서비스 로직 작성

응답 메시지 정규화

공통 응답 Dto 설계

public으로 컨트롤러에서 호출할 시,
통일성 문제 발생

ex. code와 message의 내용이
개발자마다 달라질 수 있음

=> 따라서, 응답 구조 통일화

응답 생성자 3가지
(생성 성공, 일반 성공, 오류)

```java
@ToString
@NoArgsConstructor(access = AccessLevel.PRIVATE)
@Getter
public class ApiResponseDto<T> {
    private String code;
    private String message;
    private T data;

    private ApiResponseDto(String code, String message) {
        this.code = code;
        this.message = message;
    }

    private ApiResponseDto(String code, String message, T data) {
        this.code = code;
        this.message = message;
        this.data = data;
    }

    public static <T> ApiResponseDto<T> createOk(T data) {
        return new ApiResponseDto<>("OK", "요청이  성공하였습니다.", data);
    }

    public static ApiResponseDto<String> defaultOk() {
        return ApiResponseDto.createOk(null);
    }

    public static ApiResponseDto<String> createError(String code, String message) {
        return new ApiResponseDto<>(code, message);
    }
}
```

Api 응답에 공통 응답 Dto 적용

```
@GetMapping(value = "/test")
public ApiResponseDto<String> test() {
    String response = remoteAlimService.sms();
    return ApiResponseDto.createOk(response);
}

@PostMapping(value = "/sms")
public ApiResponseDto<SendSmsDto.Response> sms(@RequestBody SendSmsDto.Request request) {
    var response = remoteAlimService.sendSms(request);
    return ApiResponseDto.createOk(response);
}
```

(String 클래스를 데이터로 반환)

(alim API 요청 시, 반환된 객체를 데이터로 반환)

AOP Advice

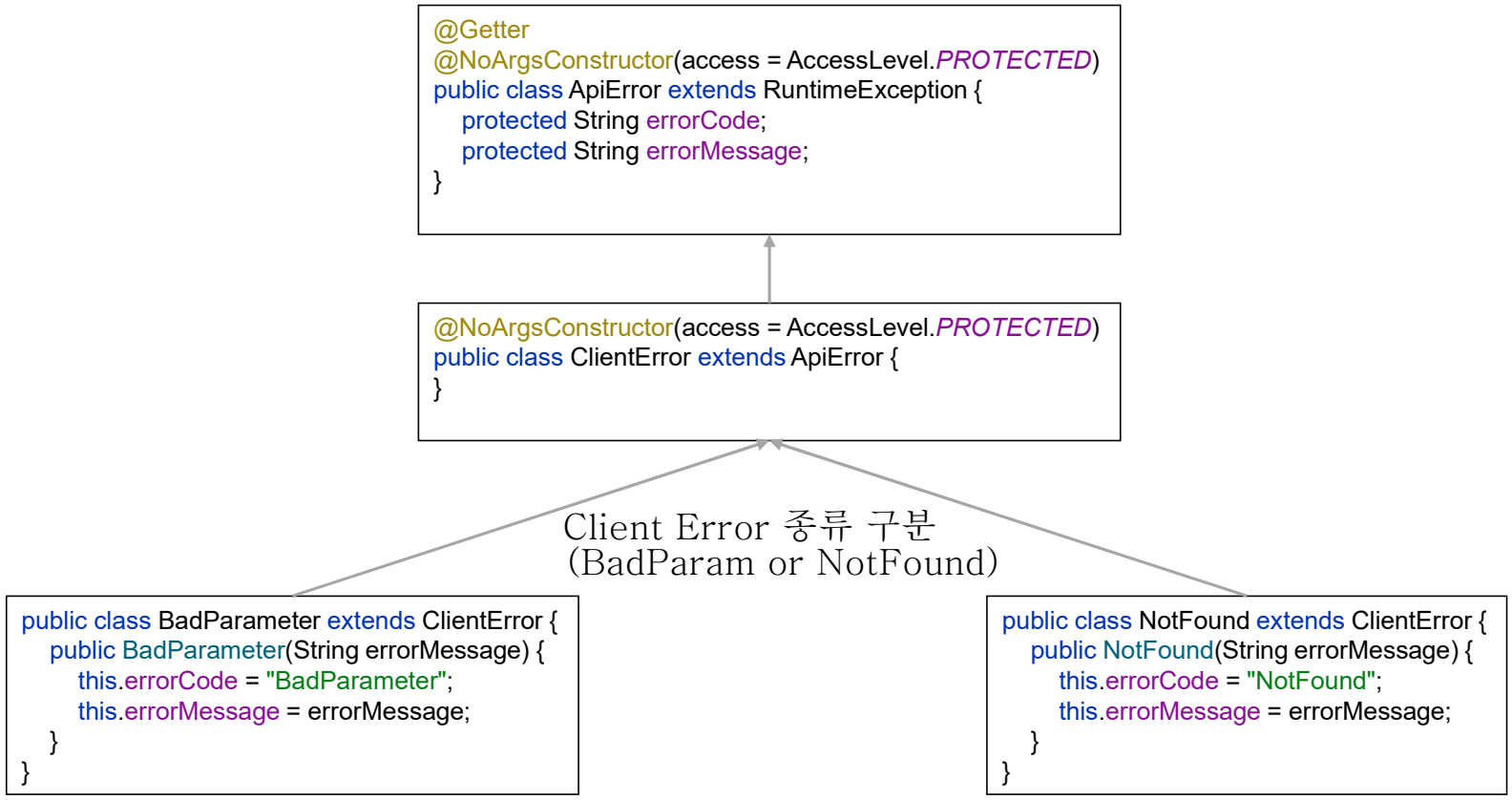
공통 에러 처리

```
@Slf4j
@Order(value = 1)
@RestControllerAdvice
public class ApiCommonAdvice {
    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
    @ExceptionHandler({Exception.class})
    public ApiResponseDto<String> handleException(Exception e) {
        return ApiResponseDto.createError(
                "ServerError",
                "서버  에러입니다."
        );
    }
}
```

모든 에러에 대해
500번 에러 처리

공통 형식인 ApiResponseDto으로 반환

Api Exception 설계



enum을 통해 Client Error 종류를 구분하는 방법도 존재

AOP Advice에 Api Exception 적용

```
@Slf4j
@Order(value = 1)
@RestControllerAdvice
public class ApiCommonAdvice {
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    @ExceptionHandler({ClientError.class})
    public ApiResponseDto<String> handleClientError(ClientError e) {
        return ApiResponseDto.createError(
                e.getErrorCode(),
                e.getErrorMessage()
        );
    }

    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
    @ExceptionHandler({Exception.class})
    public ApiResponseDto<String> handleException(Exception e) {
        return ApiResponseDto.createError(
                "ServerError",
                "서버  에러입니다."
        );
    }
}
```

(BadParam 처리)

정의된 에러 외
남은 모든 에러 처리