# Java 8

* Lambda expressions
  + Functional interface
    - Predicate<T> -> **boolean** *test*(T t)
    - BiPredicate<T, U> -> **boolean** *test*(T t, U u)
    - Supplier<T> -> **T** *get()*
    - Consumer<T> -> **void** *accept*(T t)
    - BiConsumer<T, U> -> **void** *accept*(T t, U u)
    - Function<T, R> -> **R** *apply*(T t)
    - BiFunction<T, U, R> -> **R** *apply*(T t, U u)
    - UnaryOperator<T> -> **T** *apply*(T t)
    - BinaryOperator<T > -> **T** *apply*(T t1, T t2)
  + Final and Effectively Final
    - Lambda takes a snapshot of local variables. Local vars that’s used in lambda MUST NOT change (effectively final), it gives compiler error, even if it’s changed before lambda definition.
  + Method References – turns into lambdas in the background
    - Bound – bounded to some instance -> don’t have to specify which instance to call it on
    - Unbound – need to specify which instance to call it on. First parameter is used for executing the instance method
    - Static (Unbound) – calling static method in lambda
    - Constructor MR – calling constructor in lambda -> in MR just Type::new