Becoming one of the first Java 17 certified programmers (and learning new features)

Jeanne Boyarsky

Sept 15, 2021

KCDC

speakerdeck.com/boyarsky

At end of session

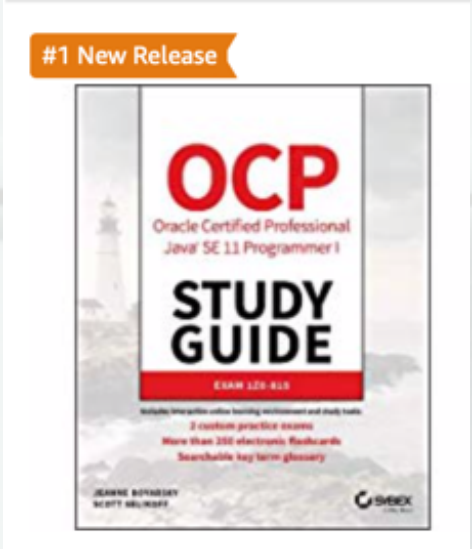
1. https://speakerdeck.com/boyarsky
2. https://github.com/boyarsky/2021-kcdc-java17

About Me

- Java Champion
- Author
- Developer at NYC bank for 19+ years
- FIRST Robotics Mentor

@jeanneboyarsky

Pause for a Commercial



Java certs

- Java 8
- Java 11
- Java 17 next

Book giveaway at end!

@jeanneboyarsky

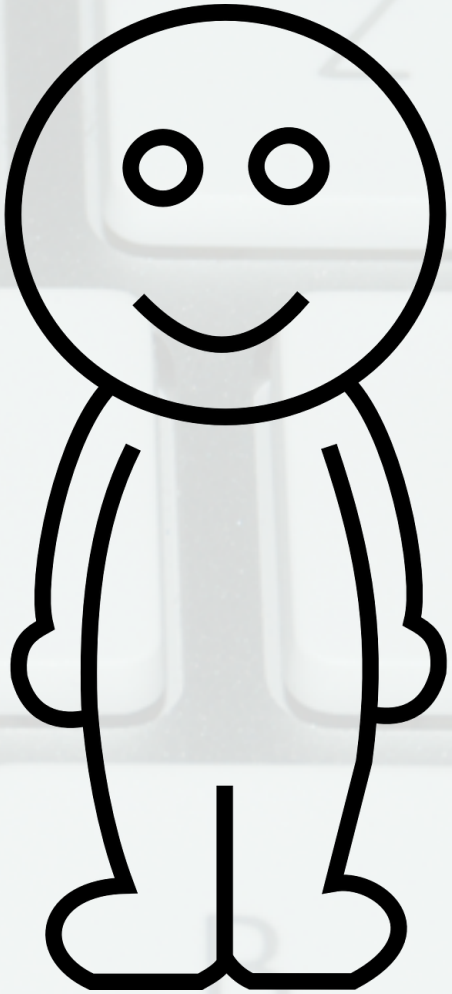
Another Commercial

@jeanneboyarsky

Disclaimer

- A bit of the material is from my books.
- Some of the material in this presentation may appear in our upcoming certification books.
- Oracle can announce anything (I don't work for Oracle.)

Introductions



@jeanneboyarsky

Agenda

Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

Module Flow

- Review lab from previous module
- Lecture/review
- Hands on exercises
- 10 minute break

This means if a colleague needs to call you, the last 15-20 minutes of each hour is best.

Required Software for Lab

Option 1

- Java 17 - http://jdk.java.net/17/
- Text editor of your choice
- No IDE!

or…

Required Software for Lab

Option 2

- https://www.compilejava.net

```
System.out.println(System.getProperty(“java.version"));
```

- 17-ea (on Sunday night)

- Can only do some labs
  - Module 3 - sealed classes preview (in 16)
  - Module 5 and 6 - no IO
  - Module 7 - no JPMS

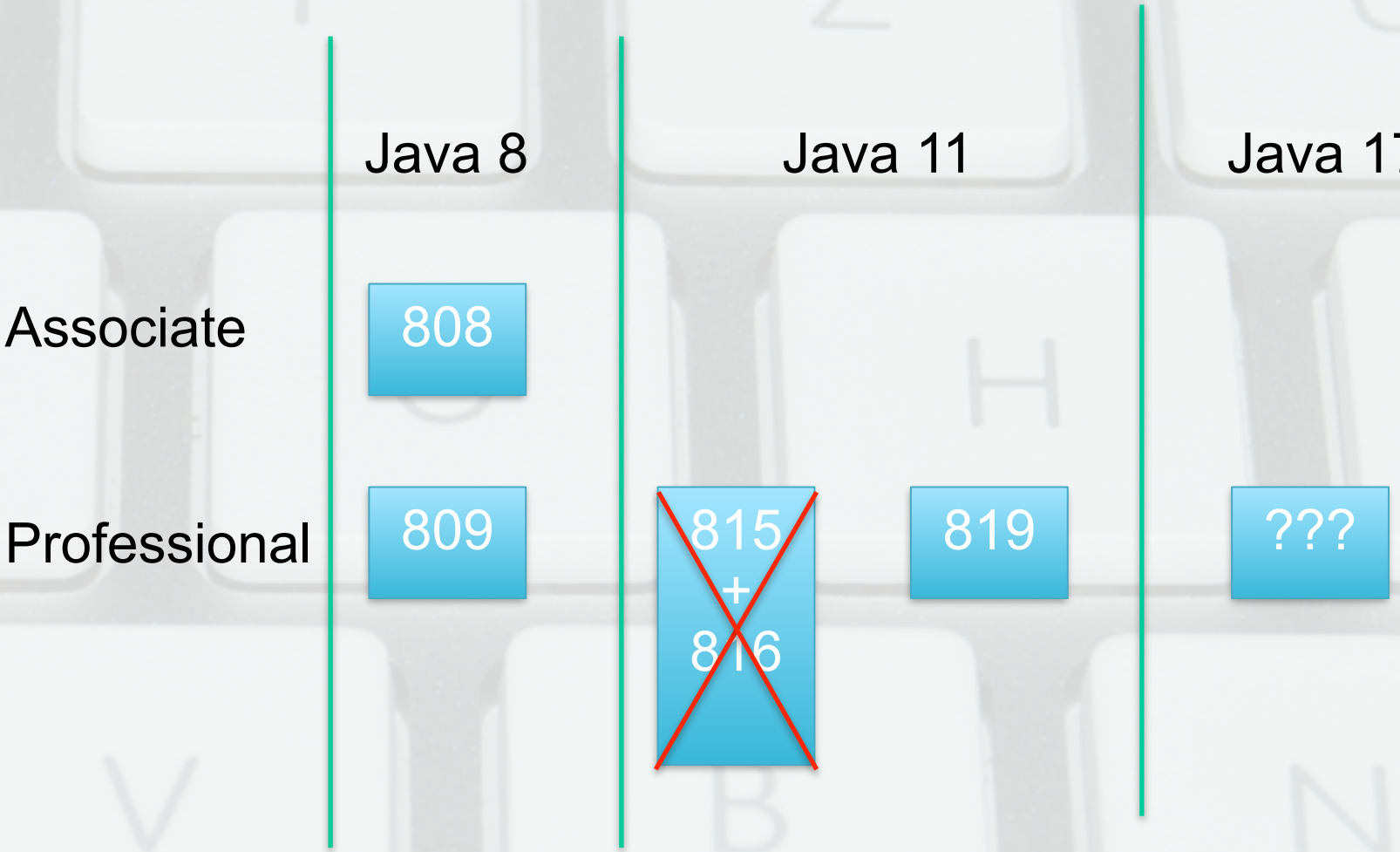
Optional aliases

```
alias javac17=/Library/Java/
JavaVirtualMachines/jdk-17.X.jdk/
Contents/Home/bin/javac

alias java17=/Library/Java/
JavaVirtualMachines/jdk-17.X.jdk/
Contents/Home/bin/java

alias jshell17=/Library/Java/
JavaVirtualMachines/jdk-17.X.jdk/
Contents/Home/bin/jshell
```

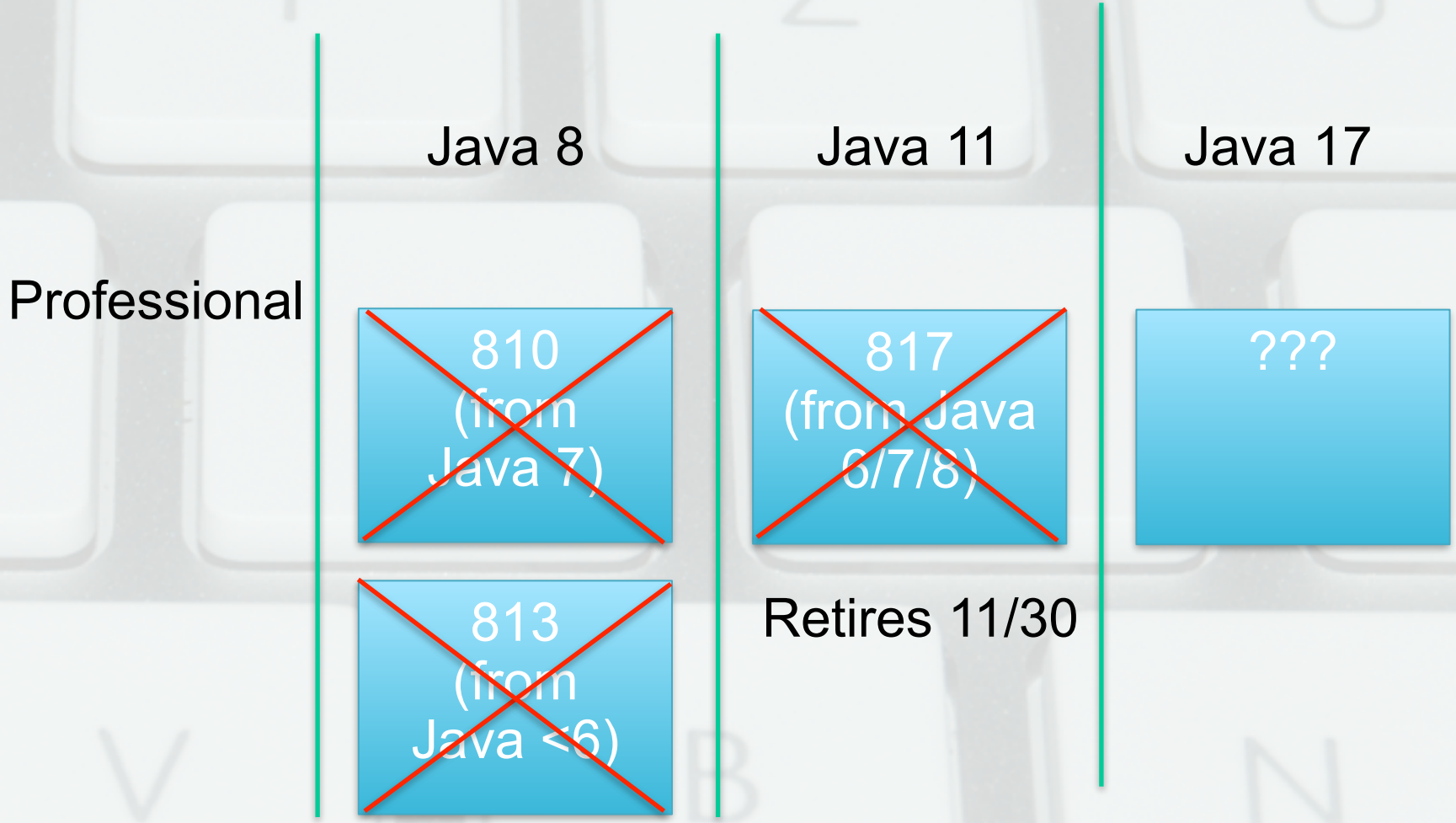
Agenda

Module	Topics
1 	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

If new to the cert



If hold a cert



Features

Feature	Preview	Final
Text Blocks	13, 14	15
APIs		various

What's wrong?

15

```
String old = "kcdc,Kansas City,"session,workshop"
        + "meetup,Various,lecture\n";
```

Doesn't compile: missing \

Missing quote on line 1

Missing line break

Compare

15

```
String old = "kcdc,Kansas City,\"session,workshop\"\n"
        + “meetup,Various,lecture\n";

String textBlock = """
        kcdc,Kansas City,"session,workshop"
        meetup,Various,lecture
        """;
```

Easier to read and code!

Text Block Syntax

15

```
String textBlock = """
    kcdc,Kansas City,"session,workshop"
    meetup,Various,lecture
    """;
```

start block

incidental whitespace

end block

Essential Whitespace

15

```
String textBlock = """
        <session>
            <speaker>
                Jeanne Boyarsky
            </speaker>
        </session>
        """;
```

incidental whitespace

essential whitespace

Ending lines

15

```
String textBlock = """
        <session>
            <speaker>
                Jeanne Boyarsky     \s
            </speaker>
            <title>
                Becoming one of the first Java 17 \
                certified programmers \
                (and learning new features)
            </title>
        </session>
        """;
```

new escape character
keeps trailing whitespace

tab

continue on next line
without a line break

New lines


```
String textBlock = """
        <session>\n
            <speaker>
                Jeanne\nBoyarsky
            </speaker>
        </session>""";
```

Two new lines
(explicit and implicit)

One new line (explicit)

no line break at end

Escaping Three Quotes

15

```
String textBlock = """
        better \"""
        but can do \"\"\"
        """;
```

Indent

12

```
String option1 = "    a\n   b\n   c\n";
String option2 = "a\nb\nc\n".indent(3);
String option3 = """
         a
         b
         c
         """.indent(3);
String option4 = """
             a
             b
             c
         """;
```

Which do you like best?

Also normalizes (bye \r)

Transform

12

```
String option1 = “chiefs".transform(
    s -> s.toUpperCase());

String option2 = """
         chiefs
         """.transform(s -> s.toUpperCase());
```

Which do you like best?

Strip Indent

15

Method	From beginning	From end	Per line
strip()	Leading	Trailing	No
stripIndent()	Incidental	Incidental	Yes
stripLeading()	Leading	n/a	No
stripTrailing()	n/a	Trailing	No

Module 1 - Question 1

Which is true about this text block?

```
String sql = """
             select *
             from mytable
             where weather = 'snow';
             """;
```

A. Has incidental whitespace

B. Has essential whitespace

C. Both A and B

D. Does not compile

Module 1 - Question 2

Which is true about this text block?

```
String sql = """select *
        from mytable \
        where weather = 'snow';
        """;
```

A. Has incidental whitespace

B. Has essential whitespace

C. Both A and B

D. Does not compile

Module 1 - Question 3

Which is true about this text block?

```
    String sql = """
                 select *     \s
                 from mytable      \s
                 where weather = 'snow';
""";
```

A. Has incidental whitespace

B. Has essential whitespace

C. Both A and B

D. Does not compile

Module 1 - Question 4

Which is true about this text block?

```
String sql = """select * from mytable;""";
```

A. Has incidental whitespace

B. Has essential whitespace

C. Both A and B

D. Does not compile

Module 1 - Question 5

How many lines does this print out?

```
String sql = """
        select * \n
        from mytable;
        """;
System.out.print(sql);
```

A. 2

B. 3

C. 4

D. Does not compile

Module 1 - Question 6

How many lines does this print out?

```
String sql = """
         select * \
         from mytable;""".stripIndent();
System.out.print(sql);
```

A. 1

B. 2

C. 3

D. Does not compile

Module 1 - Question 7

How many lines does this print out?

```
String sql = """
             select *     \s
             from mytable;
         ".stripIndent();
System.out.print(sql);
```

A. 1

B. 2

C. 3

D. Does not compile

Module 1 - Question 8

How many whitespace characters are removed by strip()?

```
String sql = """
             select "name"
             from mytable;
           """;
```

A. 1

B. 2

C. 3

D. Does not compile

Module 1 - Question 9

How many whitespace characters are removed by stripIndent()?

```
String sql = """
             select "name"
             from mytable;
           """;
```

A. 0

B. 1

C. 2

D. 3

Module 1 - Question 10

How many escapes can be removed without changing the behavior?

```
String sql = """
             select \"name\"
             from mytable \
             where value = '\"\"'
           """;
```

A. 2

B. 3

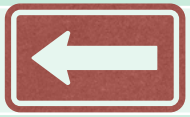
C. 4

D. Does not compile

Lab & Break

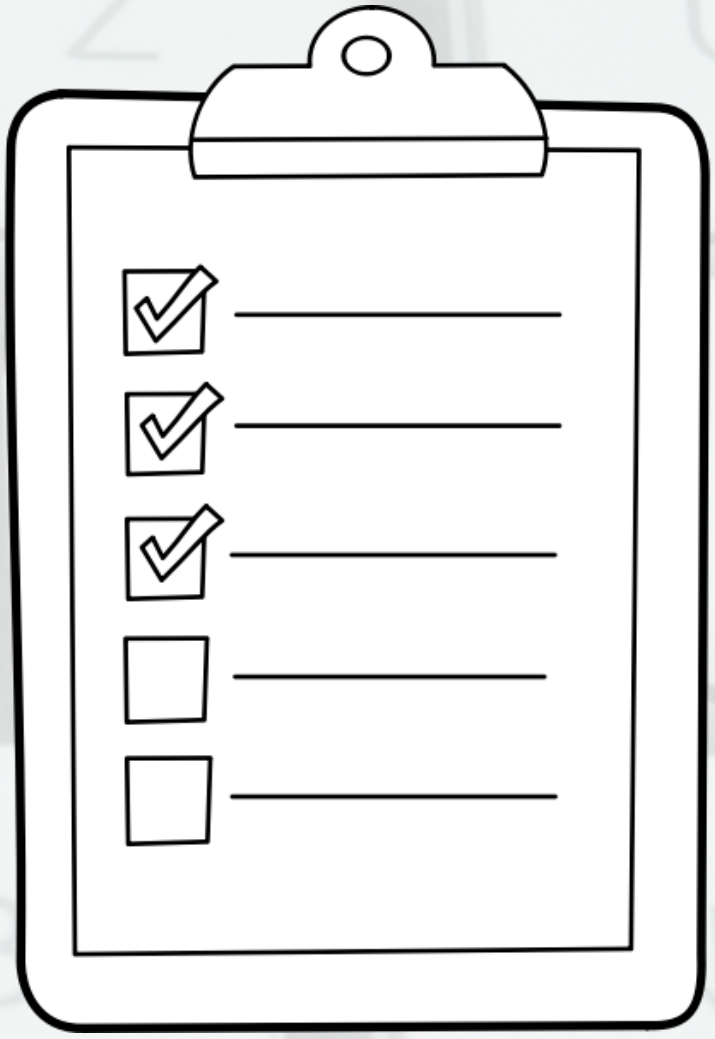
See handout

Agenda

Module	Topics
1	Intro & Changes to Strings
2 	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

Lab Review

- Any questions?
- Solution review



Features

Feature	Preview	Final
Switch Expressions	12, 13	14
Pattern Matching for instanceof	14, 15	16
Pattern Matching for switch	17	?

What does this print?

1?

```
String store = "Hallmark";

switch(store) {
     case "Hallmark" : System.out.println("KC");
     case "Crayola" : System.out.println("PA");
     default: System.out.println("anywhere");
}
```

Three lines (missing break)

Switch Expressions


```
String store = "Hallmark";

switch(store) {
     case "Hallmark" -> System.out.println("KC");
     case "Crayola", "H&R"
       -> System.out.println("PA");
     default -> System.out.println("anywhere");
}
```

Arrow labels

No break keyword

Multiple values

Switch Expressions

14

```
String store = "Hallmark";

String output = switch(store) {
    case "Hallmark" -> "KC";
    case "Crayola" -> "PA";
    default -> "anywhere";
};
System.out.println(output);
```

Switch returns values

More features

14

```
String output = switch(store) {
    case "Hallmark" -> "KC";
    case "Legoland" -> {
        int random = new Random().nextInt();
        String city = random % 2 == 0
            ? "KC" : "Carlsbad";
        yield city;
    }
    default -> throw new
        IllegalArgumentException("Unknown");
};
System.out.println(output);
```

Block

yield

throws exception
so no return value
needed

Is this legal?

14

```
private void confusing() {
    this.yield();
}

private void yield() {
    String store = "Legoland";

    String output = switch(store) {
        case "Legoland" -> {
            yield "Carlsbad";
        }
        default -> "other";
    };
    System.out.println(output);
}
```

Yes. yield is like var

@jeanneboyarsky

How about now?

14

```
private void confusing() {
    yield();
}

private void yield() {
    String store = "Legoland";

    String output = switch(store) {
        case "Legoland" -> {
            yield "Carlsbad";
        }
        default -> "other";
    };
    System.out.println(output);
}
```

No

to invoke a method called yield, qualify the yield with a receiver or type name

@jeanneboyarsky

Yield with Switch Stmt

14

```
Position pos = Position.TOP;

int stmt = switch(pos) {
    case TOP: yield 1;
    case BOTTOM: yield 0;
};

int expr = switch(pos) {
    case TOP -> 1;
    case BOTTOM -> 0;
};
```

Same!

@jeanneboyarsky

Missing value


```
enum Position { TOP, BOTTOM };


Position pos = Position.TOP;

int stmt = switch(pos) {
    case TOP: yield 1;
};

int expr = switch(pos) {
    case BOTTOM -> 0;
};
```

Does not compile because assigning value

(poly expression)

Pattern matching for if

16

```
if (num instanceof Integer) {
    Integer numAsInt = (Integer) num;
    System.out.println(numAsInt);
}
if (num instanceof Double) {
    Double numAsDouble = (Double) num;
    System.out.println(numAsDouble.intValue());
}

if (num instanceof Integer numAsInt) {
    System.out.println(numAsInt);
}
if (num instanceof Double numAsDouble) {
    System.out.println(numAsDouble.intValue());
}
```

Pattern variable

@jeanneboyarsky

Flow Scope

16

```
if (num instanceof Double d1
    && d1.intValue() % 2 == 0) {

    System.out.println(d1.intValue());
}

if (num instanceof Double d2
    || d2.intValue() % 2 == 0) {

    System.out.println(d2.intValue());
}
```

Compiles

Does not compile because d2 might not be double

Does this compile?

16

```
if (num instanceof Double n)
    System.out.println(n.intValue());

if (num instanceof Integer n)
    System.out.println(n);
```

Yes. Only in scope for if statement

Does this compile?

16

```
if (num instanceof Double n)
     System.out.println(n.intValue());

System.out.println(n.intValue());
```

No. If statement is over

Does this compile?

16

```
if (!(num instanceof Double n)) {
    return;
}
System.out.println(n.intValue());
```

Yes. Returns early so rest is like an else

Does this compile?

16

```
if (!(num instanceof Double n)) {
    return;
}
System.out.println(n.intValue());

if (num instanceof Double n)
    System.out.println(n.intValue());
```

No. n is still in scope

Reusing a variable

16

```
if (num instanceof Integer numAsInt) {
     numAsInt = 6;
     System.out.println(numAsInt);
}
```

Legal. please don't.

Pattern matching for switch

17 preview

```
static int toInt(Object obj) {
    return switch (obj) {
        case Integer i -> i;
        case Double d -> d.intValue();
        case String s -> Integer.parseInt(s);

        default -> throw new
            IllegalArgumentException("unknown type");
    };
}
```

Reminder: Syntax can change

But wait, there's more

17 preview

```
static void printOddOrEven(Object obj) {
    switch (obj) {

        case Integer i && i % 2 == 1 ->
            System.out.println("odd");

        case Integer i && i % 2 == 0 ->
            System.out.println("even");

        default -> System.out.println("not an int");
    };
}
```

Reminder: Feature can still change

Module 2 - Question 1

What is output?

```
char ch = 'b';

int count = 0;
switch (ch) {
    case 'a' -> count++;
    case 'b' -> count+=2;
    case 'c' -> count+=3;
 }

 System.out.println(count);
```

A. 1
B. 2
C. 5
D. Does not compile

Module 2 - Question 2

How many changes are needed to have this code print 2?

```
char ch = 'b';

int value = switch (ch) {
    case 'a' -> 1;
    case 'b' -> yield 2;
    case 'c' -> 3;
 }

 System.out.println(value);
```

A. 1
B. 2
C. 3
D. 4

Module 2 - Question 3

How many lines need changing to make this code compile?

```
char ch = 'b';

int value = switch (ch) {
    case 'a' : yield 1;
    case 'b' : { yield 2; }
    case 'c' -> yield 3;
    default -> 4;
};
System.out.println(value);
```

A. 1
B. 2
C. 3
D. 4

Module 2 - Question 4

What can fill in the blank to have the code print 2?

```
char ch = 'b';

______ value = switch (ch) {
    case 'a' -> 1;
    case 'b' -> 2L;
    case 'c' -> 3.0;
    default -> 4;
};
System.out.println(value);
```

A. int
B. Object
C. Either A or B
D. None of the above

Module 2 - Question 5

What does the following output?

```
char ch = 'b';

switch (ch) {
    case 'a' -> System.out.println(1);
    case 'b' -> System.out.println(2);
    case 'c' -> { System.out.println(3); }
};
```

A. 1

B. 2

C. 3

D. Does not compile

Module 2 - Question 6

What does the following print?

```
Object robot = "694";

if (robot instanceof String s) {
    System.out.print("x");
}
if (robot instanceof Integer s) {
    System.out.print("y");
}
System.out.println(robot);
```

A. x694
B. xy694
C. y694
D. Does not compile

Module 2 - Question 7

Which lines have s in scope?

```
Object robot = "694";

if (robot instanceof String s) {
    // line 1
}
if (robot instanceof int i) {
    // line 2
}

// line 3
```

A. 1
B. 1 and 3
C. 1, 2 and 3
D. Does not compile

Module 2 - Question 8

What is true about this class?

```
class Sword {
    int length;

    public boolean equals(Object o) {
        if (o instanceof Sword sword) {
            return length == sword.length;
        }
        return false;
    }
    // assume hashCode properly implemented
}
```

A. equals() is correct
B. equals() is incorrect
C. equals() does not compile

Module 2 - Question 9

How many if statements fail to compile?

```
Number n = 4;

if (n instanceof Integer x) {}
if (n instanceof Integer x
  && x.intValue() > 1) {}
if (n instanceof Integer x
  || x.intValue() > 1) {}
if (n instanceof Integer x
  || x.toString().isEmpty()) {}
```

A. 0
B. 1
C. 2
D. 3

Module 2 - Question 10

What does printLength(3) print?

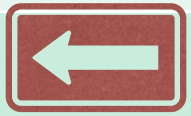
```
class Sword {
    int length = 8;

    public void printLength(Object x) {
        if (x instanceof Integer length) {
            length = 2;
        }
        System.out.println(length);
    }

}
```

A. 2
B. 3
C. 8
D. Does not compile

Lab & Break

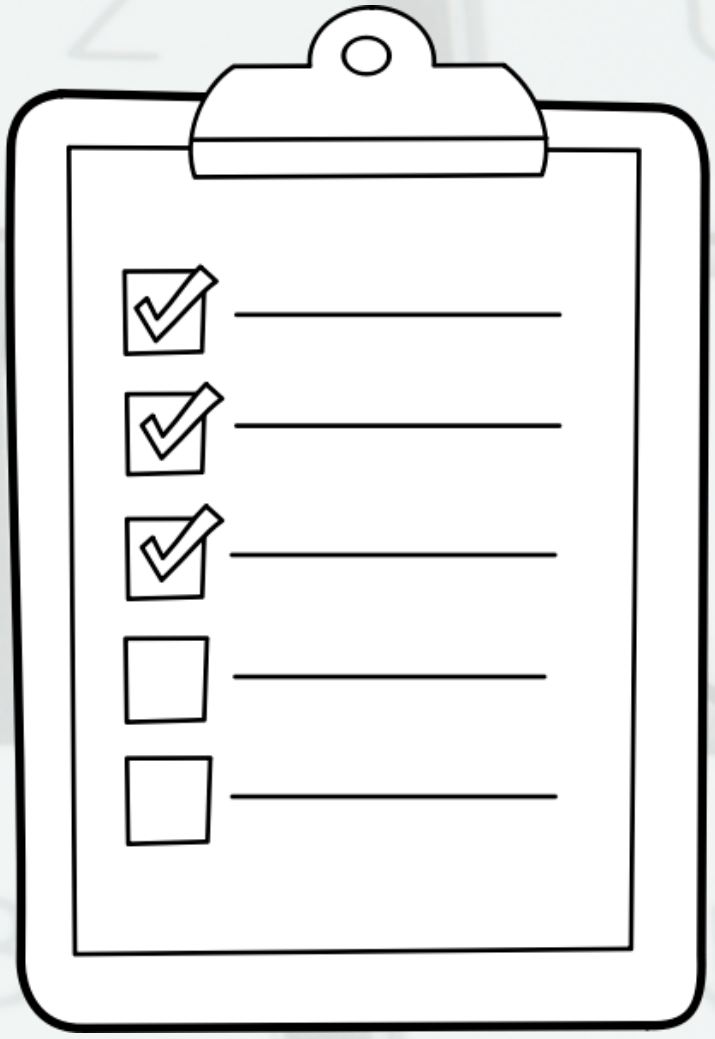
See handout

Agenda

Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3 	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

Lab Review

- Any questions?
- Solution review



Features

Feature	Preview	Final
Records	14, 15	16
Sealed classes	15, 16	17

Now allowed

16

```
public class Kangaroo {

    class Joey {
        static int numJoeys = 0;
    }

    void hop() {
        interface Hopper {}
        enum Size { EXTRA_SMALL, SMALL };
    }
}
```

Static field in inner classes and method local enums/interfaces

Immutable class

16

1. Make fields final and private
2. Don’t provide setters
3. No subclasses (ex: make class final)
4. Write constructor taking all fields

POJO

16

- constructor
- toString()
- hashCode() - more rules
- equals() - still more rules

Simple Record

16

```
public record Book (String title, int numPages) {
}
```

New type

Automatically get
* final record
* private final instance variables
* public accessors
* constructor taking both fields
* equals
* hashCode
* no instance initializers

@jeanneboyarsky

Using the Record

16

```
Book book = new Book("Breaking and entering", 289);

System.out.println(book.title());
System.out.println(book.toString());
```

No “get”

Outputs:
Breaking and entering
Book[title=Breaking and entering, numPages=289]

Add/change methods

16

```
public record Book (String title, int numPages) {

    @Override
    public String title() {
        return '"' + title + '"';
    }

    public boolean isLong() {
        return numPages > 300;
    }

}
```

Change behavior

Custom method

@jeanneboyarsky

Not really immutable

16

```
public record Book (String title, int numPages,
    List<String> chapters) {
}


Book book = new Book("Breaking and entering", 289,
    chapters);

chapters.add("2");
book.chapters().add("3");
System.out.println(book.chapters());
```

Prints [1,2,3] because shallow immutability

Now immutable

16

```
public record Book (String title, int numPages,
    List<String> chapters) {

    public Book {
        chapters = List.copyOf(chapters);
    }
}
```

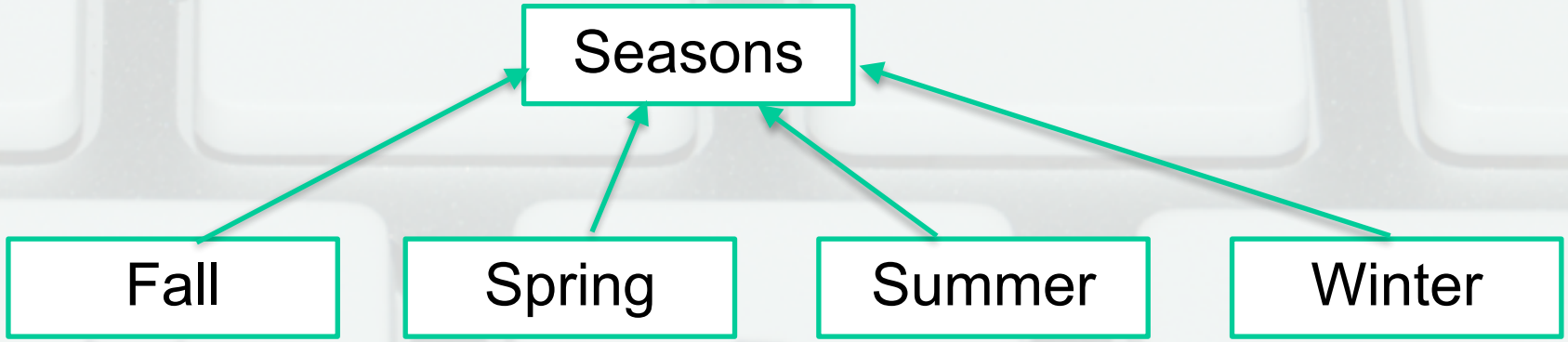
Compact constructor

Must match record access modifier

Sealed classes

17

```
public abstract sealed class Seasons
    permits Fall, Spring, Summer, Winter { }

final class Fall extends Seasons {}
final class Spring extends Seasons {}
final class Summer extends Seasons {}
final class Winter extends Seasons {}
```



Subclass modifiers

17

Modifer	Meaning
final	Hierarchy ends here
non-sealed	Others can subclass
sealed	Another layer

Subclass reference

17

Location	How reference in permits clause
same file	Package name optional Permits clause optional
same package in named/ unnamed module	Package name optional
different package in named module	Package name required
different package in unnamed module	Not allowed

Sealed interface

17

```
public sealed interface TimeOfDay
    permits Morning, Afternoon, Evening {
     boolean early();
}
public non-sealed class Morning implements TimeOfDay {
     public boolean early() { return true; }
}
public non-sealed class Afternoon implements TimeOfDay {
     public boolean early() { return false; }
}
public record Evening(int hour) implements TimeOfDay {
     public boolean early() { return false; }
}
```


Records are implicitly final

instanceof

17

```
public class InstanceOf {
 static sealed class BoolWrapper
    permits TrueWrapper, FalseWrapper { }
 static final class TrueWrapper extends BoolWrapper {}
 static final class FalseWrapper extends BoolWrapper {}
 public static void main(String[] args) {
   Map<?, ?> map = new HashMap<>();
   String string = "";
   BoolWrapper boolWrapper = new TrueWrapper();

   System.out.println(map instanceof List);          // false
   System.out.println("" instanceof List);           // error
   System.out.println(boolWrapper instanceof List); // error
}
```

Module 3 - Question 1

How many lines need to be removed for this code to compile?

```
public record BBQ(String type) {}

public static void main(String[] args) {
    BBQ bbq = new BBQ("chicken");
    System.out.println(bbq.setType("pork"));
    System.out.println(bbq.getType());
    System.out.println(bbq.equals(bbq));
}
```

A. 0
B. 1
C. 2
D. None of the above

Module 3 - Question 2

What does this output?

```
public record BBQ(String type) {
     BBQ {
          type = type.toUpperCase();
     }
}

public static void main(String[] args) {
     BBQ bbq = new BBQ("chicken");
     System.out.println(bbq.type());
}
```

A. chicken
B. CHICKEN
C. Does not compile
D. None of the above

Module 3 - Question 3

What does this output?

```
record BBQ(String type) {
     BBQ {
          type = type.toUpperCase();
     }
}

public static void main(String[] args) {
     BBQ bbq = new BBQ("chicken");
     System.out.println(bbq.type());
}
```

A. chicken
B. CHICKEN
C. Does not compile
D. None of the above

Module 3 - Question 4

How many compiler errors are in the following code?

```
public final record BBQ(String type) {
    { type = ""; }
    public BBQ(String type) {
        type = type.toUpperCase();
    }
    public void type() { return ""; }
    public String toString() { return ""; }
}
```

A. 1
B. 2
C. 3
D. 4

Module 3 - Question 5

What does this output?

```
public record BBQ(String type)
     implements Comparable<BBQ> {

     public int compareTo(BBQ bbq) {
          return type.compareTo(bbq.type);
     } }
public static void main(String[] args) {
     BBQ beef = new BBQ("beef");
     BBQ pork = new BBQ("pork");
     System.out.println(pork.compareTo(beef));
}
```

A. Negative #
B. Positive #
C. 0
D. Does not compile

Module 3 - Question 6

What does this output?

```
static sealed interface Weather permits Wet, Dry{
    boolean needUmbrella(); }
static non-sealed class Wet implements Weather {
    public boolean needUmbrella() { return true; } }
static record Dry(boolean needUmbrella)
  implements Weather {}

public static void main(String[] args) {
    Weather weather = new Dry(false);
    System.out.println(weather.needUmbrella());
}
```

A. true
B. false
C. Does not compile
D. None of the above

Module 3 - Question 7

Which of the following are true? (Choose all that apply)

A. There is only one hyphenated modifier in Java (non-sealed)

B. A sealed interface may permit another interface.

C. Sealed records are allowed

D. Sealed enums are allowed

Module 3 - Question 8

Given the following, where could Android be? (Choose all that apply)

```
package general;
static sealed class Phone
  permits IPhone, Android { }
```

A. In the same file as Phone

B. In the same package as Phone within a module

C. In a different package from Phone, but in the same module

D. In a different module

Module 3 - Question 9

Given the following, where could Android be? (Choose all that apply)

```
package general;
static sealed class Phone
  permits mac.IPhone, google.Android { }
```

A. In the same file as Phone

B. In the same package as Phone within a module

C. In a different package from Phone, but in the same module

D. In a different module

Module 3 - Question 10

How many compiler errors are in this code?

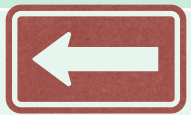
```
public sealed class Phone {

    class IPhone extends Phone {
    }

    class Android extends Phone {
    }
}
```

A. 0
B. 1
C. 2
D. 3

Lab & Break

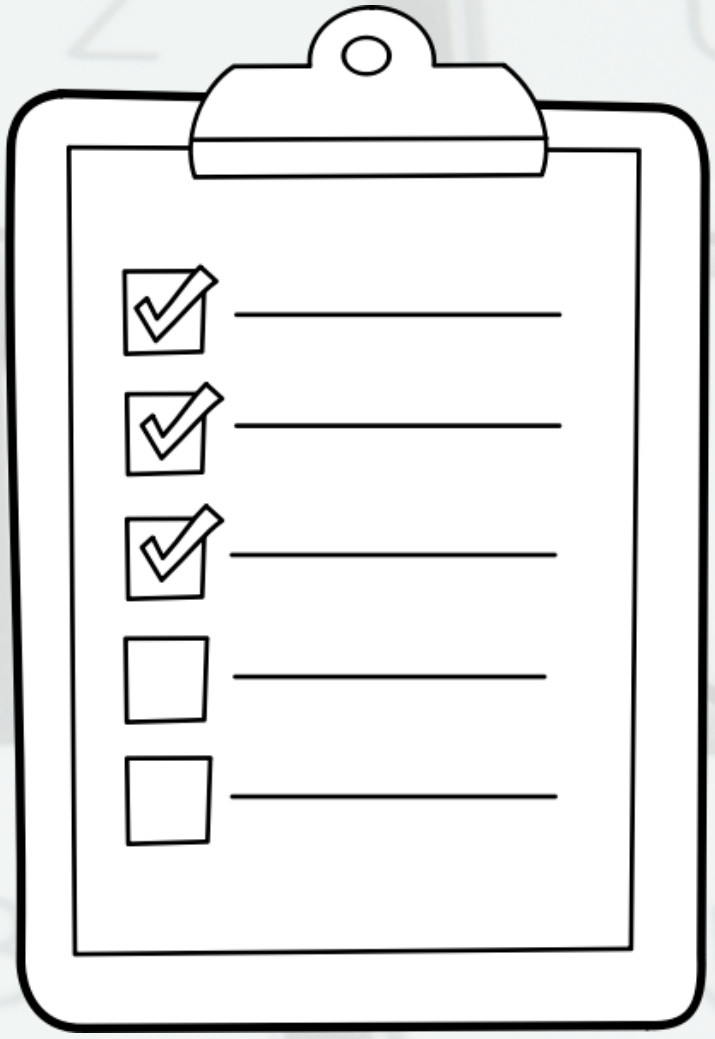
See handout

Agenda

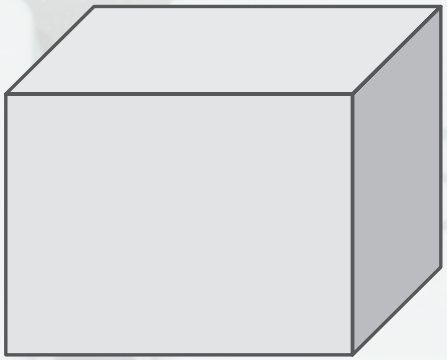
Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4 	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

Lab Review

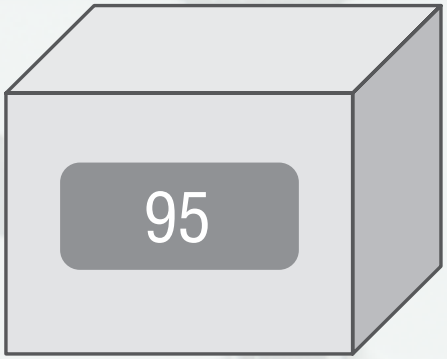
- Any questions?
- Solution review



Optional

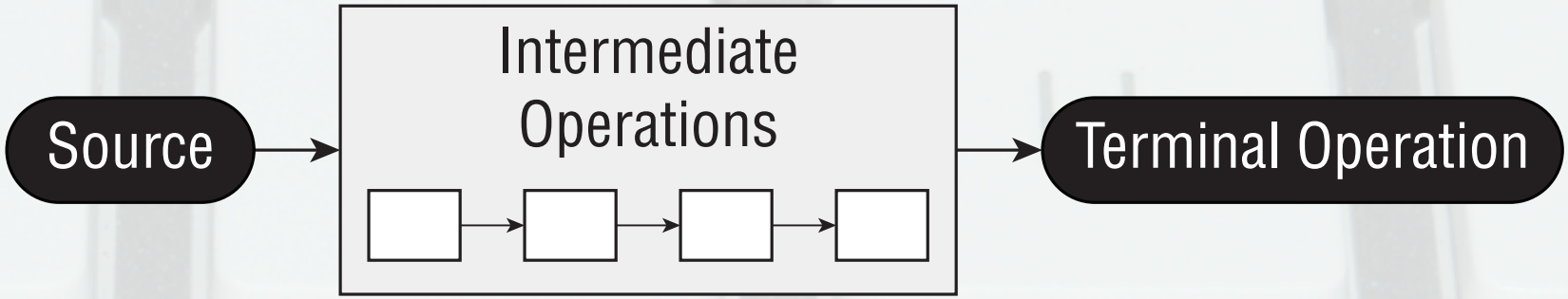


`Optional.empty()`

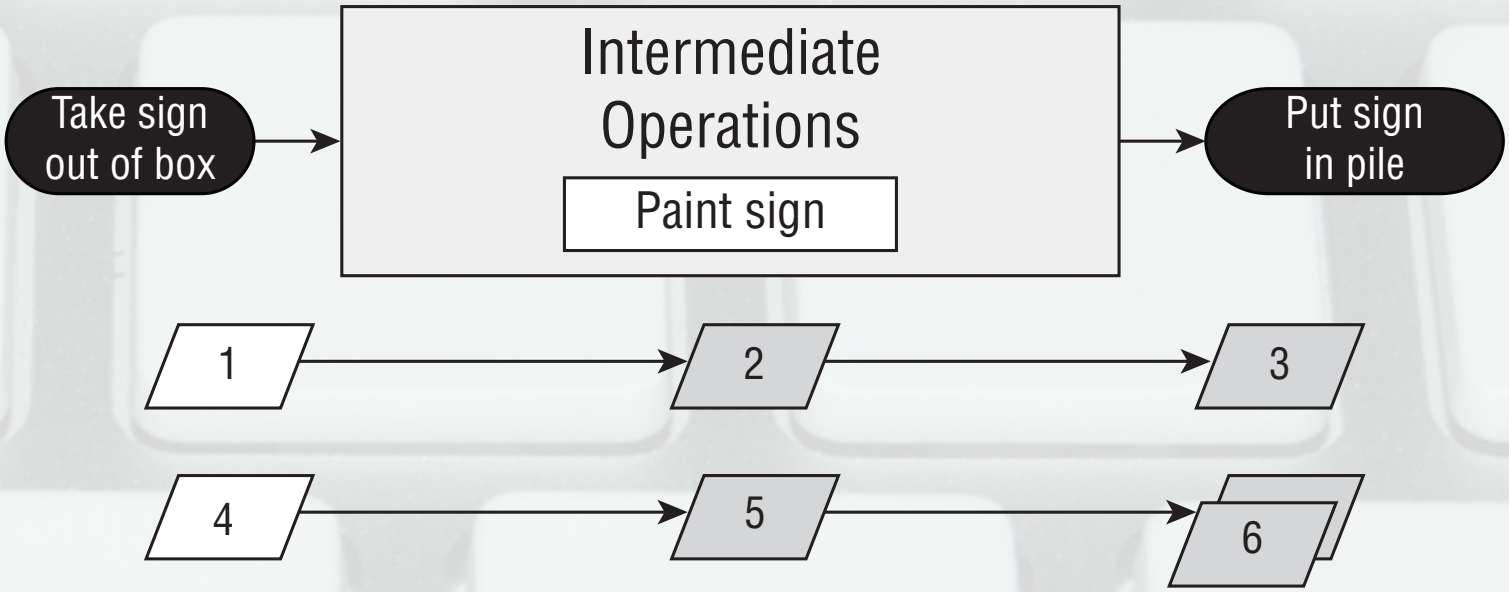


`Optional.of(95)`

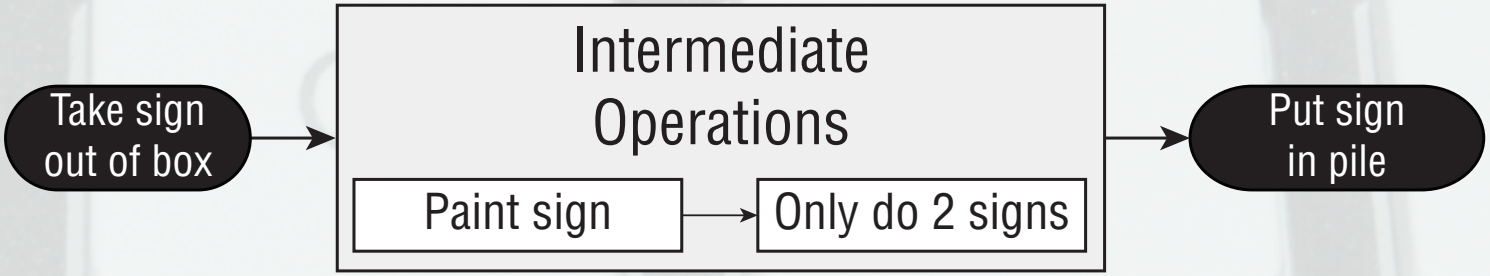
Stream Flow



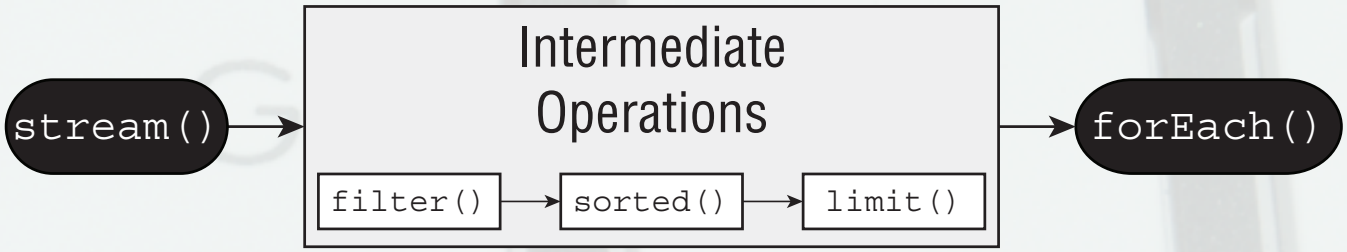
Painting Example



Painting Example



Painting Example



Basic Stream

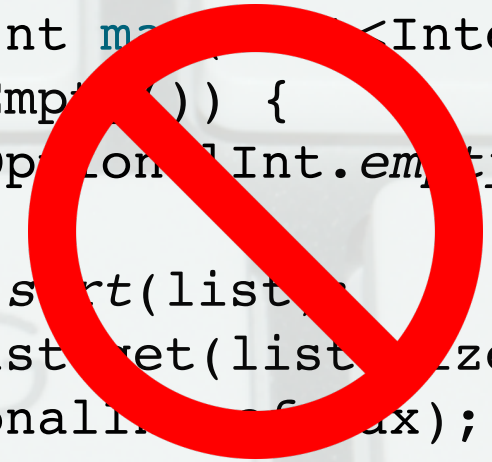
```
list.stream()
        .map(String::strip)
        .filter(s -> ! s.isBlank())
        .map(String::length)
        .collect(Collectors.toList());
```

Source

Intermediate ops

Terminal op

Find Max



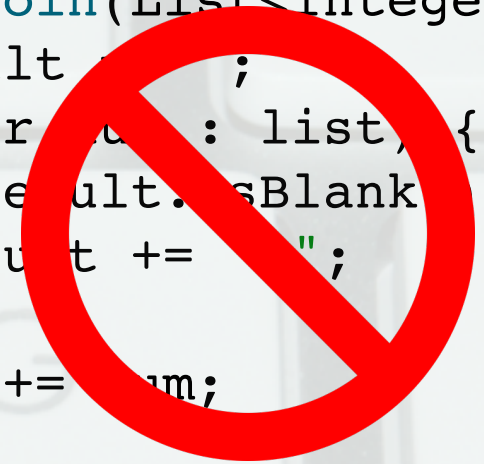
```
public OptionalInt ma<Integer> list) {
    if (list.isEmp()) {
        return Op lInt.em ty();
    }
    Collections.s rt(list);
    int max = list et(list ze() - 1);
    return Optionali ax);
}
```

```
public OptionalInt maxStream(List<Integer> list) {
    list.stream().mapToInt(x -> x).max();
}
```

@jeanneboyarsky

Join a String

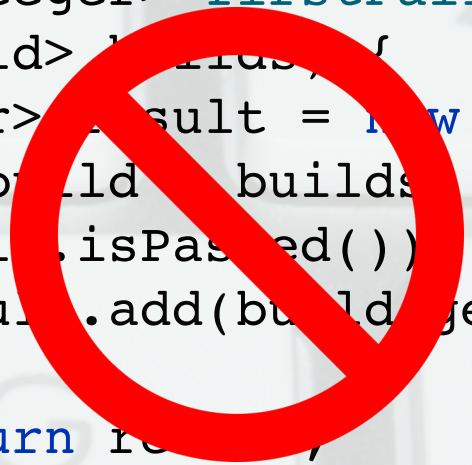
```
public String join(List<Integer> list) {
    String result = "";
    for (Integer i : list) {
        if (! result.isBlank()) {
            result += " ";
        }
        result += i;
    }
    return result;
}

public String joinStream(List<Integer> list) {
    return list.stream()
            .map(Object::toString)
            .collect(Collectors.joining(" "));
}
```



@jeanneboyarsky

Get First Failures

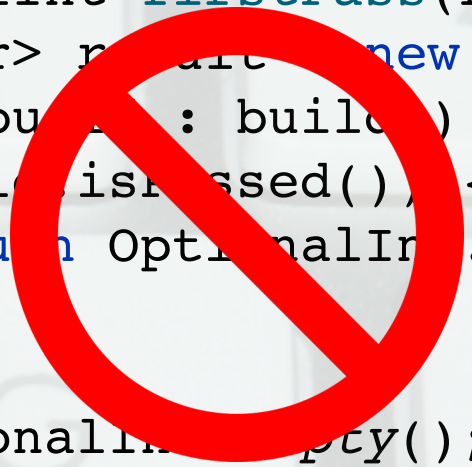
```
public List<Integer> firstFailures(
        List<Build> builds) {
    List<Integer> result = new ArrayList<>();
    for (Build build : builds)
        if (!build.isPassed())
            result.add(build.getNum());
        else
            return result;
    return Collections.emptyList();
}
public List<Integer> firstFailuresAsStream(
        List<Build> builds) {
    return builds.stream()
            .takeWhile(b -> ! b.isPassed())
            .map(Build::getNum)
            .collect(Collectors.toList());
}
```



@jeanneboyarsky

Get First Pass

```
public OptionalInt firstPass(List<Build> builds) {
    List<Integer> r      = new ArrayList<>();
    for (Build bu    : build  ) {
        if (buil  is  ssed()) {
            retu  n Opti  alIn  .of(build.getNum());
        }
    }
    return Optionall    pty();
}

public OptionalInt firsrtPassAsStream(
      List<Build> builds) {
    return builds.stream()
            .dropWhile(b -> ! b.isPassed())
            .mapToInt(Build::getNum)
            .findFirst();
}
```



@jeanneboyarsky

Strings

```
public long countLinesWithQuestionMark(String text) {
    return text.lines()
            .filter(s -> s.contains("?"))
            .count();
}


public long countQuestionMark(String text) {
      return text.chars()
              .filter(c -> c == '?')
              .count();
}
```

Common Terminal Ops

- count()
- collect() - more on this shortly
- allMatch(), anyMatch(), noneMatch()
- findFirst(), findAny()
- min()/max()
- forEach()

Common Intermediate Ops

- filter()
- map()
- distinct()
- limit(), skip()
- sorted()
- peek()

Generating Infinite Stream

```
public String tenStars() {
    return Stream.generate(() -> "*")
            .limit(10)
            .collect(Collectors.joining());
}
```

Iterative Infinite Stream

```
public String counting() {
    return Stream.iterate(1, i -> i+1)
            .limit(10)
            .map(Object::toString)
            .collect(Collectors.joining(" "));
}


public String countingWithLimit() {
    return Stream.iterate(1, i -> i <=10, i -> i+1)
            .map(Object::toString)
            .collect(Collectors.joining(" "));
}
```

FlatMap

```
public List<String> flatten() {
    var first = List.of("a", "b");
    List<String> second = List.of();
    var listOfLists = List.of(first, second);

    return listOfLists.stream()
            .flatMap(Collection::stream)
            .collect(Collectors.toList());
}
```

var!

Why List<String>?

Creating a Map

```
public Map<String, Integer> mapByName(
        List<String> list) {
    return list.stream()
            .collect(Collectors.toMap(
                Function.identity(),
                String::length));
}

public Map<Integer, String> mapBySize(
        List<String> list) {
    return list.stream()
            .collect(Collectors.toMap(
                String::length,
                Function.identity(),
                (a, b) -> a));
}
```

@jeanneboyarsky

Adding a level

```
public Map<Integer, Long> grouping(
        List<String> list) {
    return list.stream()
            .collect(Collectors.groupingBy(
                    String::length,
                    Collectors.counting()));
}

public Map<Boolean, Long> partitioning(
        List<String> list) {
    return list.stream()
            .collect(Collectors.partitioningBy(
                    String::isEmpty,
                    Collectors.counting()));
}
```

Grouping vs Partitioning

```
public Map<Integer, List<String>> grouping(
        List<String> list) {
    return list.stream()
            .collect(Collectors.groupingBy(
                    String::length));
}


public Map<Boolean, List<String>> partitioning(
        List<String> list) {
    return list.stream()
            .collect(Collectors.partitioningBy(
                    String::isEmpty));
}
```

Reducing

```
public Integer sum(List<Integer> list) {
    return list.stream()
               .reduce(0, (x,y) -> x + y);
}
```

```
public Optional<Integer> min(List<Integer> list) {
    return list.stream()
               .reduce((x, y) -> x < y ? x : y);
}
```

Advanced Ops

- Sources
  - generate()
  - iterate()
  - ofNullable()
- Intermediate
  - flatMap()
- Terminal
  - reduce()
- Collectors
  - toMap()
  - groupingBy()
  - partitioningBy()

Teeing Collector

12

```
record Separations(String spaceSeparated,
    String commaSeparated) {}

var list = List.of("x", "y", "z");
Separations result = list.stream()
        .collect(Collectors.teeing(
              Collectors.joining(" "),
              Collectors.joining(","),
              (s, c) -> new Separations(s, c)));

System.out.println(result);
```

Module 4 - Question 1

What does this output?

```
public static void main(String[] args) {
    long count = Stream
            .iterate(1; i-> i< 10;
                i-> i+2).count();
    System.out.println(count);
}
```

A. 0

B. 5

C. Does not compile

D. None of the above

Module 4 - Question 2

What does this output?

```
Stream.iterate(1, i-> i< 10, i-> i++)
        .takeWhile(i -> i < 5)
        .forEach(System.out::println);
```

A. The numbers 1-4

B. The numbers 5-10

C. Does not compile

D. None of the above

Module 4 - Question 3

What does this output?

```
var map = Stream.generate(() -> 1)
          .limit(5)
          .collect(Collectors.partitioningBy(
               x -> x % 2 ==0));
System.out.println(map);
```

A. {false=[1, 1, 1, 1, 1]}

B. {false=[1, 1, 1, 1, 1], true=[]}

C. Does not compile

D. None of the above

Module 4 - Question 4

What does this output?

```
long count = Stream.of(null).count();
System.out.println(count);
```

A. 0

B. 1

C. null

D. Does not compile

E. None of the above

Module 4 - Question 5

What does this output?

```
long count = Stream.ofNullable(null).count();
System.out.println(count);
```

A. 0

B. 1

C. null

D. Does not compile

E. None of the above

Module 4 - Question 6

What does this output?

```
List.of(1,2,3).stream()
        .collect(Collectors.teeing(
              Collectors.toSet(),
              System::forEach));
```

A. 1,2,3

B. {1,2,3}

C. Both A and B

D. Does not compile

E. None of the above

Module 4 - Question 7

Which can fill in the blank to print false?
(Choose all that apply)

```
var s = Stream.generate(() -> "meow");
var match = ____________________(String::isEmpty);
System.out.println(match);
```

A. anyMatch

B. allMatch

C. noneMatch

D. findFirst

Module 4 - Question 8

Which fill in the blank to compile but cause the program to hang?(Choose all that apply)

```
var s = Stream.generate(() -> "meow");
var match = _________________(String::isEmpty);
System.out.println(match);
```

A. anyMatch

B. allMatch

C. noneMatch

D. findFirst

Module 4 - Question 9

What is the output of the following?

```
var x = Stream.of("a", "b", "c")
         .filter(c -> c != "b");
System.out.println(x);
```

A. [b]

B. [a, c]

C. [a, b, c]

D. Does not compile

E. None of the above

Module 4 - Question 10

If ____ fills in the blank, then the output is_____ (Choose two)

```
var x = Stream.of("a", "bb", "_________")
                  .collect(Collectors.toMap(
                       String::length,
                       Function.identity()));
System.out.println(x);
```

A. cc, {1=a, 2=bb, 3=cc}

B. cc, a stack trace

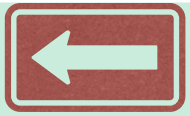
C. ccc, {1=a, 2=bb, 3=ccc}

D. ccc, a stack trace

Lab & Break

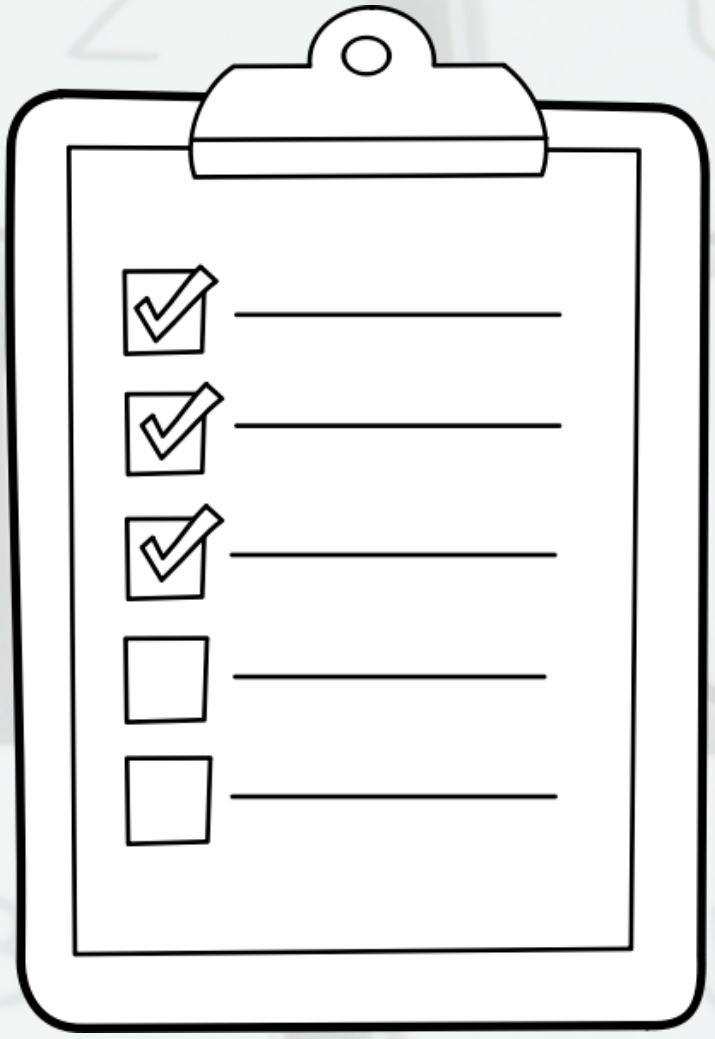
See handout

Agenda

Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5 	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

Lab Review

- Any questions?
- Solution review



Formatting a String


```
String firstName = "Jeanne";
String lastName = "Boyarsky";
String str = String.format(
    "Hi %s %s!", firstName, lastName);
System.out.println(str);

System.out.println("Hi %s %s!".formatted(
     firstName, lastName));
```

Outputs:
Hi Jeanne Boyarsky!
Hi Jeanne Boyarsky!

Common Conversions

Conversion	What it does
%s	Formattable as String
%d	Decimal integer (no dot)
%c	Char
%f	Float (decimal)
%n	New line

Many more out of scope. Examples:

- %e - scientific notation
- %t - time
- %S - converts to all uppercase

Conversion Examples

12

Code	Output
`"%d%%".formatted(1.2)`	exception
`"%d%%".formatted(1)`	1%
`"%s%%".formatted(1)`	1%
`"%s%%".formatted(1.2)`	1.2%
`“%f%%".formatted(1.2)`	1.200000f

Formatting a Number

Char	What it does
-	Left justified
+	Always include +/-
space	Leading space if positive

Char	What it does
0	Zero padded
,	Group numbers
(	Negative # in parens

Flag Examples

12

Code	Output
`"%,d".formatted(1234)`	1,234
`"%+d".formatted(1234)`	1234
`“% d".formatted(1234)`	1234
`“%,(d”.formatted(-1234)`	(1,234)
`“%,f”.formatted( 1.23456789)`	1.234568

Compact Number

12

```
NumberFormat defaultFormat =
NumberFormat.getCompactNumberInstance();
NumberFormat shortFormat = NumberFormat
    .getCompactNumberInstance(
        Locale.US, NumberFormat.Style.SHORT);
NumberFormat longFormat = NumberFormat
    .getCompactNumberInstance(
        Locale.US, NumberFormat.Style.LONG);

System.out.println(defaultFormat.format(1_000_000));
System.out.println(shortFormat.format(1_000_000));
System.out.println(longFormat.format(1_000_000));
```

```
1M
1M
1 million
```

Date Formatting

Letter	What it does
M	Month
d	Day
Y	Year

Letter	What it does
h/H	Hour (12 or 24 hours)
m	Minute
s	Second
S	Millisecond
Z	Timezone

Advanced Examples

```
LocalDate date = LocalDate.of(2021, Month.SEPTEMBER, 15);
LocalTime time = LocalTime.of(13, 23);
LocalDateTime dateTime = LocalDateTime.of(date, time);

System.out.println(dateTime.format(
   DateTimeFormatter.ofPattern("_______")));
```

Code	Output
`MM-dd`	09-15
`MM/dd/yyyy`	Sep/15/2021
`MMMM dd, yyyy`	September 15, 2021
`hh:mm:ss`	13:23:00
`h 'o''clock'`	1 o’clock

Create a Locale

```
Locale loc = Locale.getDefault();
Locale constant = Locale.JAPANESE;
Locale lang = new Locale("en");
Locale country = new Locale("en", "US");
Locale invalid = new Locale("US");
```

Invalid locales don't throw an exception, but don't match

Formatting with Locales

```
NumberFormat shortFormat = NumberFormat.
    getCompactNumberInstance(
          Locale.CANADA_FRENCH,
          NumberFormat.Style.SHORT);
NumberFormat longFormat = NumberFormat.
    getCompactNumberInstance(
          Locale.CANADA_FRENCH,
          NumberFormat.Style.LONG);
System.out.println(shortFormat.format(3_000));
System.out.println(longFormat.format(3_000));
```

```
3 k
3 mille
```

Reading a Resource Bundle

```
# Zoo_en.properties
hello=Hello
open=The zoo is open
```

```
# Zoo_fr.properties
hello=Bonjour
open=Le zoo est ouvert
```

```
// in main method
Locale french = new Locale("fr", "FR");
ResourceBundle rb =
   ResourceBundle.getBundle("Zoo", french);
System.out.println(rb.getString("open"));
```

Le zoo est ouvert

Picking the right bundle

Looks for file	Reason
Zoo_fr_FR.properties	Requested locale
Zoo_fr.properties	Requested language
Zoo_en_US.properties	Default locale
Zoo_en.properties	Default language
Zoo.properties	Default bundle
throws exception	No matches

Once finds bundle, can only get values from it or parent

Message Format

```
# Zoo_en.properties
hello=Hello {0}
open=The zoo is open
```

```
# Zoo_fr.properties
hello=Bonjour {0}
open=Le zoo est ouvert
```

```
// in main method
ResourceBundle rb =
   ResourceBundle.getBundle("Zoo");
String format = rb.getString("hello");
System.out.println(MessageFormat.format(
   format, "Jeanne"));
```

Hello Jeanne

Module 5 - Question 1

What does this output?

```
System.out.println("%s %d".formatted(
        "KCDC", "2021"));
```

A. KCDC2021

B. KCDC2021.0

C. KCDC 2021

D. KCDC 2021.0

E. None of the above

Module 5 - Question 2

What does this output?

```
System.out.println("%s +%05d".formatted(
        "KCDC", 2021));
```

A. KCDC 2021

B. KCDC +2021

C. KCDC +02021

D. KCDC +002021

E. None of the above

Module 5 - Question 3

How many things need to be changed to print: 2K 3.14?

```
CompactNumberFormat fmt =
    new CompactNumberFormat();
System.out.println(fmt.format(2000) +
    " %.2d".formatted(3.14));
```

A. 1

B. 2

C. 3

D. 4

Module 5 - Question 4

What does this print?

```
LocalDateTime dateTime = LocalDateTime.of(2021,
    Month.SEPTEMBER, 14, 6, 15, 44);
DateTimeFormatter fmt =
    DateTimeFormatter.ofPattern("M HH");
System.out.println(dateTime.format(fmt));
```

A. 9 6

B. 9 06

C. 15 6

D. 15 06

E. None of the above

Module 5 - Question 5

How many of these are valid locales?

```
Locale a = new Locale("en");
Locale b = new Locale("US");
Locale c = Locale.of("US");
Locale d = Locale.of("US");
```

A. 0

B. 1

C. 2

D. 3

E. 4

Module 5 - Question 6

Given the answers, which will be the first bundle to match?

```
Locale.setDefault(Locale.US);
ResourceBundle rb =
    new ResourceBundle("Zoo",
    new Locale("fr"));
```

A. Zoo_en.properties

B. Zoo_fr_FR.properties

C. Zoo.properties

D. None of the above

Module 5 - Question 7

Given the answers, which will be the first bundle to match?

```
Locale.setDefault(Locale.US);
ResourceBundle rb =
   ResourceBundle.getBundle("Zoo",
   new Locale("fr"));
```

A. Zoo_en.properties

B. Zoo_US.properties

C. Zoo.properties

D. None of the above

Module 5 - Question 8

Suppose rb is Locale_en.properties in this example. Which file will Java look at first if there is no match in that file?

```
ResourceBundle rb =
   ResourceBundle.getBundle("Zoo",
   new Locale("fr"));
```

A. Zoo_en_US.properties

B. Zoo_fr_FR.properties

C. Zoo.properties

D. None of the above

Module 5 - Question 9

What does the following output?

```
String format = "{2} < {1}";
System.out.println(format.formatted(3, 5));
```

A. 3 < 5

B. 5 < 3

C. {2} < 3

D. {2} < 5

E. None of the above

Module 5 - Question 10

What does the following output?

```
String format = "{1} < {0}";
System.out.println(
  MessageFormat.format(3, 5));
```

A. 3 < 5

B. 5 < 3

C. {2} < 3

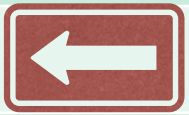
D. {2} < 5

E. None of the above

Lab & Break

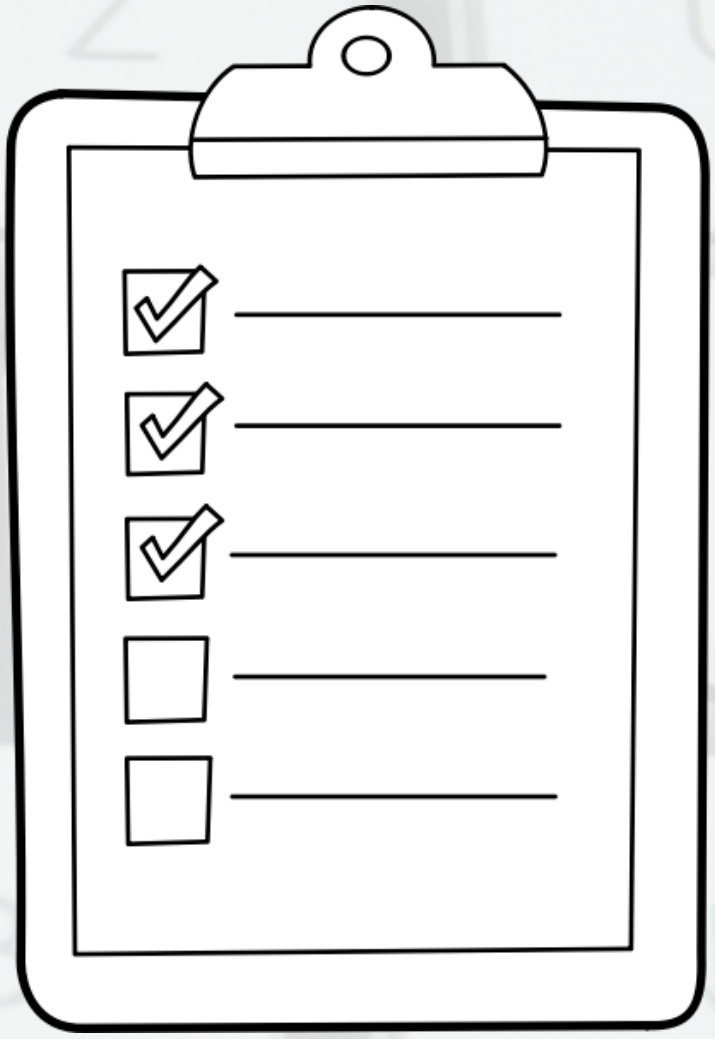
See handout

Agenda

Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6 	I/O including Helpful NullPointers
7	Modules
8	Other topics + open Q&A

Lab Review

- Any questions?
- Solution review



Getting started

File	Class	I/O
Path	Interface	NIO.2
Paths	Class	NIO.2
Files	Class	NIO.2

Common ways to create a Path

```
Path path1 = Paths.get("files","kcdc.txt");

Path path2 = Paths.get("files/kcdc.txt");

Path path3 = Path.of("files/kcdc.txt");

Path path4 = new File("files/kcdc.txt").toPath();
```

/ vs \

Side note - NullPointer


```
Path path = null;
// lots of code here
System.out.println(path.toAbsolutePath());
```

```
% java11 HelpfulNullPointer.java
Exception in thread "main" java.lang.NullPointerException
    at HelpfulNullPointer.main(HelpfulNullPointer.java:9)

% java17 HelpfulNullPointer.java
Exception in thread "main" java.lang.NullPointerException: Cannot invoke
"java.nio.file.Path.toAbsolutePath()" because “path" is null
    at HelpfulNullPointer.main(HelpfulNullPointer.java:9)
```

Reading a File - Originally

```
File file = new File("files/kcdc.txt");
BufferedReader reader = null;
try {
    reader = new BufferedReader(new FileReader(file));
    String line = null;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} finally {
    if (reader != null) {
        reader.close();
    }
}
```

Try with Resources

```
File file = new File("files/kcdc.txt");
try (BufferedReader reader = new BufferedReader(
     new FileReader(file))) {

     String line = null;
     while ((line = reader.readLine()) != null) {
          System.out.println(line);
     }
}
```

3 ways with NIO.2

```
Path path = Path.of("files/kcdc.txt");

String str = Files.readString(path);
List<String> list = Files.readAllLines(path);
String strOld = new String(Files.readAllBytes(path),
   Charset.defaultCharset());

System.out.println(str);
list.forEach(System.out::println);
System.out.println(strOld);
```

Fixed with UncheckedIOException

```
public String readCombinedValues() throws IOException{
    Path path = Path.of("files");
    return Files.list(path)
        .map(this::readAsString)
        .collect(Collectors.joining(","));


}

private String readAsString(Path path) {
    try {
        return Files.readString(path);
    } catch(IOException e) {
        throw new UncheckedIOException(e);
    }
}
```

What's wrong?

```
Files.lines(path).forEach(System.out::println);
```

Resource leak :(

```
try (Stream<String> stream = Files.lines(path)) {
    stream.forEach(System.out::println);
}
```

What's wrong?

```
public String readCombinedValues() {
    Path path = Path.of("files");
    return Files.list(path)
        .map(this::readAsString)
        .collect(Collectors.joining(","));

}

private String readAsString(Path path) {
    return Files.readString(path);
}
```

Files list() and readString() throw checked IOException

Writing a File - IO

```
File file = new File("kcdc.txt");
try (BufferedWriter writer = new BufferedWriter(
  new FileWriter(file))) {
    writer.write("Hello\n");
}
try (PrintWriter writer = new PrintWriter(
  new FileWriter(file))) {
    writer.println("Hello\n");
}
try (BufferedOutputStream stream =
   new BufferedOutputStream(
   new FileOutputStream(file))) {
    stream.write("Hello\n".getBytes(
      Charset.defaultCharset()));
}
```

Writing a File - NIO.2

```
Path path = Path.of(“kcdc.txt”);

Files.write(path, “hello".getBytes(
    Charset.defaultCharset()));

Files.writeString(path, "hello");

Files.write(path, List.of("hello"));
```

Writing a File - Common Options

```
Files.writeString(path, "hello",
    StandardOpenOption._________);
```

APPEND	Adds to file
CREATE	Creates new file if doesn't exist
CREATE_NEW	Creates new file if exists. Otherwise fails

Defaults to creating if doesn't exist and overwriting if does

New Files.mismatch()

12

```
Path kcdc = Path.of("files/kcdc.txt");
Path kc = Path.of("files/kc.txt");

System.out.println(Files.mismatch(kcdc, kc));
System.out.println(Files.mismatch(kcdc, kcdc));
```

11 (index of first character different)
-1 (same file contents regardless of whether exists)

Other Useful Files methods

isSameFile()	Resolve to same file
createDirectory() createDirectories()	The later adds interm folders
copy()	Copy path/input stream to file/directory
move()	Move between paths
delete() deleteIfExists()	The former throws exception
newBufferedReader() newBufferedWriter()	Interact with legacy I/O
walk()	Visit all files in tree

A number of these take optional varargs to customize

Useful Path methods

toFile()	Legacy file
toAbsolutePath()	/ or c: or …
getParent()	One level up
getRoot()	Top level
normalize()	Simplifies . and ..
relativize(Path)	Relative path from object to parameter (both must be absolute or relative)
resolve(Path)	Add parameter to path of object (if absolute path passed in, object ignored)

Types of I/O

- Byte vs character
- Input vs output
- Low level vs high level

Examples

- ByteArrayInputStream
- FileWriter
- ObjectOutputStream

Console

```
Console c = System.console();
String line = c.readLine("Name?");
String formatting = c.readLine("%s?", "Name");
char[] password = c.readPassword();
char[] pwd = c.readPassword("%s?", “Secrets");

c.printf("%d", 1);
c.format("%b", true);
```

Console can be null!

reader() and writer()
for direct access

Module 6 - Question 1

What does the following output?

```
Path clock = new Path("time/device/../clock");
Path phone = new Path("time/phone");
System.out.println(clock.relativize(phone));
```

A. phone

B. ../phone

C. time/device/phone

D. time/device/../clock/..phone

E. None of the above

Module 6 - Question 2

What does the following output?

```
Path clock = Paths.get(“time/device/../clock”);
Path phone = Path.of(“time/phone");
System.out.println(clock.relativize(phone));
```

A. phone

B. ../phone

C. time/device/phone

D. time/device/../clock/..phone

E. None of the above

Module 6 - Question 3

Which are true?

```
Path path = Path.of("file");
System.out.println(Files.mismatch(path, path));
```

A. If the file exists, prints -1

B. If the file exists, throws exception

C. If the file does not exist, prints -1

D. If the file does not exist, throws exception

E. Does not compile

Module 6 - Question 4

Which are true if file1 contains “howdy”?

```
Path path1 = Path.of("file1");
Path path2 = Path.of("file2");
System.out.println(Files.mismatch(path1, path2));
```

A. If file2 contains “howdy”, prints -1

B. If file2 contains “howdy”, prints 0

C. If file2 does not exist, prints -1

D. If file2 does not exist, throws exception

E. Does not compile

Module 6 - Question 5

Which are true if file1 contains “howdy”?

```
Path path1 = Path.of("file1");
Path path2 = Path.of("file2");
System.out.println(Files.mismatch(path1, path2));
```

A. If file2 contains “hello”, prints -1

B. If file2 contains “hello”, prints 0

C. If file2 contains “hello”, prints 1

D. If file2 contains “uh”, prints -1

E. If file2 contains “uh”, prints 0

Module 6 - Question 6

How many problems are in this code?

```
public static void main(String[] args) {
    Files.lines(Path.of("lamp"))
            .filter(String::isEmpty)
            .collect(Collectors.toSet())
            .forEach(System.out::println);
}
```

A. 0

B. 1

C. 2

D. 3

Module 6 - Question 7

What is true of this code?

```
Console console = System.console();
char[] secret = console.readPassword();
System.out.println(secret);
```

A. Guaranteed to print password

B. Guaranteed not to print password

C. Guaranteed to complete successfully

D. Does not compile

E. May throw an unchecked exception

Module 6 - Question 8

Which of these are meant for binary data?

A. FileInputStream

B. FileReader

C. InputStreamReader

D. ObjectOutputStream

E. PrintWriter

Module 6 - Question 9

Which of these are meant for output?

A. FileInputStream

B. FileReader

C. InputStreamReader

D. ObjectOutputStream

E. PrintWriter

Module 6 - Question 10

Which of these are low level?

A. FileInputStream

B. FileReader

C. InputStreamReader

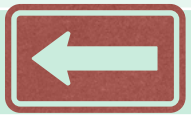
D. ObjectOutputStream

E. PrintWriter

Lab & Break

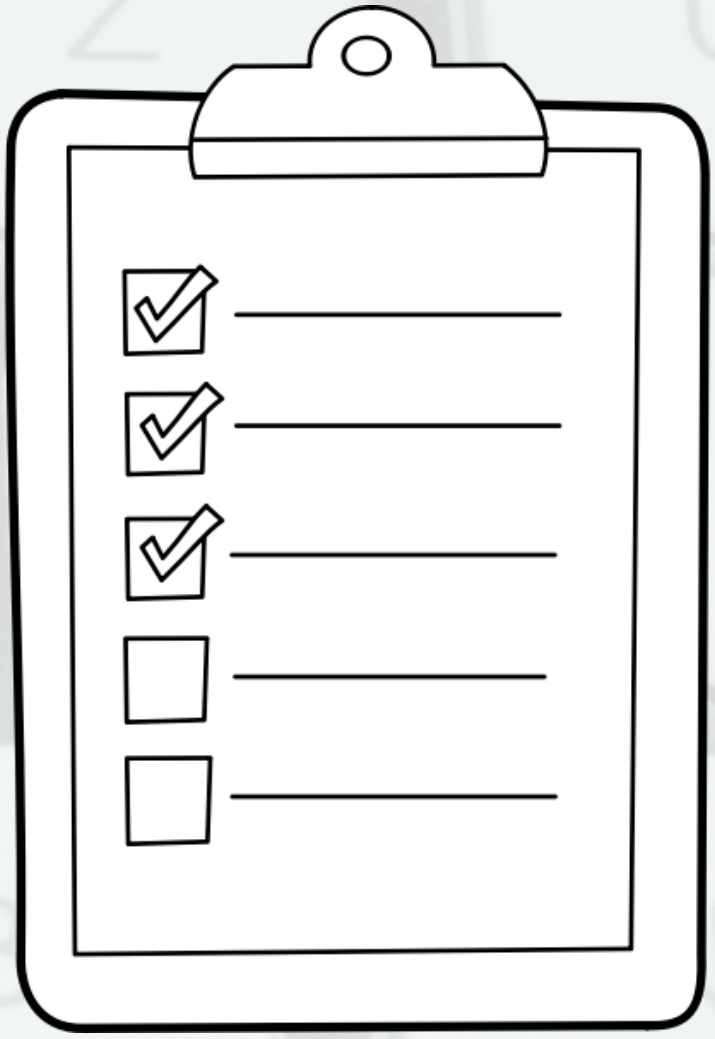
See handout

Agenda

Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7 	Modules
8	Other topics + open Q&A

Lab Review

- Any questions?
- Solution review



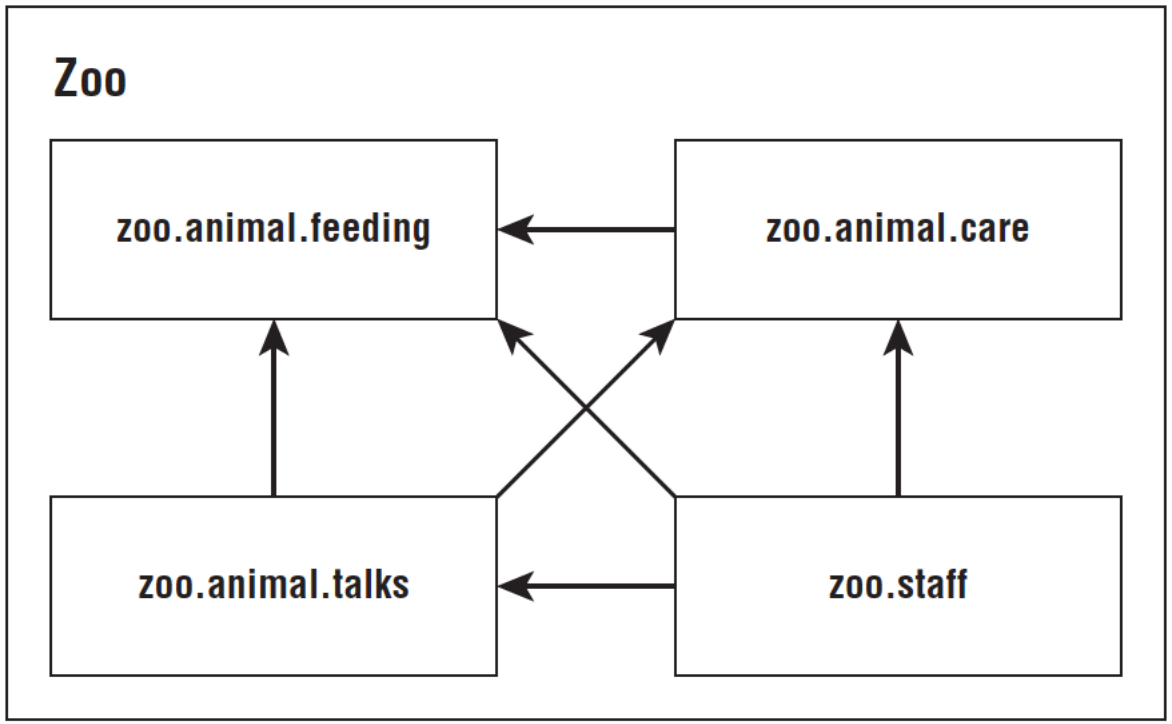
Background

- Code replies on dependencies
  - Open source jar
  - Commercial jar
  - Internal jar
  - JDK APIs
- JAR Hell

Benefits

- Specify what use
- No circular dependencies
- Unique package enforcement
- Custom Java builds/packages
- Better access control
  ("Private packages')

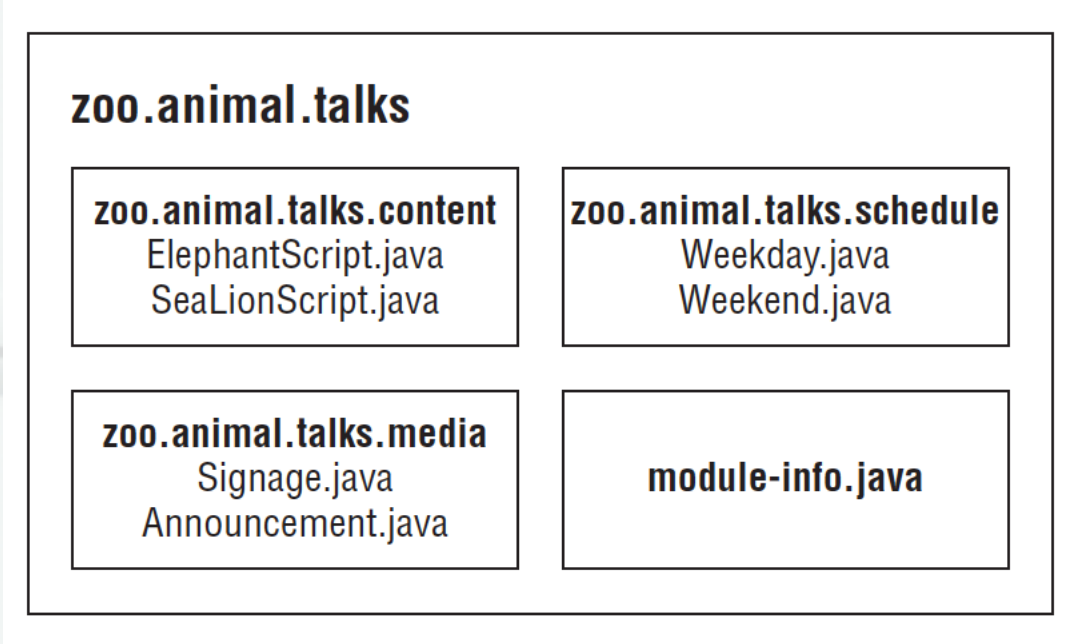
Designing with Modules



No cycles

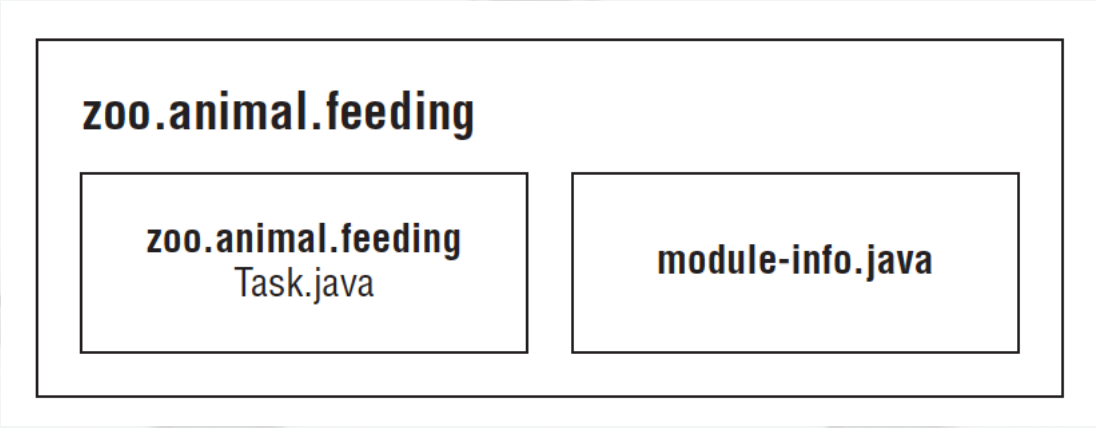
What is a module?

- One or more packages
- module-info.java file



Small module

- One package
- module-info.java file



Simplest module

- module-info.java file in module root directory

```
module zoo.animal.feeding {

}
```

Module name (periods common like packages)

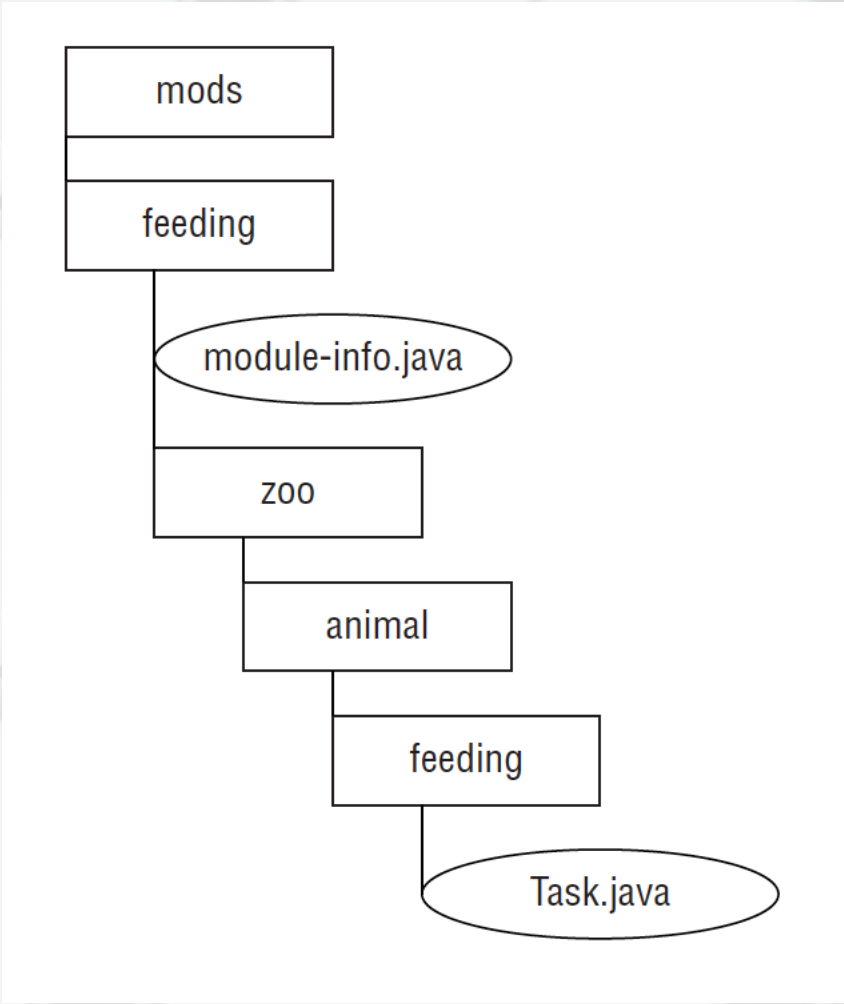
"module" keyword

Technically no packages required

Adding a class

```
package zoo.animal.feeding;

public class Task {

    public static void main(String... args) {
        System.out.println("All fed!");
    }
}
```

Directory Structure



Running and Packaging

```
javac --module-path mods -d feeding
    feeding/zoo/animal/feeding/*.java
    feeding/module-info.java
```

```
java --module-path feeding --module
    zoo.animal.feeding/zoo.animal.feeding.Task
```

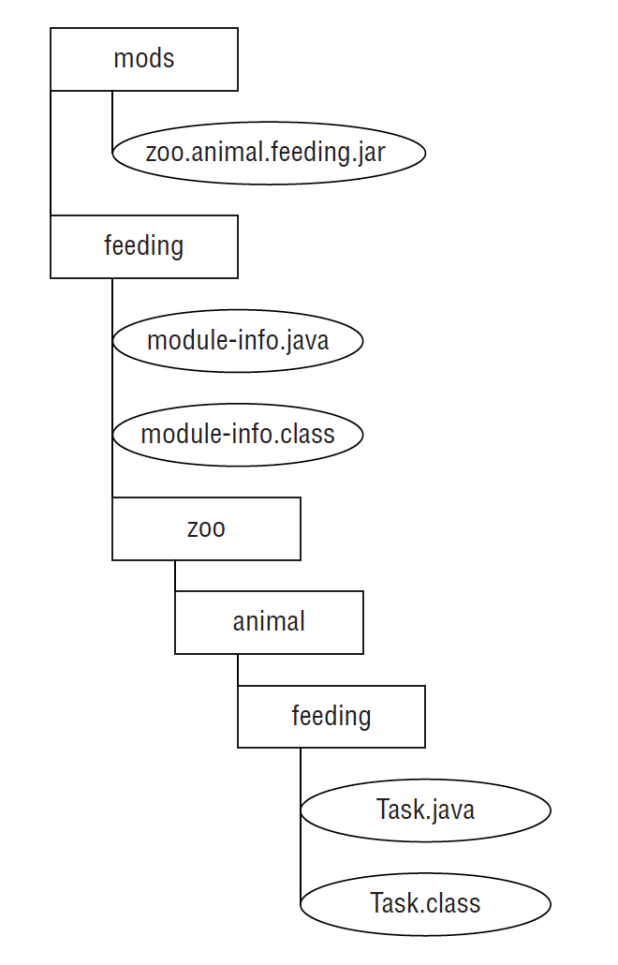
module name

fully qualified class name

```
jar -cvf mods/zoo.animal.feeding.jar -C feeding/ .
```

-p is module path
-m is module

Directory Structure



Exports Directive

```
module zoo.animal.feeding {
 exports zoo.animal.feeding;
}
```

Now package available outside of module

Adding a second module

```
module zoo.animal.care {

  exports zoo.animal.care.medical;

  requires zoo.animal.feeding;
}
```

Declares dependency on feeding

Now we can use the class

```
package zoo.animal.care.details;

import zoo.animal.feeding.*;

public class HippoBirthday {

    private Task task;
}
```

Sub-package

Levels of access control

Level	Within module	Outside module
private	Accessible only within class	No access
default (package/ package-private)	Accessible only within package	No access
protected	Accessible only within package or to subclasses	Accessible to subclasses only if exported
public	Accessible to all classes	Accessible only if package exported

Requires Transitive

- Superset of requires
- Adds all modules the target exports
- Cannot have duplicate requires statements (or requires and requires transitive)
- Can require something that is in the other modules exports list

Reviewing Directives

exports	Make package available
requires	Depends on this module
requires transitive	Depends on this module and everything it requires

More Directives

provides	Provides an implementation of a service provides zoo.staff.ZooApi with zoo.staff.ZooImpl
uses	Uses a service uses zoo.staff.ZooImpl
opens	Allows reflection opens zoo.animal.talks.schedule; opens zoo.animal.talks.media to zoo.staff;

Module 7 - Question 1

Is this is a valid module?

```
zoo.staff
|---zoo
  |-- staff
    |-- Vet.java
```

A. Yes if module-info was added under zoo.staff

B. Yes if module-info was added under zoo

C. Yes if module-info was added under staff

D. None of these make it valid

Module 7 - Question 2

Which fills in the blank if animal.behavior is a package?

```
module animal {
  _____ animal.behavior;
}
```

A. export

B. exports

C. require

D. requires

Module 7 - Question 3

Which fills in the blank if animal.behavior is a module?

```
module animal {
  _____ animal.behavior;
}
```

A. export

B. exports

C. require

D. requires

Module 7 - Question 4

Fill in the blanks to compile

```
java
____ zoo.animal.talks/zoo/animal/talks/Peacocks
____ modules
```

A. -d and -m

B. -d and -p

C. -m and -d

D. -m and -p

E. None of the above

Module 7 - Question 5

Which are true of this module? (Choose two)

```
module com.food.supplier {}
```

A. All packages are automatically exported

B. No packages are automatically exported

C. A main method can be run

D. A main method cannot be run

Module 7 - Question 6

Which is a legal command to run a program in a module? (Choose all)

A. java -p x -m x/x

B. java -p x-x -m x/x

C. java -p x -m x-x/x

D. java -p x -m x/x-x

E. java -p x -m x.x

F. java -p x.x -m x.x

Module 7 - Question 7

Which best fills in the blank?

```
module _____ {
  exports com.unicorn.horn;
  exports com.unicorn.magic;
}
```

A. com

B. com.unicorn

C. com.unicorn.horn

D. The code does not compile

E. None of these is a good choice

Module 7 - Question 8

Which is the first to have a compiler error?

```
1: module snake {
2: exports com.snake.tail;
3: exports com.snake.fangs to bird;
4: requires skin;
5: requires transitive skin;
6: }
```

A. 1

B. 3

C. 5

D. None of the above

Module 7 - Question 9

Which of the following would be a legal module name? (Choose all that apply)

A. com.book

B. com-book

C. com.book$

D. com-book$

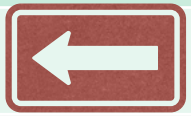
E. 4com.book

F. 4com-book

Lab & Break

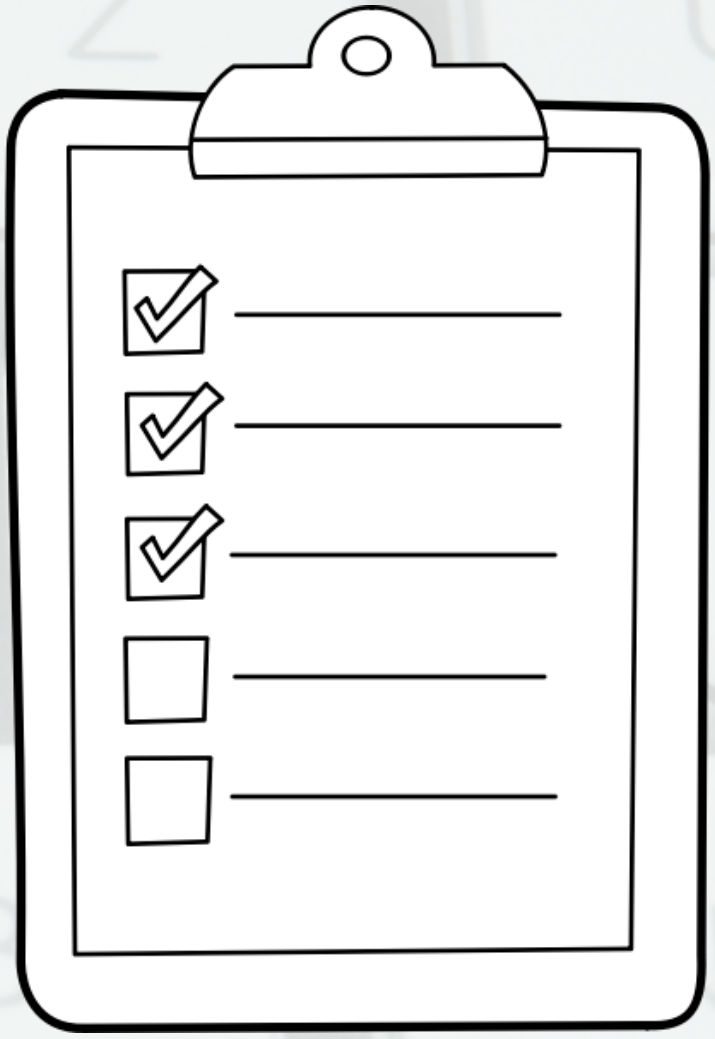
See handout

Agenda

Module	Topics
1	Intro & Changes to Strings
2	Switch expressions & Pattern matching
3	Records and Sealed classes
4	Streams including Collectors.teeing
5	Localization/Formatting including CompactNumberFormatting
6	I/O including Helpful NullPointers
7	Modules
8 	Other topics + open Q&A

Lab Review

- Any questions?
- Solution review



Using var (Local Variable Type Inference)

```
String name1 = "Jeanne";
var name2 = "Jeanne";

List list1 = List.of(1, 2, 3);
var list2 = List.of(1, 2, 3);
```

Syntactic sugar/less boilerplate

Can’t change type

```
String name = "Jeanne";
name = 1;  // DOES NOT COMPILE
```

Not immutable

```
String name = "Jeanne";
name = "Boyarsky";
```

Readablity

```
List<WebElement> headers =
    thead.findElements(By.tagName("th"));
VolunteerDashboardRow headerRow =
    new VolunteerDashboardRow(headers);
```

vs

```
var headers = thead.findElements(By.tagName("th"));
var headerRow = new VolunteerDashboardRow(headers);
```

What types are these?

```
var csvPath = createAndGetFile(CSV_DATA);

try (var csvWriter = Files.newBufferedWriter(csvPath);
     var csvPrinter = new CSVPrinter(
         csvWriter, CSVFormat.DEFAULT)) {
}
```

Evolution

```
Map<Integer, String> productMap1 =
    new HashMap<Integer, String>();

Map<Integer, String> productMap2 = new HashMap<>();

var productMap3 = new HashMap<Integer, String>();
```

Where can we use?

```
var name = "Jeanne";
var other = name + 2;

var list = List.of(1, 2, 3);
for (var num : list) {}
```

Where can't we use?

```
var instance = 1;

class Inner { var bad = "abc"; }

public static void main(String[] args) {
    var noGood;
    noGood = 4;
}
```

What's wrong?

Does this work?

```
var var = "var";
```

Yes, but please don't!

Tradeoffs

Pros	Cons
Less typing	Loss of information
Less redundancy	Variable names matter more
Can scan variable names	Be careful!

http://openjdk.java.net/projects/amber/LVTIstyle.html

Lambdas

```
Predicate<String> pred1 = p -> true;
Predicate<String> pred2 = (String p) -> true;
Predicate<String> pred3 = (var p) -> true;
```

Annotations

```
BiPredicate<Map<Integer, String>, List<String>> func =
        (@NotNull var map, var list) -> true;
```

All or nothing

```
// good
BiPredicate<Map<Integer, String>, Boolean> bi1 =
    (var map, var list) -> true;


// bad
BiPredicate<Map<Integer, String>, Boolean> bi2 =
    (var map, list) -> true;
BiPredicate<Map<Integer, String>, Boolean> bi3 =
    (var map, List list) -> true;
```

Effectively final

If typed final before param/local variable, would it still compile?

Which are effectively final?

```
int numChairs = 4;
int numLegs = numChairs * 4;

numChairs++;
System.out.println(numChairs);
```

Just numLegs

Original code

```
Path path = Paths.get("file");
try (BufferedReader reader =
                Files.newBufferedReader(path)) {
    // read file
}
```

With var

```
var path = Paths.get("file");
try (var reader = Files.newBufferedReader(path)) {
    // read file
}
```

Before try

```
var path = Paths.get("file");
var reader = Files.newBufferedReader(path);
try (reader) {
    // read file
}
```

Works because path is effectively final

What's wrong here?

```
Connection con =
        DriverManager.getConnection(url);
PreparedStatement ps =
        con.prepareStatement(sql);
ps.setInt(1, id);
ResultSet rs = ps.executeQuery();

try (con; ps; rs) {
    while (rs.next()) {
        // process result set
    }
}
```

Resource leak!

Module 8 - Question 1

Which are true if the resources are ResultSets?

```
try (rs1, rs2) {
}
```

A. rs1 is closed before rs2

B. rs2 is closed before rs1

C. Neither is closed since declared before try

D. The code does not compile

Module 8 - Question 2

Which are true if the resources are ResultSets?

```
try (rs1; rs2) {
}
```

A. rs1 is closed before rs2

B. rs2 is closed before rs1

C. Neither is closed since declared before try

D. The code does not compile

Module 8 - Question 3

What is the result of the following?

```
var a = 1;
var b = a;
int c = b;
System.out.println(c);
```

A. a does not compile

B. b does not compile

C. c does not compile

D. 1

E. None of the above

Module 8 - Question 4

What is the result of the following?

```
var a = 1;
var b = a;
double c = b;
System.out.println(c);
```

A. a does not compile

B. b does not compile

C. c does not compile

D. 1.0

E. None of the above

Module 8 - Question 5

What is the result of the following?

```
var path = Paths.get("file");
var reader = Files.newBufferedReader(path);
reader = Files.newBufferedReader(path);
try(reader) {
    // read file
}
```

A. reader is closed

B. reader remains open

C. None; the code does not compile

Module 8 - Question 6

What is the result of the following?

```
var a = 1;
var b = a;
String c = b;
System.out.println(c);
```

A. a does not compile

B. b does not compile

C. c does not compile

D. 1

E. None of the above

Module 8 - Question 7

Which variables do not compile? (Choose all)

```
var a = null;
var b = 1;
var c;
var var = 3;
```

A. a

B. b

C. c

D. All compile

Module 8 - Question 8

Which do not compile? (Choose all)

A. var a; a = 1;

B. static var b = 1;

C. int x=1, var y=1;

D. var VAR = 1;

E. All compile

Module 8 - Question 9

Which allow using var?

A. Instance variables
B. Lambda variables
C. Static variables
D. Try with resources

Module 8 - Question 10

I learned a lot today

A. True

B. False

Fun (not on exam)

- Java 11 - Unicode 11
- Java 17 - Unicode 17

```
public class Fun {

   public static void main(String[] args) {
      System.out.println("""
         Wear \uD83E\uDD7D \
         and \uD83E\uDD7C \
         when using a \uD83E\uDDEA""");
   }
}
```

```
java17 Fun.java
Wear 🥽 and 🥼 when using a 🧪
```

Book Giveaway

@jeanneboyarsky