1. Why Lambda?

- Enables functional programming.

- Readable and concise code.

- Enables to write easy to use APIs and libraries.

- Enables support for parallel programming.

1. What is Lambda Expression?

It is like an anonymous function.

- without name

- without modifiers

- without return type

1. Difference between object-oriented programming and functional programming?

In oops a function/code blocks can’t exist on its own. It needs to be associated with a class. However, in functional programming function can exist without a class.

Public void greet(action){

action();

}

Lambda is out there to achieve this. We can pass the lambda function as value and execute the lambda.

1. What is Functional Interface?

An interface which contains single abstract method is called Function Interface. If you want to invoke Lambda expression, then function interface is required. Functional Interface job is to call the Lambda expression.

Eg : Runnable, Callable, Comparable, Comparator, ActionListener

1. Difference between Anonymous and Lambda?

Anonymous can be used for the interfaces which have more than 1 abstract methods. However, Lambda can be used only if the interface has only one abstract methods.

1. What are pre-defined functional interfaces in Java?

**Predicate**

interface Predicate<T>{

public boolean test(T t);

}

**Function**

interface Function<T,R>{

public R apply(T t);

}

**Consumer**

interface Consumer<T>{

public void accept(T t);

}

**Supplier**

interface Supplier<R>{

public R get();

}

**BiPredicate**

interface BiPredicate<T1,T2>{

public boolean test(T1 t1,T2 t2);

}

**BiFunction**

interface BiFunction<T,U,R>{

public R apply(T t,U u);

}

**BiConsumer**

interface BiConsumer<T,U>{

public void accept(T t, U u);

}