LAPORAN HASIL PRAKTIKUM

MATA KULIAH :

TEKNIK PEMROGRAMAN – PERTEMUAN 9

Disusun Oleh :

NIM	NAMA	KELAS
221524061	Thoriq Muhammad Fadhli	1B

D4 TEKNIK INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2023

DAFTAR ISI

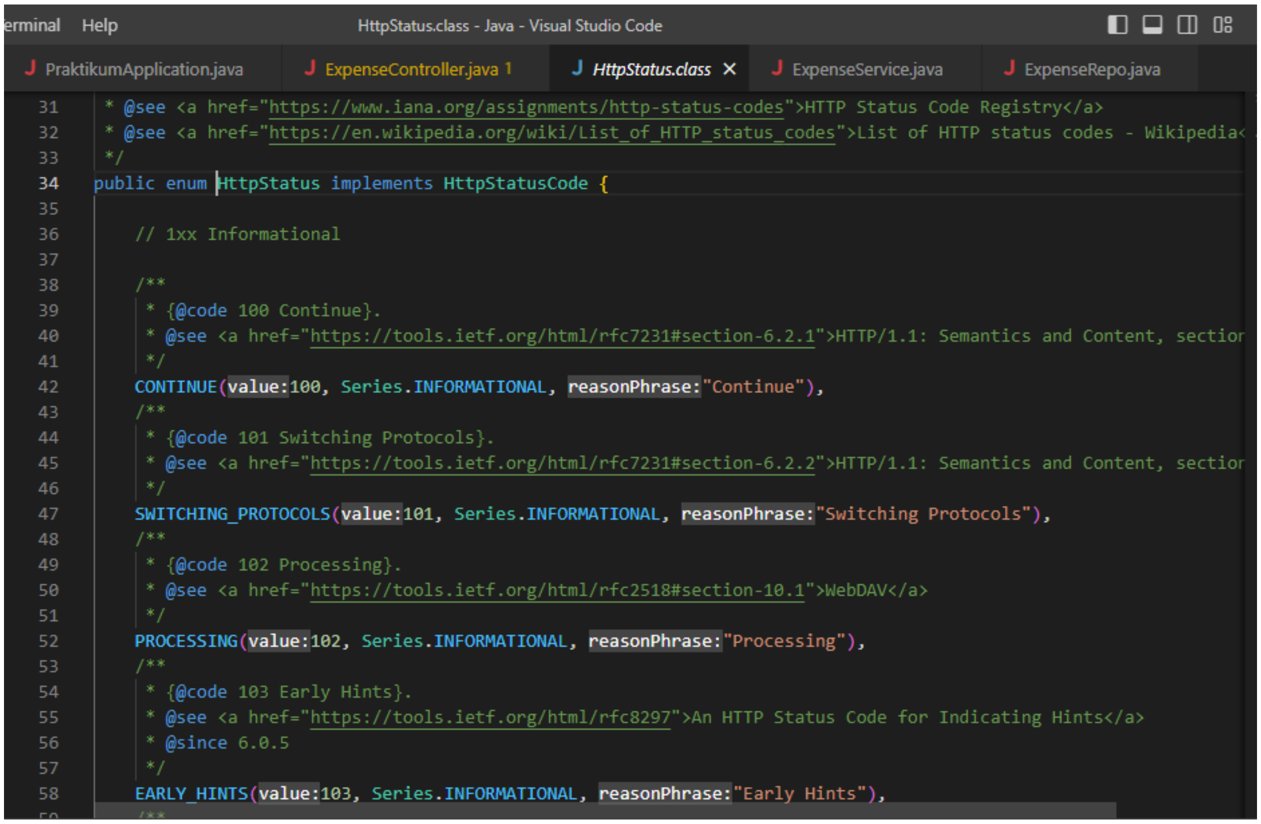
1. **Penerapan Generic Programming**

   A. ExpenseController.java

Enumeration

```
return ResponseEntity.status(HttpStatus.NO_CONTENT).build();
```

```
return ResponseEntity.status(HttpStatus.CREATED).build();
```



```
 * @see <a href="https://www.iana.org/assignments/http-status-codes">HTTP Status Code Registry</a>
 * @see <a href="https://en.wikipedia.org/wiki/List_of_HTTP_status_codes">List of HTTP status codes - Wikipedia<
 */
public enum HttpStatus implements HttpStatusCode {

    // 1xx Informational

    /**
     * {@code 100 Continue}.
     * @see <a href="https://tools.ietf.org/html/rfc7231#section-6.2.1">HTTP/1.1: Semantics and Content, sectior
     */
    CONTINUE(value:100, Series.INFORMATIONAL, reasonPhrase:"Continue"),
    /**
     * {@code 101 Switching Protocols}.
     * @see <a href="https://tools.ietf.org/html/rfc7231#section-6.2.2">HTTP/1.1: Semantics and Content, sectior
     */
    SWITCHING_PROTOCOLS(value:101, Series.INFORMATIONAL, reasonPhrase:"Switching Protocols"),
    /**
     * {@code 102 Processing}.
     * @see <a href="https://tools.ietf.org/html/rfc2518#section-10.1">WebDAV</a>
     */
    PROCESSING(value:102, Series.INFORMATIONAL, reasonPhrase:"Processing"),
    /**
     * {@code 103 Early Hints}.
     * @see <a href="https://tools.ietf.org/html/rfc8297">An HTTP Status Code for Indicating Hints</a>
     * @since 6.0.5
     */
    EARLY_HINTS(value:103, Series.INFORMATIONAL, reasonPhrase:"Early Hints"),
```

Generic Class

```
public ResponseEntity<Object> updateExpense(@RequestBody Expense expense)
```

```
return ResponseEntity.ok().build();
```

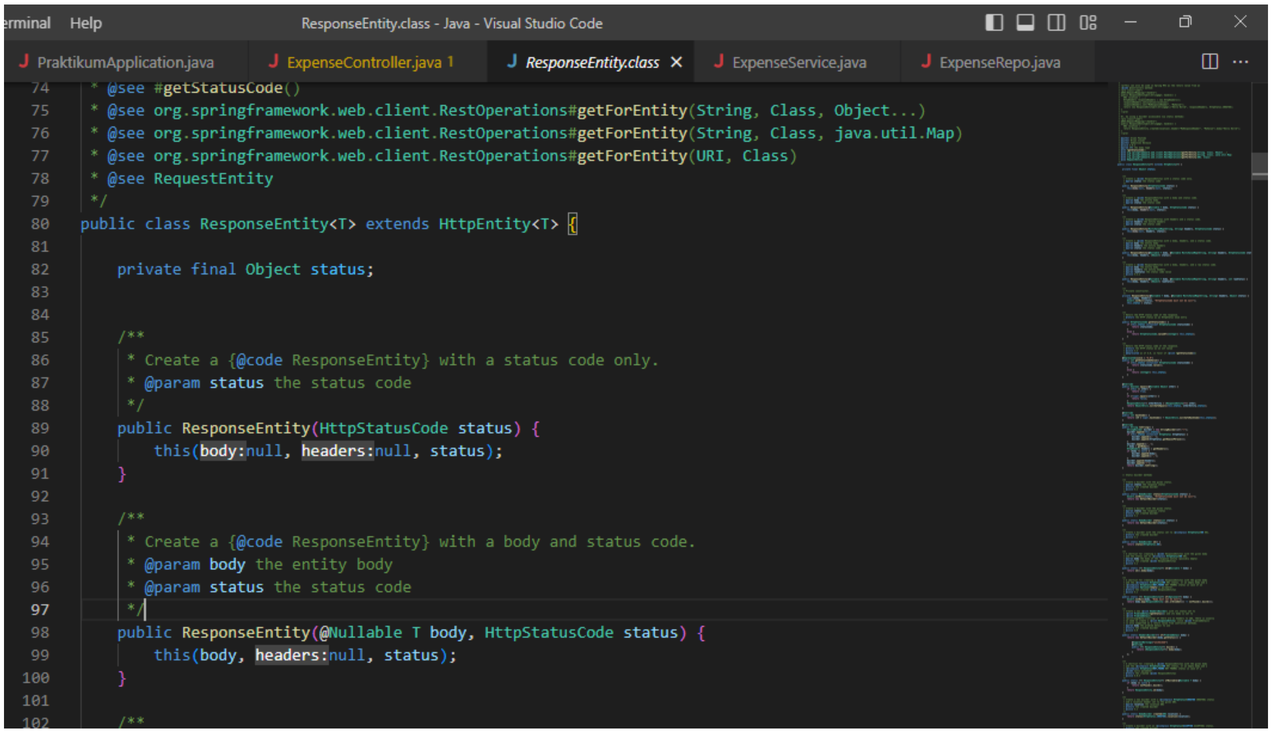
```
public ResponseEntity<Object> deleteExpense(@PathVariable String id)
```

```
public ResponseEntity<Expense> getExpenseByName(@PathVariable String name)
```

```
public ResponseEntity<List<Expense>> getAllExpense()
```

```java
return ResponseEntity.ok(expenseService.getExpenseByName(name));
```

```java
return ResponseEntity.ok(expenseService.getAllExpense());
```



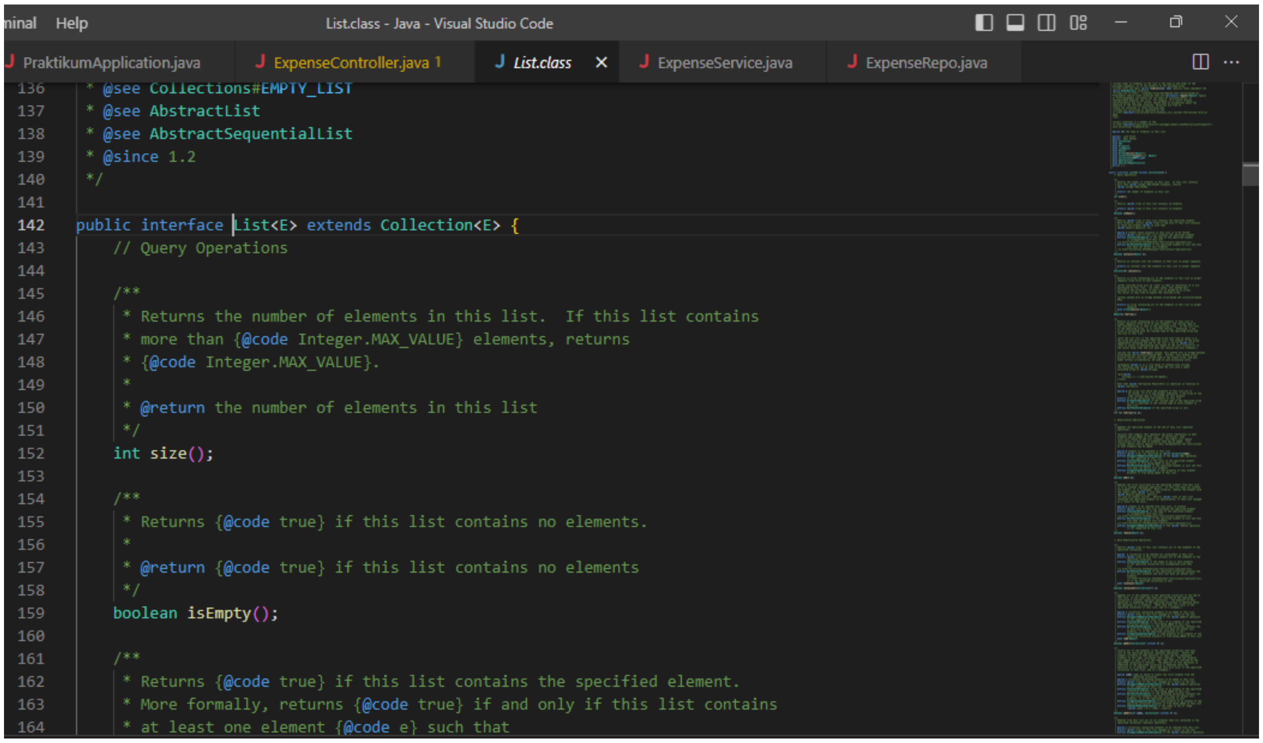
```java
 * @see #getStatusCode()
 * @see org.springframework.web.client.RestOperations#getForEntity(String, Class, Object...)
 * @see org.springframework.web.client.RestOperations#getForEntity(String, Class, java.util.Map)
 * @see org.springframework.web.client.RestOperations#getForEntity(URI, Class)
 * @see RequestEntity
 */
public class ResponseEntity<T> extends HttpEntity<T> {

    private final Object status;


    /**
     * Create a {@code ResponseEntity} with a status code only.
     * @param status the status code
     */
    public ResponseEntity(HttpStatusCode status) {
        this(body:null, headers:null, status);
    }

    /**
     * Create a {@code ResponseEntity} with a body and status code.
     * @param body the entity body
     * @param status the status code
     */
    public ResponseEntity(@Nullable T body, HttpStatusCode status) {
        this(body, headers:null, status);
    }

    /**
```



```java
 * @see Collections#EMPTY_LIST
 * @see AbstractList
 * @see AbstractSequentialList
 * @since 1.2
 */

public interface List<E> extends Collection<E> {
    // Query Operations

    /**
     * Returns the number of elements in this list.  If this list contains
     * more than {@code Integer.MAX_VALUE} elements, returns
     * {@code Integer.MAX_VALUE}.
     *
     * @return the number of elements in this list
     */
    int size();

    /**
     * Returns {@code true} if this list contains no elements.
     *
     * @return {@code true} if this list contains no elements
     */
    boolean isEmpty();

    /**
     * Returns {@code true} if this list contains the specified element.
     * More formally, returns {@code true} if and only if this list contains
     * at least one element {@code e} such that
```

B. ExpenseCategory

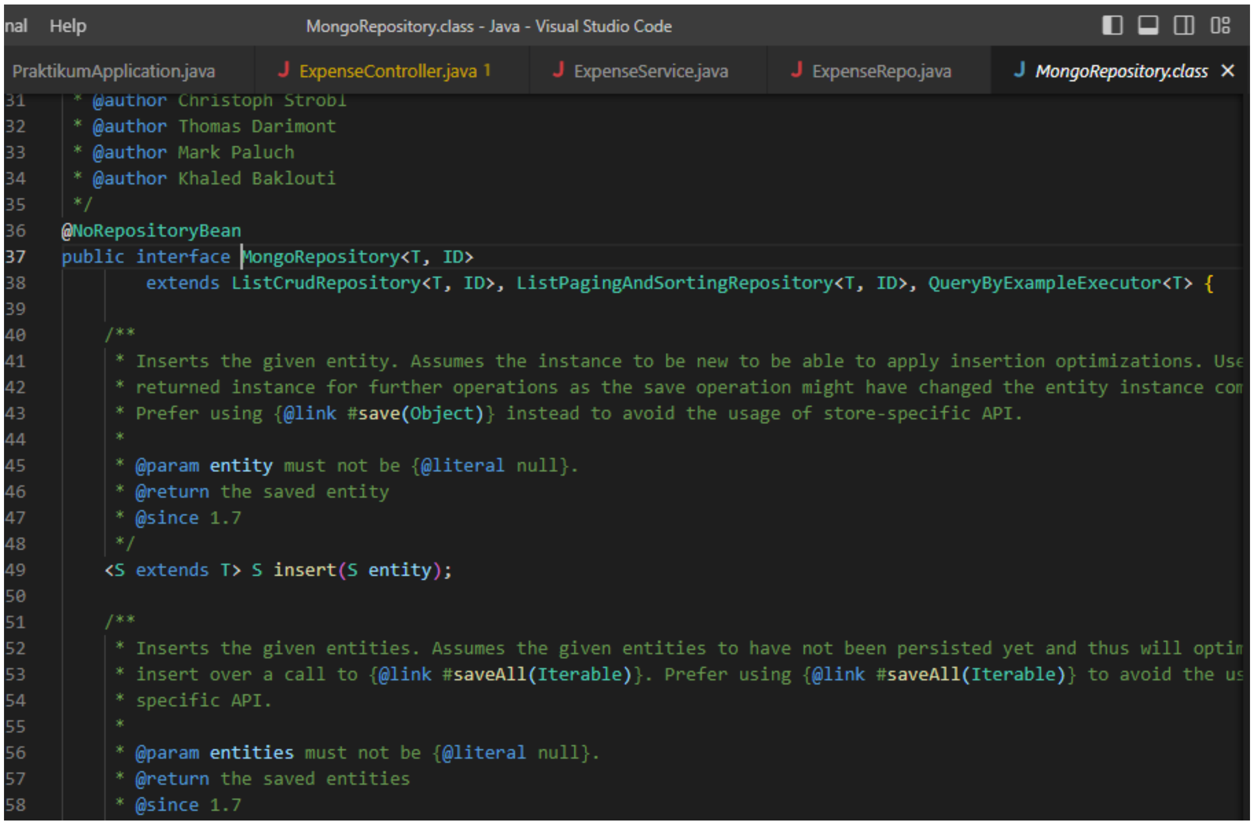
Enumeration

```
public enum ExpenseCategory {
    ENTERTAINMENT, GROCERIES, RESTAURANT, UTILITIES, MISC
}
```

C. ExpenseRepo.java

Generic Class

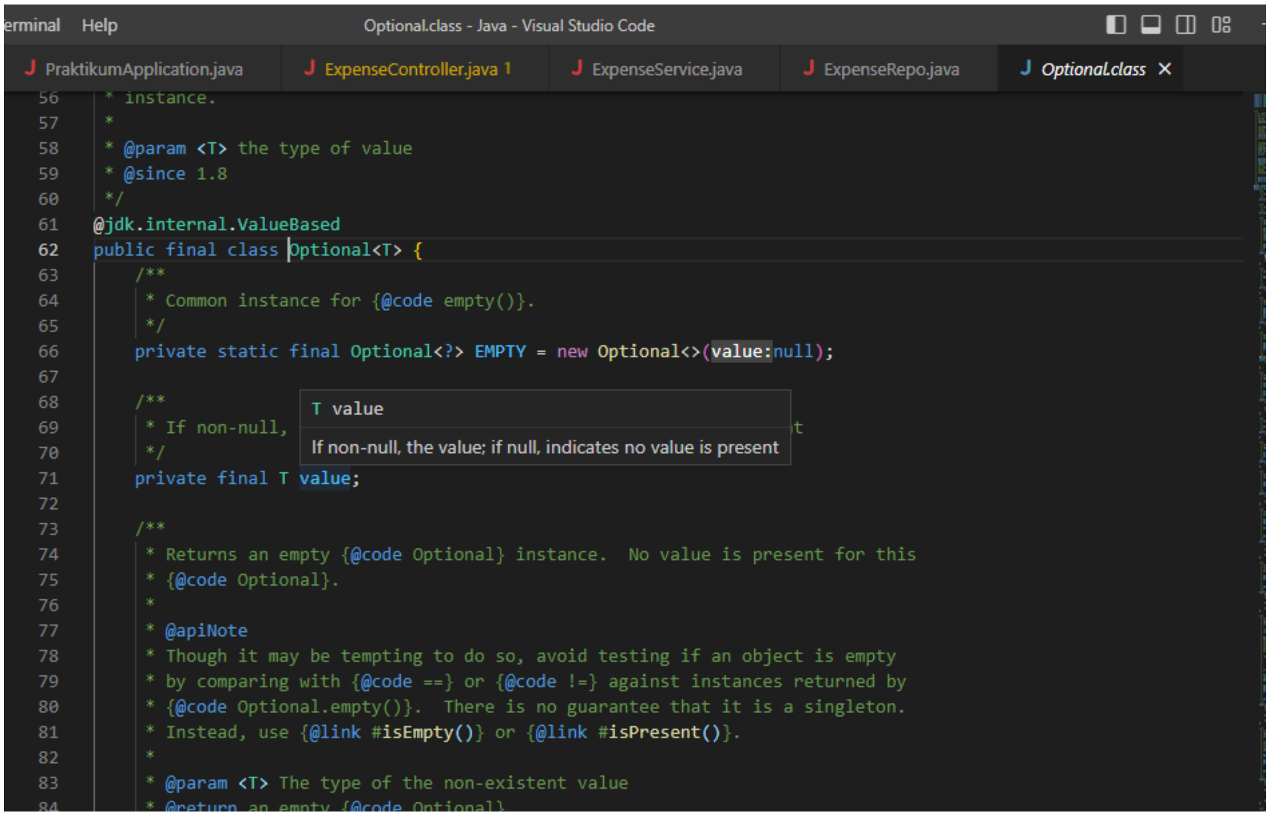
```
public interface ExpenseRepo extends MongoRepository<Expense, String> {
    @Query("{'name': ?0}")
    Optional<Expense> findByName(String name);
}
```



```
 * @author Christoph Strobl
 * @author Thomas Darimont
 * @author Mark Paluch
 * @author Khaled Baklouti
 */
@NoRepositoryBean
public interface MongoRepository<T, ID>
        extends ListCrudRepository<T, ID>, ListPagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {

    /**
     * Inserts the given entity. Assumes the instance to be new to be able to apply insertion optimizations. Use
     * returned instance for further operations as the save operation might have changed the entity instance com
     * Prefer using {@link #save(Object)} instead to avoid the usage of store-specific API.
     *
     * @param entity must not be {@literal null}.
     * @return the saved entity
     * @since 1.7
     */
    <S extends T> S insert(S entity);

    /**
     * Inserts the given entities. Assumes the given entities to have not been persisted yet and thus will optim
     * insert over a call to {@link #saveAll(Iterable)}. Prefer using {@link #saveAll(Iterable)} to avoid the us
     * specific API.
     *
     * @param entities must not be {@literal null}.
     * @return the saved entities
     * @since 1.7
```



```java
 * instance.
 *
 * @param <T> the type of value
 * @since 1.8
 */
@jdk.internal.ValueBased
public final class Optional<T> {
    /**
     * Common instance for {@code empty()}.
     */
    private static final Optional<?> EMPTY = new Optional<>(value:null);

    /**
     * If non-null,                                                  t
     */
    private final T value;

    /**
     * Returns an empty {@code Optional} instance.  No value is present for this
     * {@code Optional}.
     *
     * @apiNote
     * Though it may be tempting to do so, avoid testing if an object is empty
     * by comparing with {@code ==} or {@code !=} against instances returned by
     * {@code Optional.empty()}.  There is no guarantee that it is a singleton.
     * Instead, use {@link #isEmpty()} or {@link #isPresent()}.
     *
     * @param <T> The type of the non-existent value
     * @return an empty {@code Optional}
```

```
T value
If non-null, the value; if null, indicates no value is present
```

D. ExpenseService.java

Generic Class

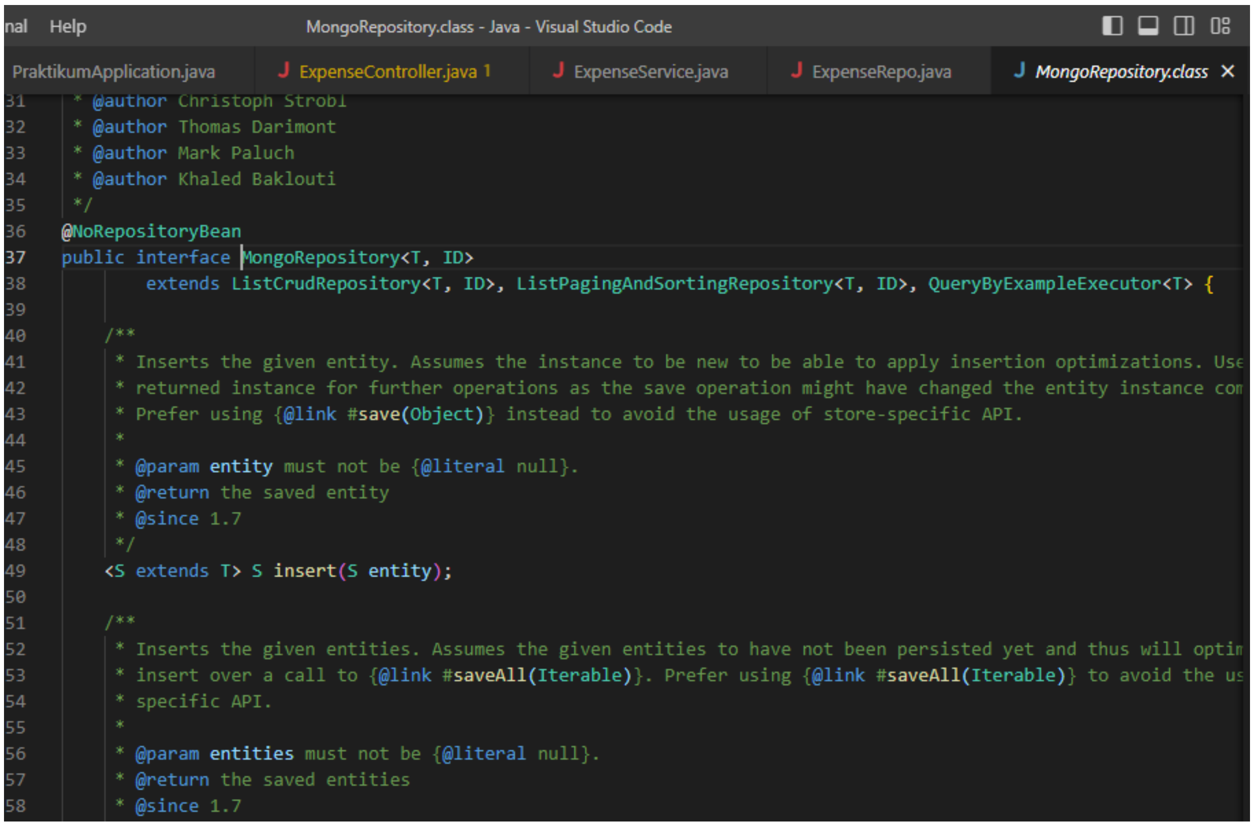
```java
expenseRepository.insert(expense);
```

```java
expenseRepository.findById(expense.getId())
```

```java
public List<Expense> getAllExpense() {
        return expenseRepository.findAll();
    }
```

```java
return expenseRepository.findByName(name)
```

```java
expenseRepository.deleteById(id);
```



```java
 * @author Christoph Strobl
 * @author Thomas Darimont
 * @author Mark Paluch
 * @author Khaled Baklouti
 */
@NoRepositoryBean
public interface MongoRepository<T, ID>
        extends ListCrudRepository<T, ID>, ListPagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {

    /**
     * Inserts the given entity. Assumes the instance to be new to be able to apply insertion optimizations. Use
     * returned instance for further operations as the save operation might have changed the entity instance com
     * Prefer using {@link #save(Object)} instead to avoid the usage of store-specific API.
     *
     * @param entity must not be {@literal null}.
     * @return the saved entity
     * @since 1.7
     */
    <S extends T> S insert(S entity);

    /**
     * Inserts the given entities. Assumes the given entities to have not been persisted yet and thus will optin
     * insert over a call to {@link #saveAll(Iterable)}. Prefer using {@link #saveAll(Iterable)} to avoid the us
     * specific API.
     *
     * @param entities must not be {@literal null}.
     * @return the saved entities
     * @since 1.7
```

2. **Solusi dari masalah yang dihadapi**

   Searching, Googling, Memahami lebih cermat.

3. **Nama teman yang membantu**

   1. Najib Alimudin Fajri