Library and Language Improvements

Sander Mak
FELLOW & SOFTWARE ARCHITECT
@Sander_Mak

Collections: The Old Way

```
List<String> books = new ArrayList<>();
books.add("Java 9 Modularity");
books.add("Designing Data-Intensive Applications");
books.add("Java 8 Lambdas");
```

Verbose

Won't work as field initializer

Collections: The Old Way

```
List<String> books =
   Arrays.asList("Java 9 Modularity", ..);
```

Arrays, really?
Lists only, no Set construction

```
List<String> books =
   new ArrayList<>() {{ add("Java 9 Modularity"); }};
```

```
List<String> books =
   Collections.emptyList();
```

Only for empty collections

Collection Factory Methods

List/Set Factory Methods

Collection Factory Methods

```
List.of()
List.of(E e1)
List.of(E... elements)
List.of(E e1, E e2)
List.of(E e1, E e2, E e3)
List.of(E e1, E e2, E e3, E e4)
List.of(E e1, E e2, E e3, E e4, E e5)
List.of(E e1, E e2, E e3, E e4, E e5, E e6)
List.of(E e1, E e2, E e3, E e4, E e5, E e6, E e7)
List.of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8)
List.of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9)
List.of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9, E e10)
```

Intermediate array allocation

Collection Factory Methods

Map Factory Methods

Collection Factory Methods

```
Map.of(K key, V value)
Map.of(K key1, V value1, K key2, V value2)
...
```

Up to 10 key/values

```
Map.ofEntries(Map.Entry<K, V>... entries)
```

Unbounded

Iteration order not guaranteed

Stream API Improvements

Quick refresher

Collection to single-use Stream

Stream pipeline with terminal operation

```
List<Integer> intList =
          List.of(1, 2, 3, 4);


Stream<Integer> ints =
          intList.stream();


ints.map(i -> i + 1)
    .filter(i -> i < 4)
    .forEach(System.out::println);
```

Stream API Improvements

Added methods:

```
Stream<T> takeWhile(Predicate<? super T> predicate)

Stream<T> dropWhile(Predicate<? super T> predicate)

static Stream<T> ofNullable(T t)

static Stream<T> iterate(T seed,
                         Predicate<? super T> hasNext,
                         UnaryOperator<T> next)
```

Stream API Improvements

```
Stream<T> takeWhile(Predicate<? super T> predicate)
```

isBlue?

```
Stream<T> dropWhile(Predicate<? super T> predicate)
```

isBlue?

Use with ordered Streams

Demo

New Stream methods

```
public class Book {
  public final String title;
  public final Set<String> authors;
  public final double price;

  public Book(String title, Set<String>
              authors, double price) {
      this.title = title;
      this.authors = authors;
      this.price = price;
  }

  // getters, toString, getBook, getBooks
}
```

Stream API: New Collectors

Quick refresher

Materialize stream into new collection

```
List<Integer> ints =
    Stream.of(1,2,3)
          .map(n -> n + 1)
          .collect(Collectors.toList());
// [2, 3, 4]
```

Advanced collector: Collectors.groupingBy

```
Map<Integer, List<Integer>> ints =
   Stream.of(1,2,3,3)
         .collect(groupingBy(i -> i % 2,
                             toList()));
// {0=[2], 1=[1, 3, 3]}
```

Stream API: New Collectors

```
public static <..> Collector<..> filtering(
    Predicate<..> predicate, Collector<..> downstream)
```

```
Map<Set<String>, Set<Book>> booksByAuthors =
  books.collect(
          groupingBy(Book::getAuthors,
                     filtering(b -> b.getPrice() > 10,
                               toSet())
          )
  );
```

Stream API: New Collectors

```
public static <..> Collector<..> flatMapping
        (Function<..> mapper, Collector<..> downstream)
```

```
Map<Double, Set<String>> authorsSellingForPrice =
  books.collect(
          groupingBy(Book::getPrice,
                     flatMapping(b -> b.getAuthors().stream(),
                                 toSet())
          )
  );
```

Additions to Optional

Quick refresher

Holds single value

Or no value
No more nulls!

Transform inside Optional

```
Optional<String> s =
          Optional.of("Hi");

Optional<Integer> i =
          Optional.empty();


s.map(String::toUpperCase); // "HI"
i.map(n -> n + 1); // Still empty
```

Additions to Optional

Added methods:

```
void ifPresentOrElse(Consumer<T> action,
                     Runnable emptyAction)
```

```
Optional<T> or(Supplier<Optional<T>> supplier)
```

```
Stream<T> stream()
```

```
Stream<Optional<Integer>> optionals = Stream.of(
   Optional.of(1), Optional.empty(), Optional.of(2));

Stream<Integer> ints = optionals.flatMap(Optional::stream);

ints.forEach(System.out::println);
```

```
1
2
```

Optional.stream()

Interoperability between Optional and Stream

```
Set<String> authors = Book.getBooks()
        .map(book -> book.authors.stream().findFirst())
        .flatMap(optAuthor -> optAuthor.stream())
        .collect(Collectors.toSet());
```

Optional.stream()

Interoperability between Optional and Stream

Stream.flatMap + Optional.stream =

Demo

New Optional methods

`ifPresentOrElse`

`or`

Small Language Changes

Underscore as identifier illegal

Improved try-with-resources

Better generic type inference for anonymous classes

Private interface methods

Small Language Changes: Underscore

Deprecated in Java 8

Illegal in Java 9

```
public class NoUnderscore {
  String _ = "underscore";
}
```

```
error: as of release 9, '_' is a
keyword, and may not be used as an
identifier
```

Possible future use: `list.forEach(_ -> doSomething())`

Small Language Changes: Try-with-resources

```
try (FileInputStream fis = new FileInputStream("~/tmp/test")) {
    fis.read();
}
```

Direct instantiation

Small Language Changes: Try-with-resources

```
public void doWithFile(FileInputStream fis) throws IOException {
  try (                                    ) {
    .read();
  }
}
```

Java 8: new variable required

Small Language Changes: Try-with-resources

```
public void doWithFile(FileInputStream fis) throws IOException {
  try (fis) {
    fis.read();
  }
}
```

Java 9: can use effectively final variables

Small Language Changes: Better Inference

Java ~~7/8~~

```
ArrayList<String> list = new ArrayList<>();
```

```
ArrayList<String> list3 = new ArrayList<>() {
    @Override
    public boolean add(String s) {
        System.out.println("Adding " + s);
        return super.add(s);
    }
};
```

Java 9

```
error: cannot infer type arguments for ArrayList<E>
reason: cannot use '<>' with anonymous inner classes
```

Small Language Changes: Private Interface Methods

```
public class Book implements PricedObject {

    String title;
    double price;

    public Book(String title, double price) {
        this.title = title;
        this.price = price;
    }

    @Override
    public double getPrice() {
        return price;
    }
}
```

```
public interface PricedObject {

 double getPrice();

}
```

Small Language Changes: Private Interface Methods

```
public class Book implements PricedObject {

    String title;
    double price;

    public Book(String title, double price) {
        this.title = title;
        this.price = price;
    }

    @Override
    public double getPrice() {
        return price;
    }
}
```

```
public interface PricedObject {

 double getPrice();

 default double getPriceWithTax() {
   return getPrice() * 1.21;
 }

}
```

```
new Book("title", 10).getPriceWithTax()
```

Small Language Changes: Private Interface Methods

```
public interface PricedObject {

 double getPrice();

 default double getPriceWithTax() {
   return getPrice() * 1.21;
 }

 default double getOfferPrice(double discount) {
   return getPrice() * 1.21 * discount;
 }

}
```

Add helper method?

Default method part of API!

Small Language Changes: Private Interface Methods

```
public interface PricedObject {

 double getPrice();

 default double getPriceWithTax() {
   return getTaxedPriceInternal();
 }

 default double getOfferPrice(double discount) {
   return getTaxedPriceInternal() * discount;
 }

 private double getTaxedPriceInternal() {
   return getPrice() * 1.21;
 }

}
```

```
new Book("a", 1).getPriceWithTax()
```

```
new Book("b", 1).getOfferPrice(0.9)
```

```
new Book("c", 1)
      .getTaxedPriceInternal()
```

Other Improvements: Javadoc

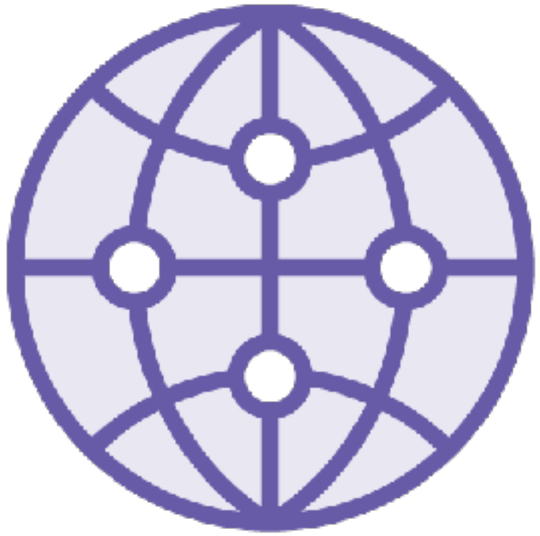
HTML5 compliant

Search

Modules

http://download.java.net/java/jdk9/docs/api/

Other Improvements: Localization



Unicode 8.0: 10.000+ new characters

Properties files: ISO-8859-1 to UTF-8

Common Locale Data Repository

```
java.locale.providers=COMPAT,CLDR,...
```

Other Improvements: java.time

Added methods:

Friendly reminder: stop using java.util.Date

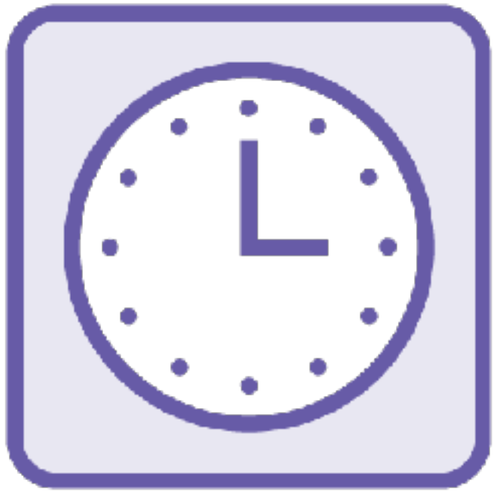
Duration:

```
long dividedBy(Duration divisor)

Duration truncatedTo(TemporalUnit unit)
```

Clock:

```
static Clock systemUTC()
```



Other Improvements: java.time

LocalDate.datesUntil

Summary

Collection factory methods

Stream/Optional extensions

Language changes