Here’s a summary of the **Java 8 Optional class** and its main methods:

**1. Purpose**

* Optional<T> is a container object that may or may not contain a non-null value.
* Helps avoid NullPointerException.

**2. Creation**

Optional<String> emptyOpt = Optional.empty();

Optional<String> valueOpt = Optional.of("Hello");         // Throws if null

Optional<String> nullableOpt = Optional.ofNullable(null); // Allows null

**3. Main Methods**

| **Method** | **Description** |
| --- | --- |
| isPresent() | Returns true if value is present |
| ifPresent(Consumer<T>) | Executes action if value is present |
| get() | Returns value if present, else throws exception |
| orElse(T other) | Returns value if present, else returns other |
| orElseGet(Supplier<T>) | Returns value if present, else gets from supplier |
| orElseThrow() | Returns value if present, else throws exception |
| map(Function<T,R>) | Maps value if present, returns Optional<R> |
| flatMap(Function<T,Optional<R>>) | Like map, but avoids nested Optional |
| filter(Predicate<T>) | Returns Optional if value matches predicate |

**4. Example Usage**

Optional<String> name = Optional.ofNullable(getName());

name.ifPresent(n -> System.out.println(n));

String result = name.orElse("Default");

Optional<Integer> length = name.map(String::length);

**Note:**  
Use Optional for return types, not for fields or method parameters. It encourages better null handling and functional programming style.