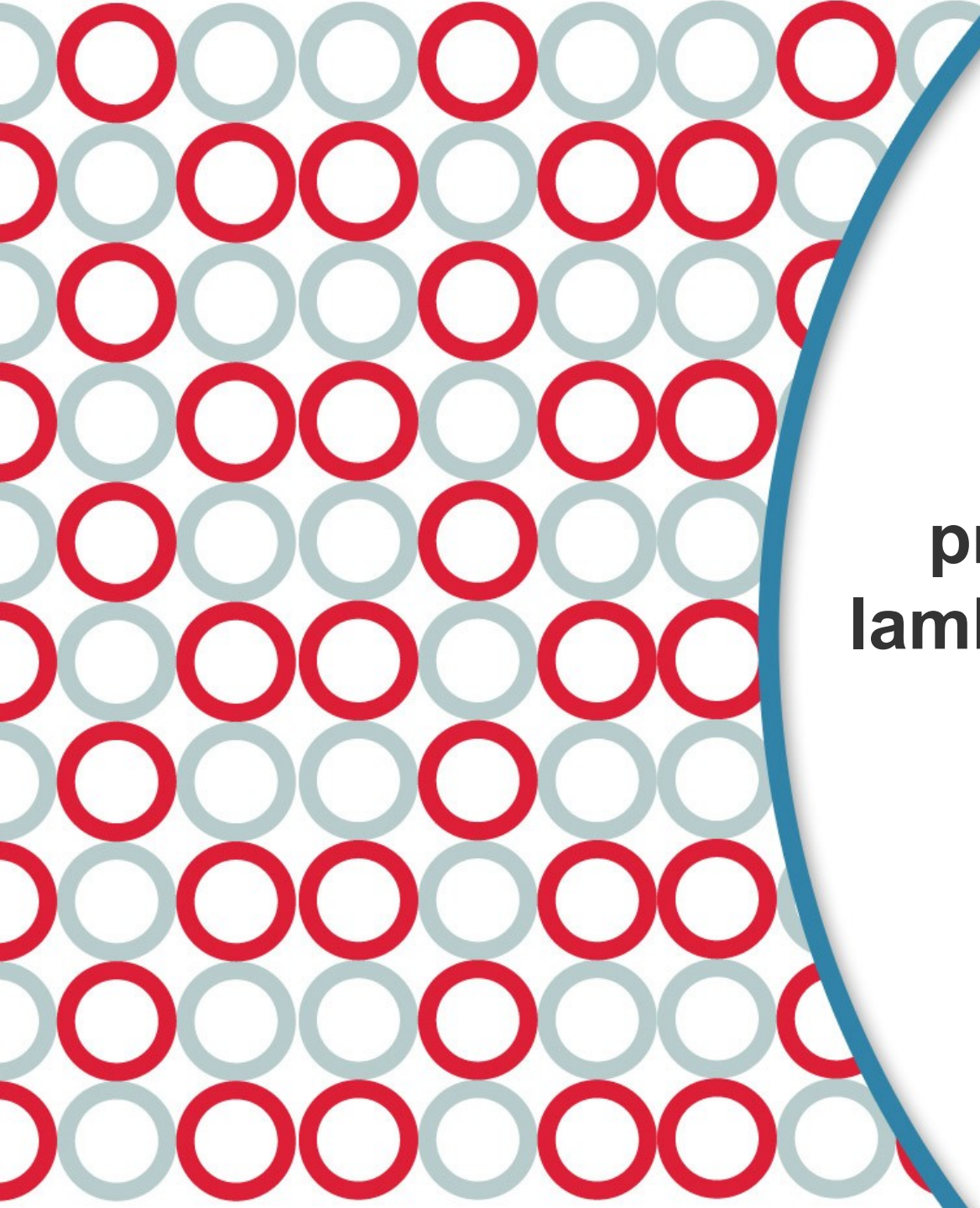

May 2019, IPT Course
Java Web Debelopment

Functional programming and lambda expressions

Trayan Iliev
tiliev@iproduct.org
http://iproduct.org

About me

Trayan Iliev

– CEO of IPT – Intellectual Products & Technologies

– Oracle® certified programmer 15+ Y

– end-to-end reactive fullstack apps with Java, ES6/7, TypeScript, Angular, React and Vue.js

– 12+ years IT trainer

– Voxxed Days, jPrime, jProfessionals, BGOUG, BGJUG, DEV.BG speaker

– Organizer RoboLearn hackathons and IoT enthusiast (http://robolearn.org)

Where to Find the Code?

Java Web Development projects and examples are available @ GitHub:

https://github.com/iproduct/course-java-web-development

Agenda for This Session

- Fundamentals
- Functional interfaces
- Method references
- Constructor references

Новости в Java™ 8

- Ламбда изрази и поточно програмиране – пакети **java.util.function** и **java.util.stream**)
- Референции към методи
- Методи по подразбиране и статични методи в интерфейси – множествено наследяване на поведение в Java 8
- Java 8 Data and Time API (JSR 310)
- Разширени възможности за използване на анотации върху Java типове (JSR 308)
- Функционално програмиране в Java 8 с използване на монади (напр. Optional, Stream) – предимства, начин на реализация, основни езикови идиоми, примери

Функционални интерфейси в Java™ 8

- Функционален интерфейс = интерфейс с един абстрактен метод SAM (Single Abstract Method) – @FunctionalInterface
- Примери за функционални интерфейси в Java 8:

```
public interface Comparator<T> {
    int compare(T o1, T o2);
}
public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}
public interface Runnable {
    public void run();
}
public interface Callable<V> {
    V call() throws Exception;
}
```

Ламбда изрази – пакет **java.util.function**

Примери:

```
(int x, int y) -> x + y
() -> 42
(a, b) -> a * a + b * b;
(String s) -> { System.out.println(s); }
book -> book.getAuthor().fullName()
voter -> voter.getAge() >= legalAgeOfVoting
(person1, person2) -> person1.getAge() - person2.getAge()
(song1, song2) ->
song1.getArtist().compareTo(song2.getArtist())
```

Правила за форматирне на ламбда изрази

- Ламбда изразите (функциите) могат да имат произволен брой параметри, които се ограждат в скоби, разделят се със запетаи и могат да имат или не деклариран тип (ако нямат - типът им се извежда от контекста на използване = target typing). Ако са само с един параметър, то скобите не са задължителни.
- Тялото на ламбда изразите се състои от произволен езикови конструкции (statements), разделени с **;** и заградени във фигурни скоби. Ако имаме само една езикова конструкция – израз то използването на фигурни скоби не е необходимо – в този случай стойността на израза автоматично се връща като стойност на функцията.

Пакет **java.util.function**

- **Predicate<T>** – предикат = булев израз представящ свойство на обекта подавано като аргумент
- **Function<A,R>**: функция която приема като аргумент A и го трансформира в резултат R
- **Supplier<T>** – с помощта на **get()** метод всеки път връща инстанция (обект) – фабрика за обекти
- **Consumer<T>** – приема аргумент (метод **accept()**) и изпълнява действие върху него
- **UnaryOperator<T>** – оператор с един аргумент T -> T
- **BinaryOperator<T>** – бинарен оператор (T, T) -> T

Поточно програмиране (1)

Примери:

```
books.stream().map(book ->
    book.getTitle()).collect(Collectors.toList());
books.stream()
       .filter(w -> w.getDomain() == PROGRAMMING)
       .mapToDouble(w -> w.getPrice()) .sum();
document.getPages().stream()
     .map(doc -> Documents.characterCount(doc))
     .collect(Collectors.toList());
document.getPages().stream()
     .map(p -> pagePrinter.printPage(p))
     .forEach(s -> output.append(s));
```

Поточно програмиране (2)

Примери:

```
document.getPages().stream()
        .map(page -> page.getContent())
        .map(content -> translator.translate(content))
        .map(translated -> new Page(translated))
        .collect(Collectors.collectingAndThen(
                Collectors.toList(),
                pages -> new
        Document(translator.translate(document.getTitle()),
pages)));
```

Референции към методи

- Статични методи на клас – Class::staticMethod
- Методи на конкретни обектни инстанции – object::instanceMethod
- Методи на инстанции референрани чрез класа – Class::instanceMethod
- Конструктори на обекти от даден клас – Class::new

```
Comparator<Person> namecomp =
                    Comparator.comparing(Person::getName);

Arrays.stream(pageNumbers).map(doc::getPageContent)
                    .forEach(Printers::print);

pages.stream().map(Page::getContent).forEach(Printers::print);
```

Статични и Default методи в интерфейси

- Методите с реализация по подразбиране в интерфейс са известни още като virtual extension methods или defender methods, защото дават възможност интерфейсите да бъдат разширявани, без това да води до невъзможност за компилация на вече съществуващи реализации на тези интерфейси (което би се получило ако старите реализации не имплементират новите абстрактни методи).
- Статичните методи дават възможност за добавяне на помощни (utility) методи – например factory методи директно в интерфейсите които ги ползват, вместо в отделни помощни класове (напр. Arrays, Collections).

Пример за default и static методи в интерфейс

```
@FunctionalInterface
interface Event {
    Date getDate();
    default String getDateFormatted() {
        return String.format("%1$td.%1$tm.%1$tY", getDate());
    }
    public static <T, U extends Comparable<? super U>>
    Comparator<T> comparing(Function<T, U> getKey) {
        return (c1, c2) -> getKey.apply(c1).compareTo(getKey.apply(c2));
    }
}
Event current = () -> new Date();
System.out.println(current.getDateFormatted());
```

Функционално програмиране и монади

- Понятие за **монада** във функционалното програмиране (теория на категориите) – **Монадата** е множество от три елемента:

  1) Параметризиран тип **M<T>**
  2) "**unit**" функция: **T -> M<T>**
  3) "**bind**" операция: **bind(M<T>, f:T -> M<U>) -> M<U>**

- В Java 8 пример за монада е класът **java.util.Optional<T>**

  1) Параметризиран тип: **Optional<T>**
  2) "**unit**" функции: **Optional<T> of(T value) ,**

     Optional<T> ofNullable(T value)
  3) "**bind**" операция:

     Optional<U> flatMap(Function<? super T,Optional<U>> mapper)

Литература и интернет ресурси

- Oracle tutorial – lambda expressions - http://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
- Java SE 8: Lambda Quick Start - http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Lambda-QuickStart/index.html
- OpenJDK Lambda Tutorial - https://github.com/AdoptOpenJDK/lambda-tutorial

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg