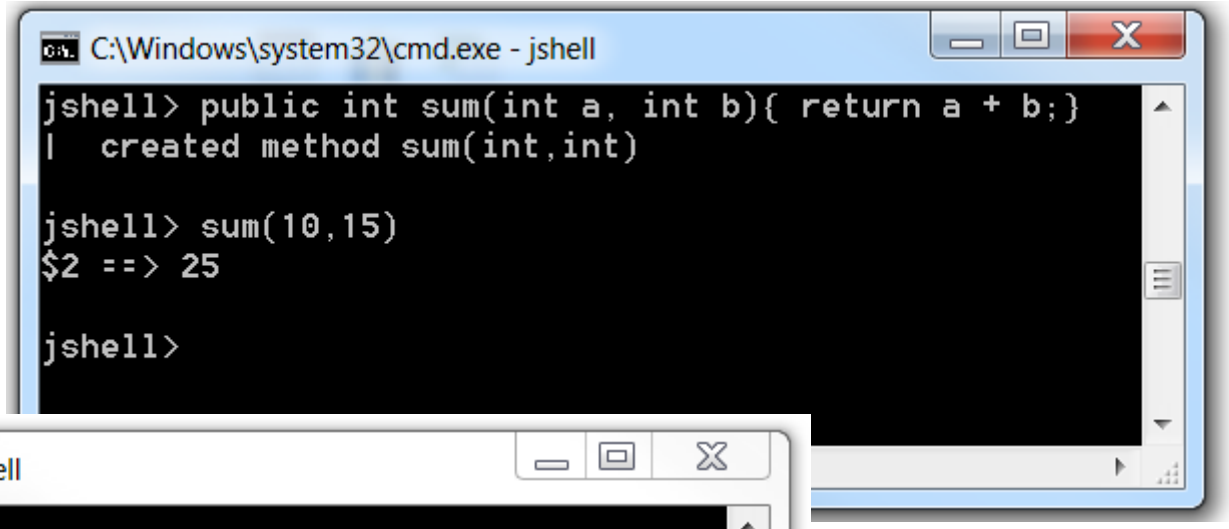
Java 9: jshell

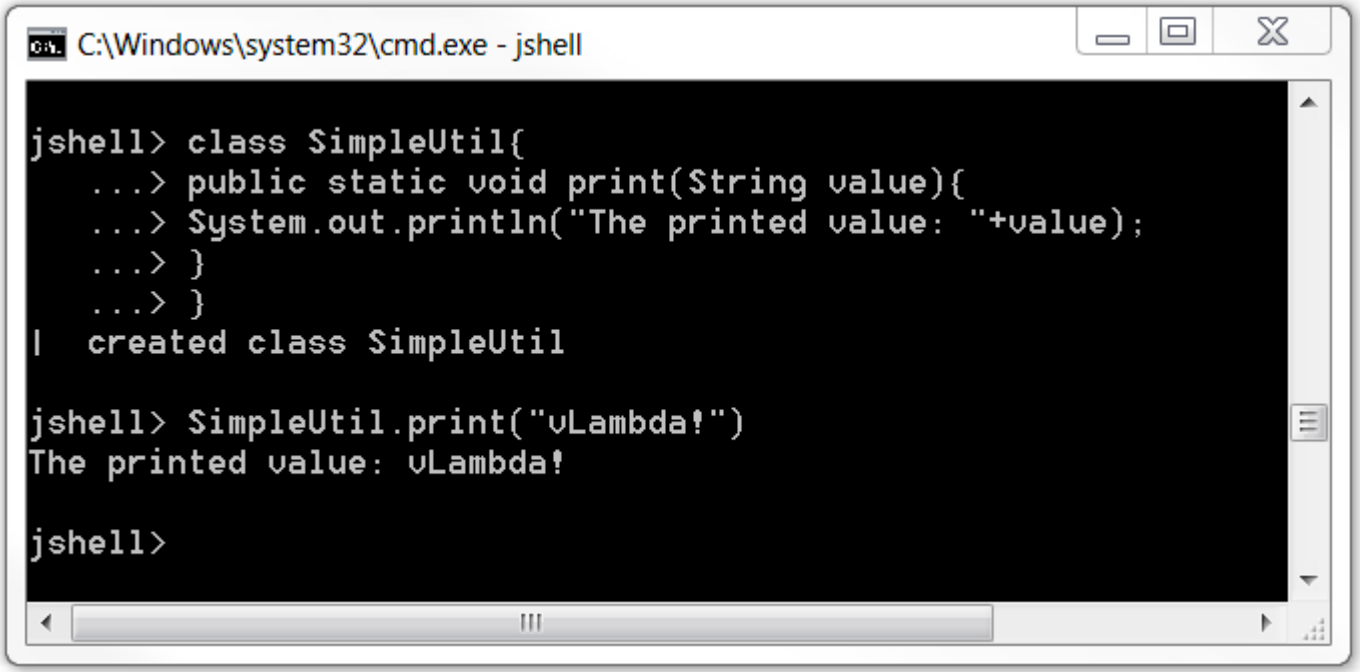
- Java Shell
- REPL (Read Evaluate Print Loop)
- Test and execute Java code



Java 9: jshell

- Functions
- Classes
- And much more





KU LEUVEN

Java 9:
Factory methods for Immutable collections

- Problem :
  - Creating small immutable collections is verbose

```
Set<String> set = new HashSet<>();
set.add("a");
set.add("b");
set.add("c");
set = Collections.unmodifiableSet(set);
```

Java 9:
Factory methods for Immutable collections

- Static methods added to List, Set and Map interfaces
  - Create compact, immutable instances
  - 0 – 10 element overloaded versions
  - Varargs version for arbitrary number of elements

```
Set<String> set = Set.of("a","b","c");
List<String> list = List.of("a","b","c");
Map<String, String> map = Map.of("k1","v1","k2","v2");
Map<String, String> map =
       Map.ofEntries(entry("k1","v1"),entry("k2","v2"));
```

Java 9: Private method interfaces

- In Java 7, Simple interface
- In Java 8, Default and Static methods
- In Java 9, Private methods
- To avoid redundant code and more re-usability

```
public interface Card{

    private Long createCardId(){
        //Method implementation goes here.
    }

    private static void displayCardDetails(){
        //Method implementation goes here.
    }
}
```

KU LEUVEN

Java 9: Private method interfaces

- Java 7:
  - Constant variables
  - Abstract methods
  - No method implementations

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

    void logInfo(String message);
    void logWarn(String message);
    void logError(String message);
    void logFatal(String message);
}
```

Java 9: Private method interfaces

- Java 8:
  - Default methods and Static methods

```
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){
        Step1: Connect to DataStore;
        Step2: Log Info Message;
        Step3: Close the DataStore connection;
    }

    default void logWarn(String message){
        Step1: Connect to DataStore
        Step2: Log Warn Message
        Step3: Close the DataStore connection
    }
    ...
}
```

KU LEUVEN

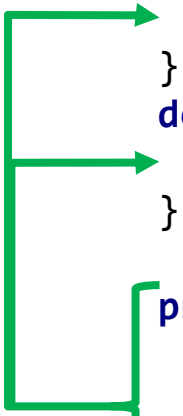
Java 9: Private method interfaces

- Java 9:
  - Private methods and Private static methods

```
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");
    }

    private void log(String message, String msgPrefix){
        Step1: Connect to DataStore;
        Step2: Log Message with Prefix and styles etc.;
        Step3: Close the DataStore connection;
    }
}
```



Java 9: Module System

- Jigsaw Project
- Problems with current Java System
- Advantages of the Module System
- What is Java 9 Module?
- Java 8 and 9 comparison

Java 9: Module System

The Jigsaw project as new concept of java module system

Main goals:

- Modular JDK
- Modular source code
- Java platform module system
- Jlink

Java 9: Module System

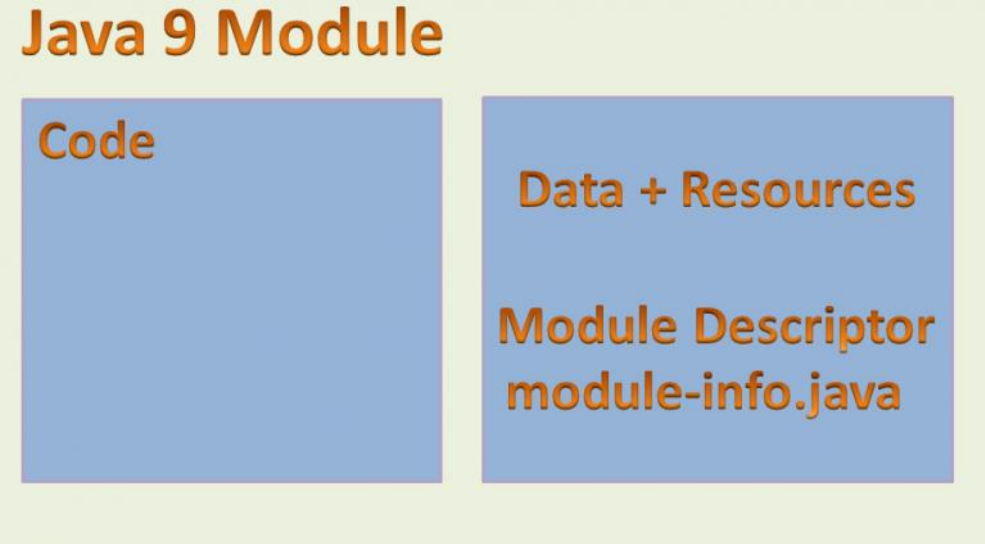
- Problems with the current system:
  - JDK and JAR files to big for small devices
  - No support for better performance
  - Hard to test and maintain
  - Applications is too big

Java 9: Module System

- Advantages using Java 9 modules:
  - Division of JDK, JRE and JAR into smaller modules
  - Ease of testing and maintainability
  - Supports better performance
  - Smaller applications

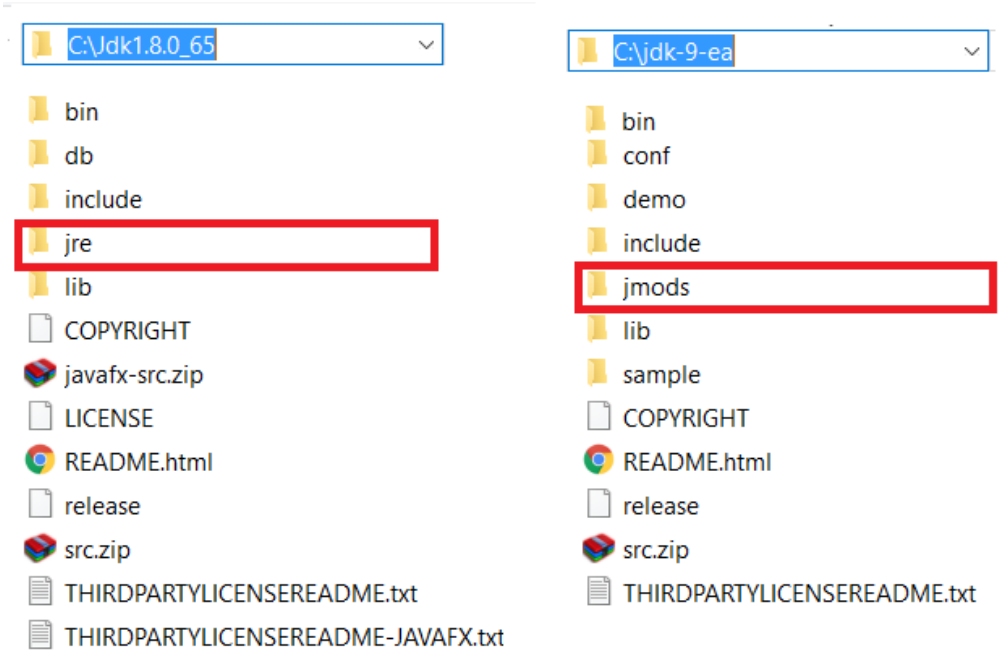
Java 9: Module System

- What is Java 9 Module:
  - Self describing collection of code, data and some resources
  - Single responsibility per module
  - Support of Modular Programming



Java 9: Module System

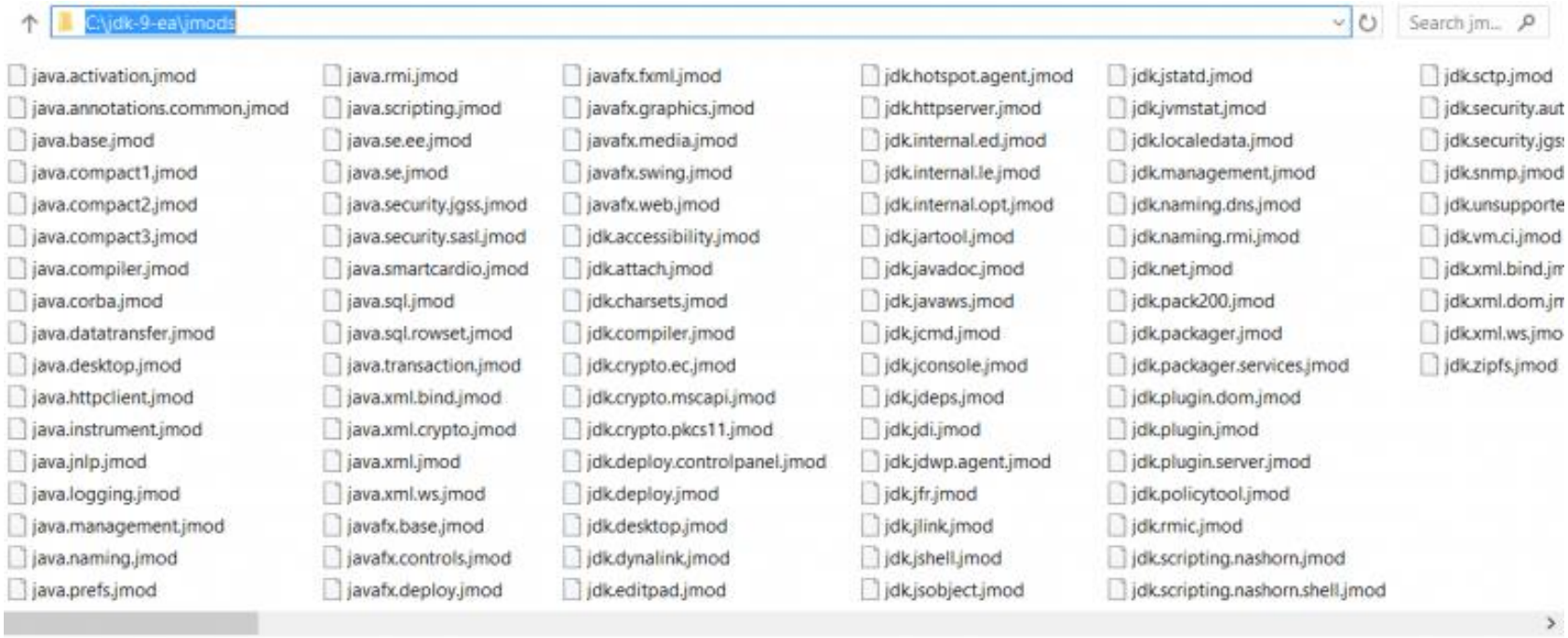
- Java 8 and 9 comparison:



JDK 8 Vs JDK 9

Java 9: Module System

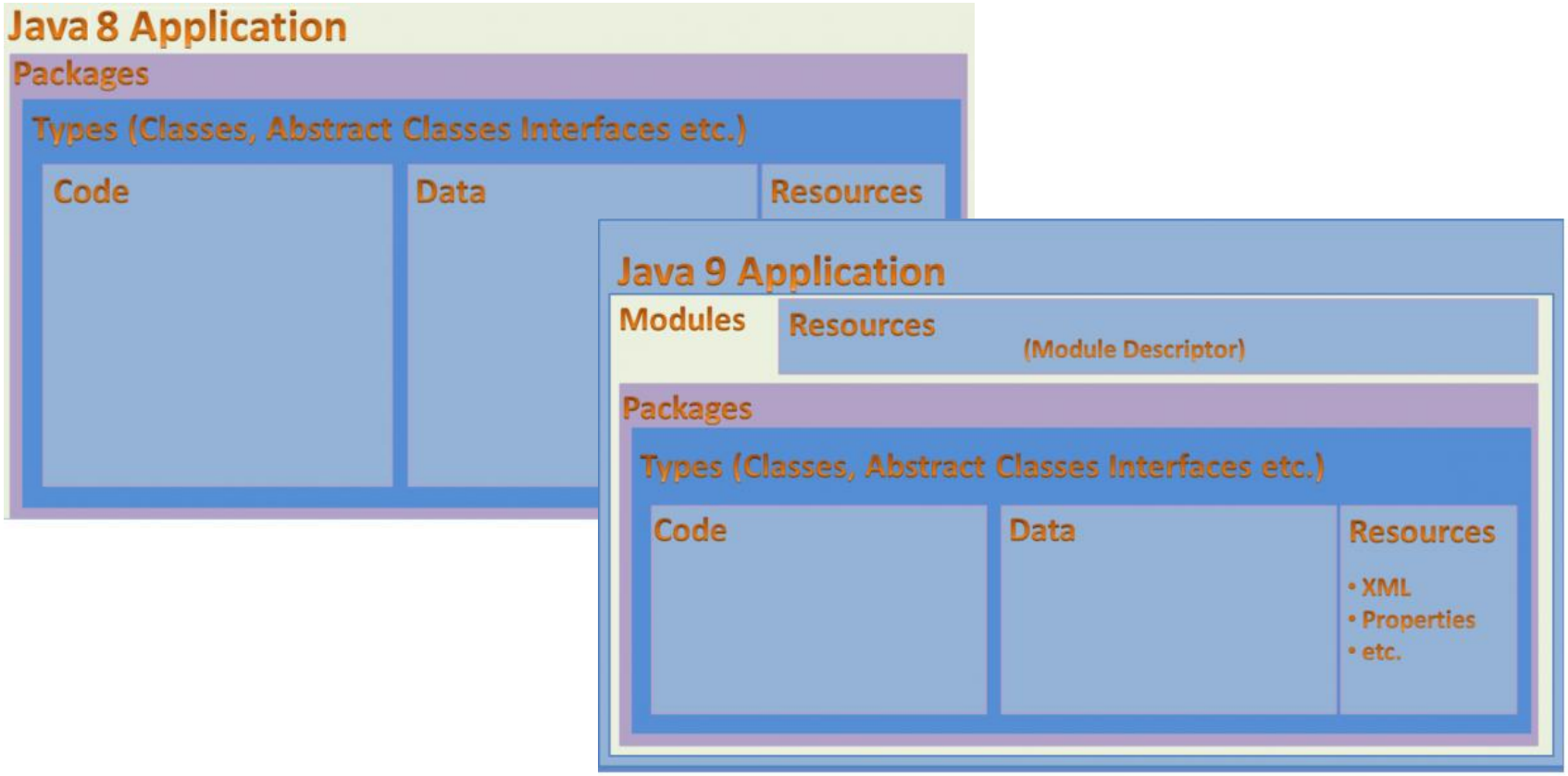
- Java 8 and 9 comparison:



KU LEUVEN

Java 9: Module System

- Java 8 and 9 comparison:



Java 9: Try with resources improvements

- Added resource references

Java 8:

```
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:

```
void testARM_With_Java9() throws IOException {
    BufferedReader reader1 = new BufferedReader(new
                                        FileReader("journaldev.txt"));
    try (reader1) {
        System.out.println(reader1.readLine());
    }
}
```

Java 9: CompletableFuture API improvements

- Added support for:
  - Delays
  - Timeouts

```
Executor exe = CompletableFuture
                        .delayedExecutor(50L, TimeUnit.SECONDS);
```

Java 9: Reactive Streams

- In Java 8 already available through Rx.Java and Akka
- Available in Java 9: concurrent Flow API



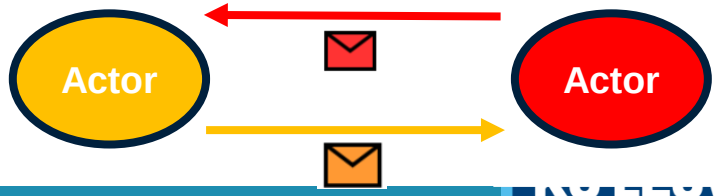
RxJava



- Observable/Observer



- Actors
- Message system



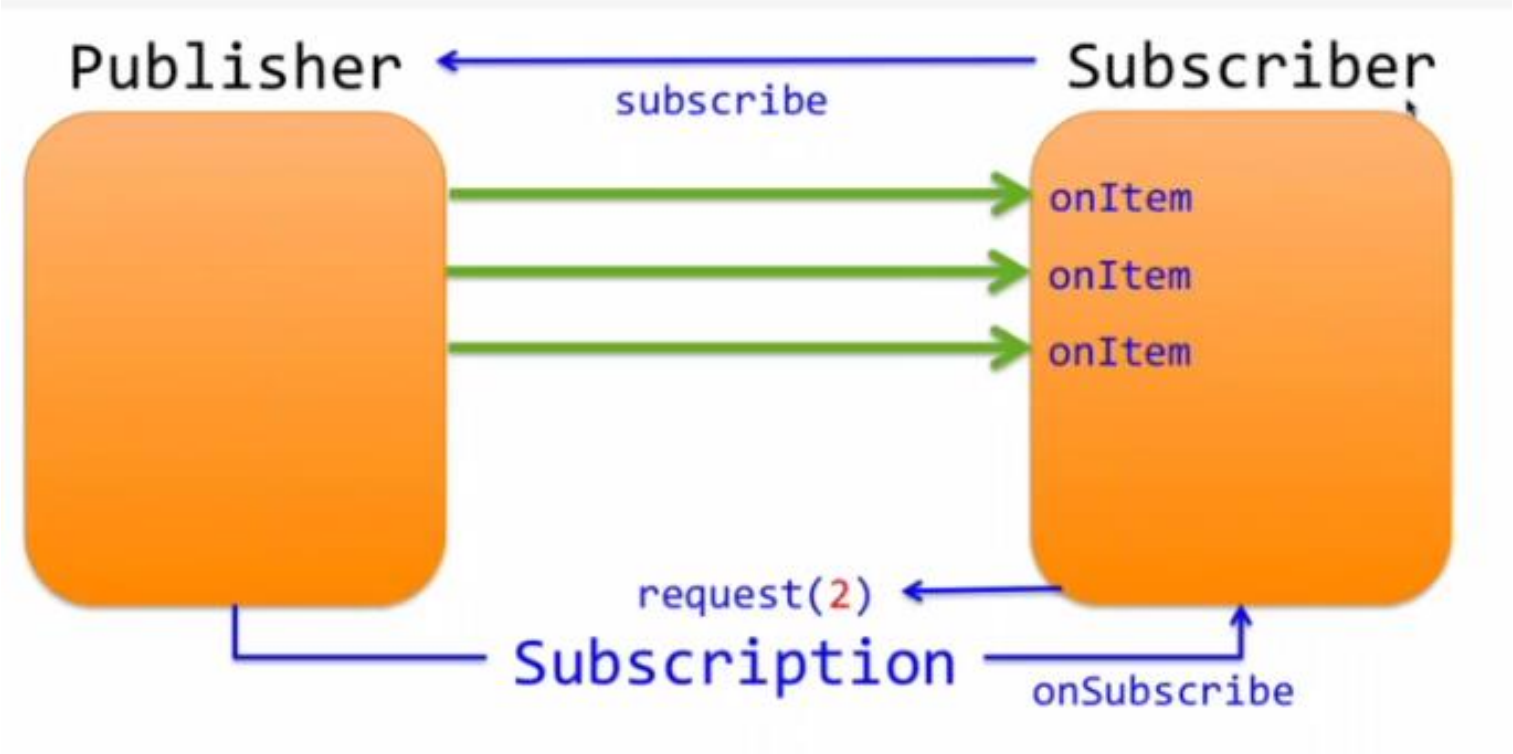
Java 9: Reactive Streams

- Main components:
  - Publisher
    - Producer of items to be received by Subscribers
  - Subscriber
    - Receiver of messages
  - Processor
    - Chaining Publisher and Subscriber
  - Subscription
    - Message control linking Publisher and Subscriber

Java 9: Reactive Streams

- Main components:
  - Publisher
    - subscribe(Subscriber)
  - Subscriber
    - onSubscribe(Subscription)
    - onNext(T item)
    - onComplete() / onError(Throwable)
  - Subscription
    - request(long)
    - cancel()

Java 9: Reactive Streams



Java 9:
Diamond operator for anonymous inner class

- Java 8: anonymous inner class

```
public abstract class
MyHandler<T> {
    abstract void handle();
}

MyHandler<Integer> intHandler = new MyHandler<Integer>() {

    public void handle() {
        // handling code...
    }
};

MyHandler<Integer> intHandler = new MyHandler<>(10) {
        // Anonymous Class
};
MyHandler<?> handler = new MyHandler<>("One hundred") {
        // Anonymous Class
};
```

Java 9:
Diamond operator for anonymous inner class

- Java 9: improvements for diamond operator

```
MyHandler<Integer> intHandler = new MyHandler<>(1) {

    @Override
    public void handle() {
        // handling code...
    }
};

MyHandler<? extends Integer> intHandler1 = new MyHandler<>(10) {

    @Override
    void handle() {
        // handling code...
    }
};

MyHandler<?> handler = new MyHandler<>("One hundred") {

    @Override
    void handle() {
        // handling code...
    }
};
```

Java 9: Optional Class Improvements

- Optional Stream
- IfPresentOrElse()
- Or()

```
Stream<Optional> emp = getEmployee(id);
Stream empStream = emp.flatMap(Optional::stream);
```

```
public void ifPresentOrElse(Consumerl<? super Tl> action, Runnable emptyAction);
```

```
public Optional<T> or(Supplier<? extends Optional<? extends T>> supplier);
```

Java 9: Optional Class Improvements

- IfPresentOrElse():

```
Optional<Integer> opt1 = Optional.of(4);

opt1.ifPresentOrElse(x -> System.out.println("Result found: " + x),
                                    () -> System.out.println("Not Found."));
```

Result found: 4

```
Optional<Integer> opt2 = Optional.empty();

opt1.ifPresentOrElse(x -> System.out.println("Result found: " + x),
                                    () -> System.out.println("Not Found."));
```

Not Found.

Java 9: Optional Class Improvements

- Or():

```
Client client = findClient(companyId);

if (client == null) {
    client = lookupCompanyDetails(companyId);
}
// client could still be null


Optional<Client> client =
        findClient(companyId)
                .or(() -> lookupCompanyDetails(companyId));
```

Java 9: Stream API Improvements

- Introduced in Java 8
- In Java 9: four added functions
  - takeWhile ()
  - dropWhile()
  - iterate()
  - ofNullable()



Java 9: Stream API Improvements

- takeWhile() method

```
jshell> Stream<Integer> stream =
Stream.of(1,2,3,4,5,6,7,8,9,10)
stream ==> java.util.stream.ReferencePipeline$Head@55d56113

jshell> stream.takeWhile(x -> x < 4).forEach(a ->
System.out.println(a))
1
2
3
```

Java 9: Stream API Improvements

- takeWhile() method

```
jshell> Stream<Integer> stream =
Stream.of(1,2,4,5,3,6,7,8,9,10)
stream ==> java.util.stream.ReferencePipeline$Head@55d56113

jshell> stream.takeWhile(x -> x < 4).forEach(a ->
System.out.println(a))
1
2
```

Java 9: Stream API Improvements

- dropWhile() method

```
jshell> Stream<Integer> stream =
Stream.of(1,2,3,4,5,6,7,8,9,10)
stream ==> java.util.stream.ReferencePipeline$Head@55d56113

jshell> stream.dropWhile(x -> x < 4).forEach(a ->
System.out.println(a))
4
5
6
7
8
9
10
```

Java 9: Stream API Improvements

- iterate() method

Java 8:

```
jshell> IntStream.iterate(2, x -> x * x).filter(x -> x <
20).forEach(System.out::println)
2
4
16
```

Java 9:

```
jshell> IntStream.iterate(2, x -> x < 20, x -> x *
x).forEach(System.out::println)
2
4
16
```

Java 9: Stream API Improvements

- ofNullable() method

```
jshell> Stream<Integer> s = Stream.ofNullable(1)
s ==> java.util.stream.ReferencePipeline$Head@1e965684

jshell> s.forEach(System.out::println)
1

jshell> Stream<Integer> s = Stream.ofNullable(null)
s ==> java.util.stream.ReferencePipeline$Head@3b088d51

jshell> s.forEach(System.out::println)

jshell>
```

Java 9: Enhanced @deprecated annotation

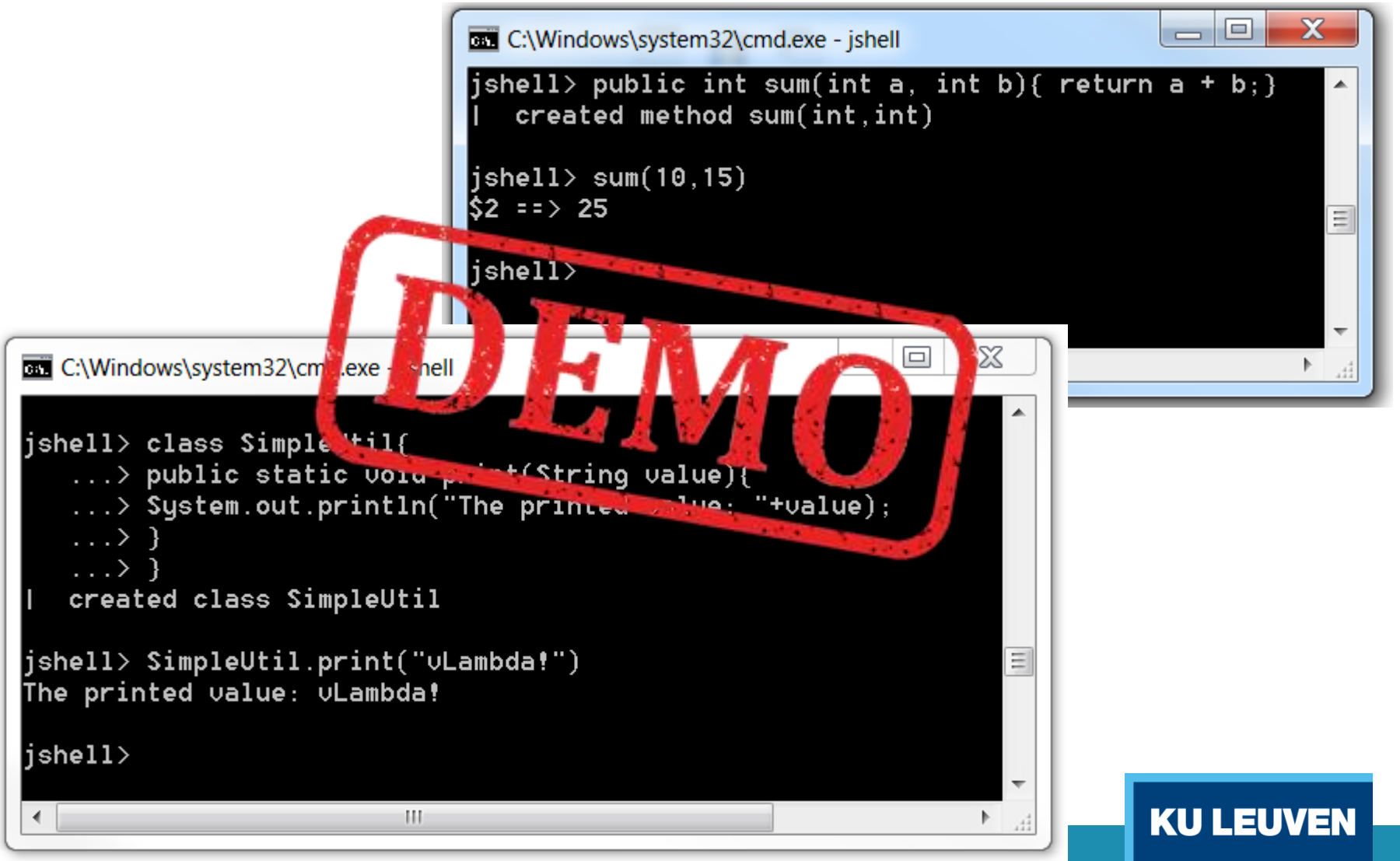
- In Java 8: only notation for a Java API
- In Java 9: enhanced
  - More information: forRemoval, since
  - Tool to analyze the depreciated API

Summary

- C# 7.0:
  - Data consumption
  - Code simplification
  - Performance

- Java 9:
  - Jshell
  - Reactive Streams
  - Optional improvements
  - Stream Improvements

Java 9: Small JShell Demo



KU LEUVEN