

# Agenda.

→ Functional Interfaces.

→ Some important functional Interfaces.

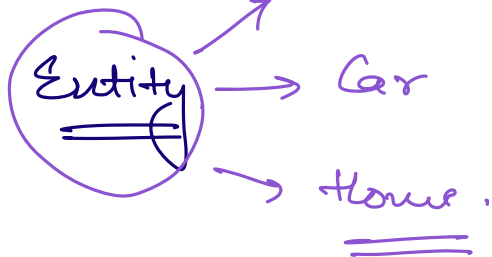
→ Lambda Expressions.

→ Streams.

↳ All the important functions of Streams API.

⇒ Class.

↳ Blueprint of an Entity



Entity is a central concept in a circle, with arrows pointing to Student, Car, and House.

⇒ Interface.

↳ Contract for behaviours.

↳ Blueprint of behaviours.

↓  
Methods.

Animal {

↳ Anything that can eat/walk & run is an Animal.

Interface Animal {

void eat();  $\Rightarrow$  Method without body  
Abstract Method.

void walk();

void run();

}

Class Tiger implements Animal {

eat() {

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

}

run() {

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

}

walk() {

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

}

}

# # Functional Interface.

→ Single Abstract Method.

→ But it can have any # of default method.

interface Runnable ← FI

void run(); } SAM.

Keyword ⇒ default void walk();

=====

3

3

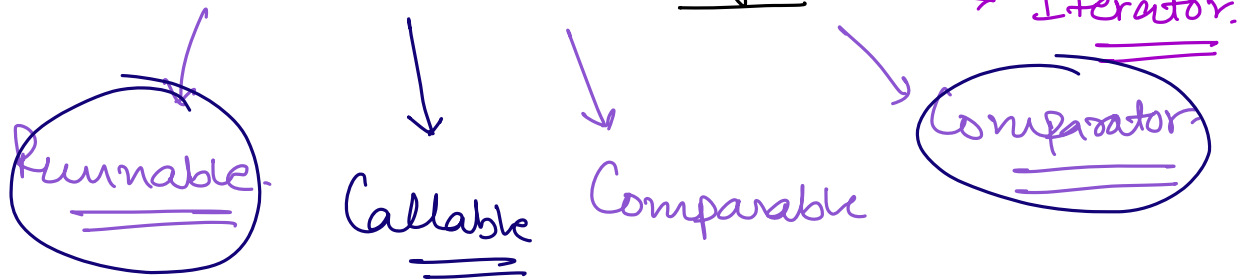
⇒ FI can also have a static keyword.

⇒ @FunctionalInterface.

↓

Optional.

## Famous Functional Interfaces.



⇒ 7 most important functional Interface.

→ Consumer	Predicate	Function
BiConsumer	BiPredicate	BiFunction

## Binary Operator.

Consumer  $\langle T \rangle$  & BiConsumer  $\langle T, U \rangle$



void accept( $T t$ )

void accept( $T t, U u$ )

Predicate  $\langle T \rangle$  (vs) BiPredicate  $\langle T, U \rangle$

boolean test( $T t$ )

boolean test( $T t, U u$ )

Function  $\langle T, R \rangle$  (vs)

Bifunction  $\langle T, U, V \rangle$

$R$  apply  $(T \ t)$

$V$  apply  $(T \ t, U \ u)$

Binary Operator  $\langle T \rangle$

$T$  apply  $(T \ t1, T \ t2)$