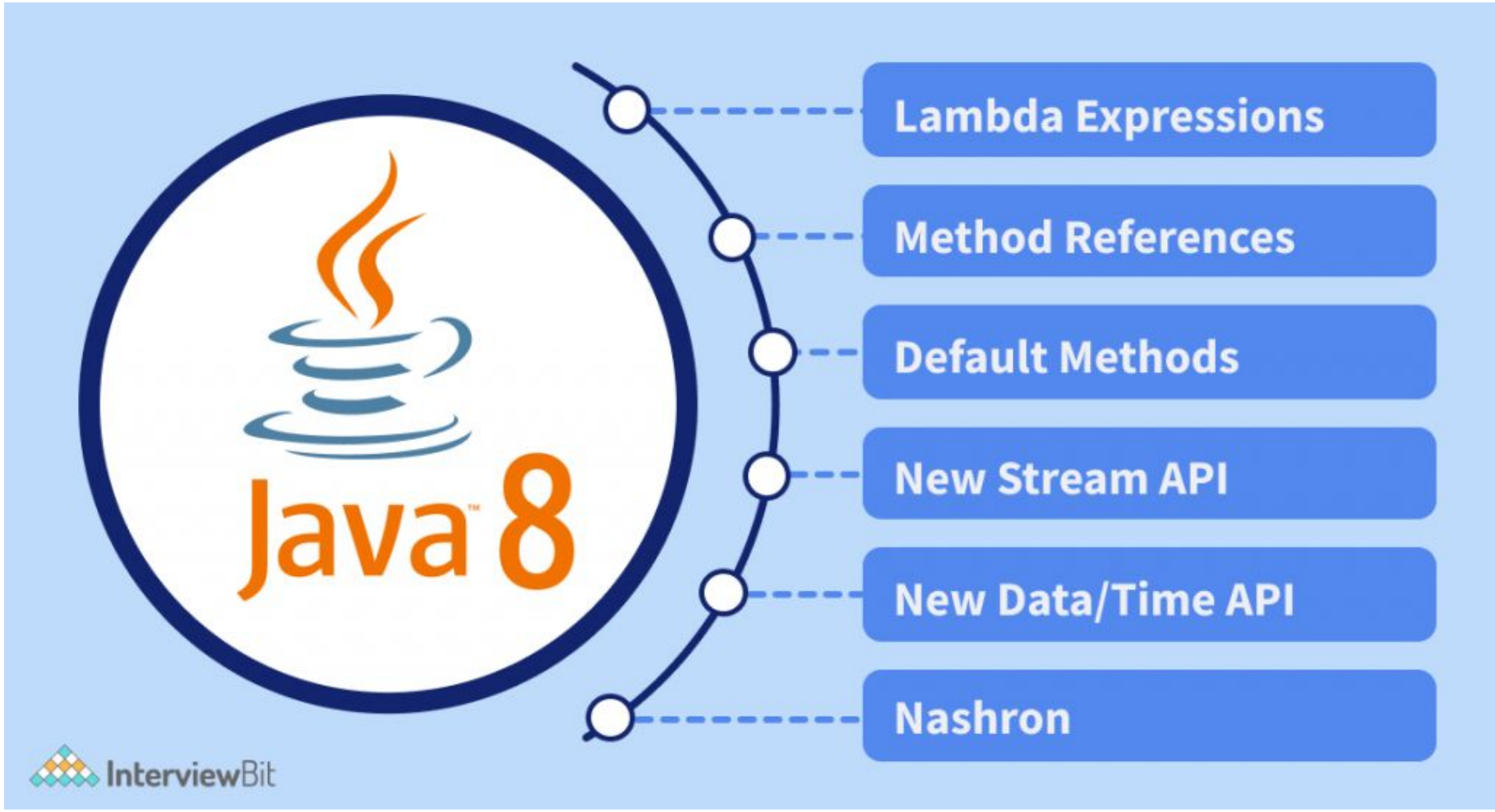
Java Professional
module #3 Lambdas & Stream API

lecture #1 Lambdas, Functional Interfaces, Method references.

Mentor: Khakov Rustam

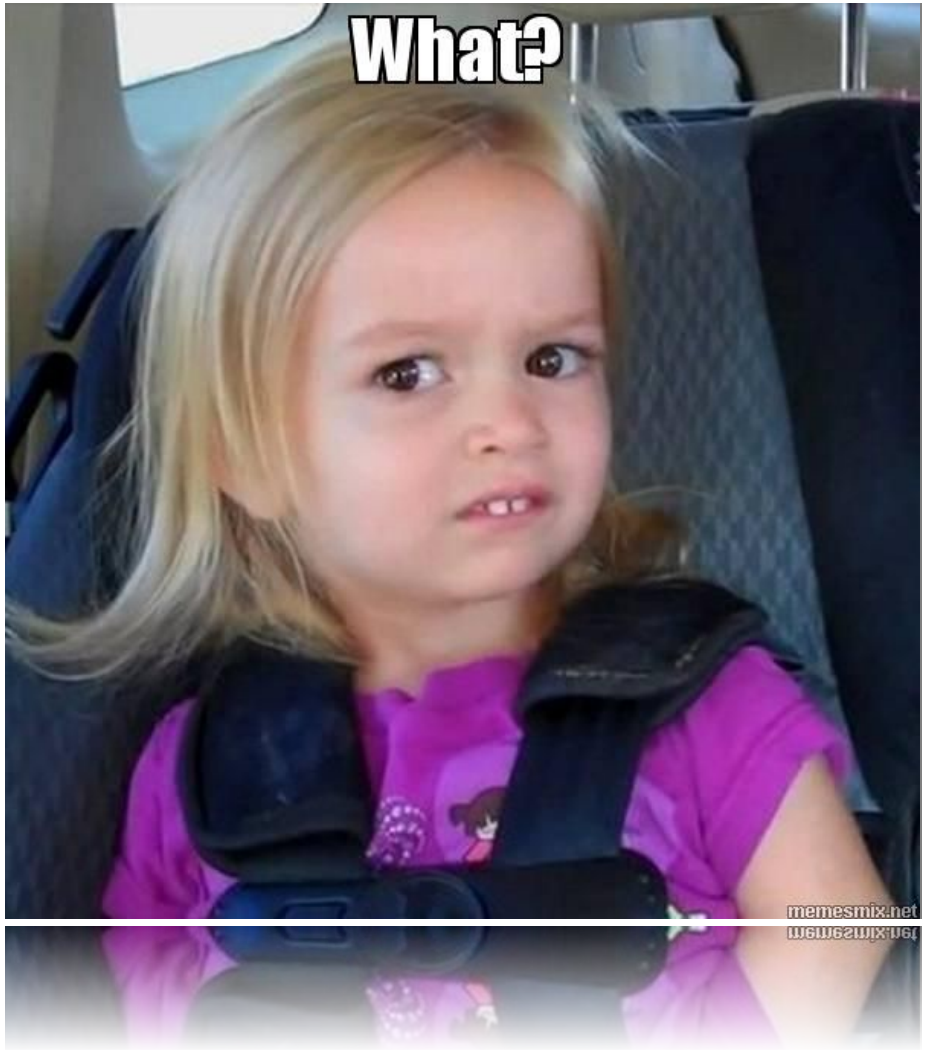
Lambdas, Functional Interfaces, Method references

- Functional Interfaces
  - @FunctionalInterface Annotation
  - Important Points
- Lambdas
  - Lambda Expressions
- Method References
- Practice


Java 8
Lambda Expressions
Method References
Default Methods
New Stream API
New Data/Time API
Nashron
InterviewBit

λ - calculus

- Симуляция машины Тьюринга
  - x (variable) - математический параметр.
  - (λx.M) (abstraction) - объявление функции (M лямбда терма x- входной параметр)
  - (M N) (application) - применение функции к аргументу


What?
memesmix.net

λ - calculus

Операция	как писать	математика	код
variable	λ.x = x	v = 1	int v = 1;

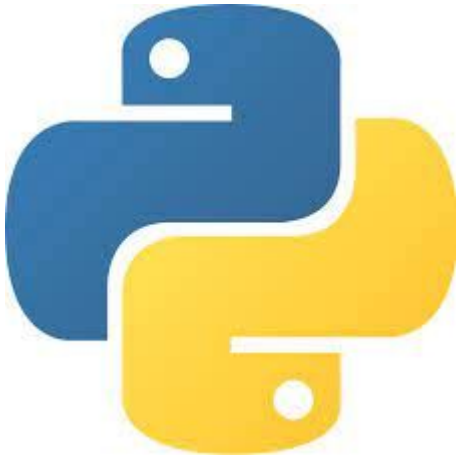
abstraction	λx.M	*f*(x) = x+1	f = Function {
			int apply(x) {
			return x+1;
			}
			}

application	(M N)	*f*(1)	f.apply(1);

λ - calculus

λ - calculus

λ function - анонимная функция (класс)



everything is function

Haskell

```
plus = λm. λn. λf. λx. m f (n f x)
```

```
plus = lam "m" $ lam "n" $ lam "f" $ lam "x" $ app (app (sym "m")(sym "f")) (app (app(sym "n")(sym "f"))(sym "x"))
```

λ - calculus

1. Immutable - неизменяемый
2. Purity - нет состояния

Functional Interfaces

- Содержит только один абстрактный метод
- Используется в лямбда выражениях и method reference
- Может иметь любое количество методов по умолчанию (default)
- Может иметь любое количество (static) методов
- Делает код более читабельным, чистым и простым
- Представляется путем аннотации @FunctionalInterface

Function in Java

- Интерфейс с SAM(Single Abstract Method)
- Лямбда - анонимный класс который реализует функциональный интерфейс
  - Example: (Integer someVal) -> someVal +1;

- Ссылка на метод
- Как Аргумент

Functional Interfaces

Аннотация @FunctionalInterface гарантирует, что интерфейс не может иметь более одного абстрактного метода

В Java есть встроенные функциональные интерфейсы, размещенные в пакете java.util.function

- Consumer
- Predicate
- Function
- Supplier

Functional Interfaces

Function<T,R>	1 входной и 1 выходной параметр	(val) -> val +1;
BiFunction<T,U,R>	2 входных и 1 выходной параметр	(first, second) -> first+second;
Consumer<T>	1 входной и 0 выходных	(v)->print(v)
Supplier<R>	0 входных и 1 выход	()->return 1;

Consumer

- Расширения Consumer — DoubleConsumer, IntConsumer и LongConsumer.

Supplier

- Ленивая генерация значений или для определения
- Расширения функционального интерфейса BooleanSupplier, DoubleSupplier, LongSupplier и IntSupplier.

Function

Unary Operator and Binary Operator - расширяют Function и Bi-Function

- унарный оператор принимает только один аргумент и возвращает только один аргумент
- бинарный оператор принимает два значения и возвращает одно значение, сравнимое с бифункцией

Predicate

Функция, которая принимает аргумент и, в свою очередь, генерирует логическое значение в качестве ответа, называется предикатом.

Function<T, Boolean>

Расширения - IntPredicate, DoublePredicate и LongPredicate.

Bi-Predicate является расширением функционального интерфейса.

Important

- Один абстрактный метод.
- @FunctionalInterface - необязательна.
- Можно добавить бесконечное количество методов (будь то статических или по умолчанию).
- Переопределение методов из родительского класса не нарушает правил функционального интерфейса в Java.
- Пакет java.util.function содержит множество встроенных функциональных интерфейсов в Java 8.

Lambdas

Лямбда - реализация функционального интерфейса - аля анонимный класс

```
(x) -> x+1;
int i -> i+2;
str -> str.length();
```

Method References

-> Чтобы обратиться к методу в объекте
Объект :: имя метода
-> Чтобы напечатать все элементы в списке
list.forEach(s -> System.out.println(s));
-> Сокращение для печати всех элементов в списке
list.forEach(System.out::println);