JAVA 1

Lambda

Teme:

- Lambda račun
- Funkcionalno programiranje
- *@Functional* Interface
- Sintaksa lambda izraza
- Java8 lambda izrazi

Lambda račun

- Alonzo Church – matematički sistem za definiranje računskih funkcija
- Po snazi ekvivalentno Turingovom stroju
- Podloga mnogih programskih jezika (Lisp, Haskell...)
- C# - koncept delegata
- Java – koncept *@Functional* Interface

Funkcionalno programiranje

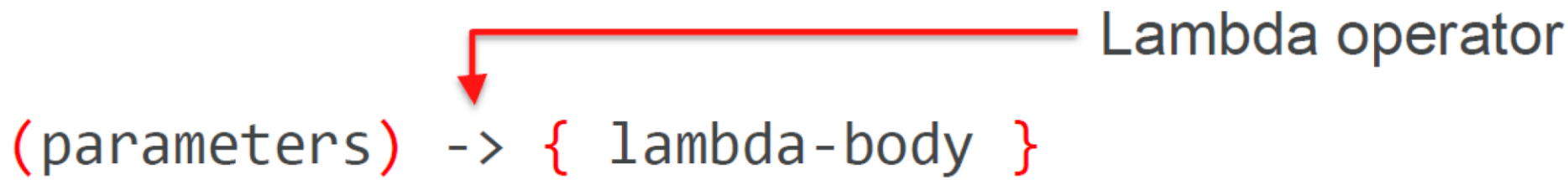
- tretira programiranje kao evaluaciju matematičkih funkcija
- deklarativna programska paradigma
- izbjegava promjenu stanja (*immutable* data)
  - svaki poziv funkcije sa istim argumentima mora dati isti rezultat
  - argumenti su (*effectively*) *final*
- imperativna programska paradigma – mijenjanje stanja
  - problem sa višedretvenosti
- fokus na problemu, a ne na kodu koji rješava problem
- funkcije višeg reda - metode kao argumenti i povratne vrijednosti

@Functional Interface

- *compile-time* anotacija koja osigurava da sučelje ima samo jednu metodu
- može imati drugih metoda, ali samo jednu apstraktnu:
  - *static*
  - *private – od Java9*
  - *default – nužno radi podrške:*

```
public interface Iterable<T> {
    Iterator<T> iterator();
    default void forEach(Consumer<? super T> action) {
        Objects.requireNonNull(action);
        for (T t : this) {
            action.accept(t);
        }
    }
}
```

Sintaksa lambda izraza



- bez parametara – samo zagrade
- jedan parametar – zagrade nepotrebne
- tijelo od jedne linije:
  - zagrade nepotrebne
  - nema return *statement*
- parametri su *effectively final* – modifikator se može izostaviti

Java8 lambda izrazi

- *java.util.function* paket

- Consumer<T> - prima parametar T tipa, vraća void
- Predicate<T> - prima parametar T tipa, vraća boolean
- Function<T, R> - “pretvara” T u R – prima T tip, vraća R tip
- Supplier<T> - “proizvodi” T – ne prima ništa, vraća T tip
  - prikladno za konstruktor

Primitivni tipovi u lamda izrazima

- izbjegavanje skupog *autoboxing-a* i *unboxing-a*

- IntFunction<R> - prima primitiv(int)
- ToIntFunction<T> - vraća primitiv(int)
- IntToDoubleFunction – prima primitiv(int) i vraća primitiv(double)
- ostale varijante primitiva analogno

Binarni oblici lambda izraza

- primanje 2 parametra

- BiConsumer<T, U> - prima T, U parametre, vraća void
- BiPredicate<T, U> - prima T, U parametre, vraća boolean
- BiFunction<T, U, R> - prima T, U parametre, vraća R
- BinaryOperator<T> - prima 2 T parametra, vraća T

Unarni oblici lambda izraza

- primanje i vraćanje istog tipa parametra

- UnaryOperator<T> - prima parametar T, vraća parametar T
- IntUnaryOperator - prima int, vraća int
- IntBinaryOperator – prima 2 int parametra, vraća int

Method references

- lambda - anonimna implementacija jedno metodnog sučelja
- ponekad lambde samo zovu postojeće metode:
  - names.forEach(n -> System.out.println(n));
- stoga se mogu koristiti direktno reference na metode:
  - names.forEach(System.out::println);
- kompaktne, lako čitljive

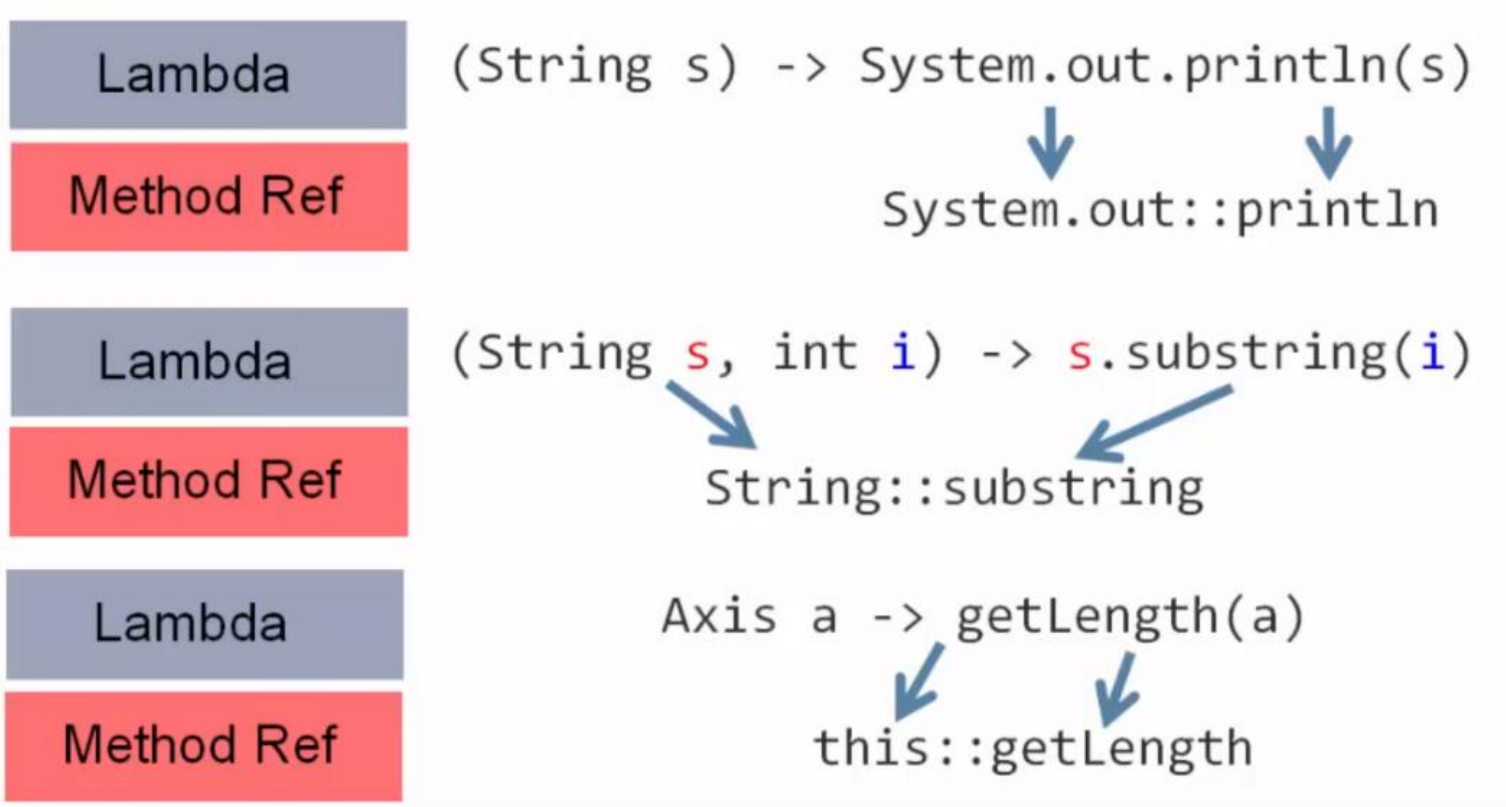
Method references

- 4 osnovna tipa

Kind	Example
Reference to a static method	`ContainingClass::staticMethodName`
Reference to an instance method of a particular object	`containingObject::instanceMethodName`
Reference to an instance method of an arbitrary object of a particular type	`ContainingType::methodName`
Reference to a constructor	`ClassName::new`

Izvor:https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html

Method references



Izvor:https://davidfuentes.blog/2019/03/18/java-8-referencias-a-metodos-y-constructores/

Demo

- Project

Izvor:http://www.jnhsolutions.com/contact-us/request-a-demo/

Hvala na pažnji!