@ResponseBody VS ResponseEntity

* if you put @ResponseBody annotation in the method level, Spring will convert the return object in to the http response body.
* ResponseEntity works similar as @ResponseBody annotation. But when you create ResponseEntity object, you can add the response header to the http response as well.
* If you use ResponseEntity as return object, then you don’t need to use @ResponseBody annotation.
* @RequestBody VS HttpEntity
* if you put @RequestBody annotation aside the parameter in a method, Spring will convert the http request body to that declare class type in the method signature.
* HttpEntity works smilar as @RequestBody annotation. Plus you can access the http header from HttpEntity object, which @RequestBody annotation doesn’t offer.
* If you use HttpEntity as parameter object, then you don’t need to use @RequestBody annotation.

@ControllerAdvice

public class GlobalExceptionHandler{

@ExceptionHandler(value = ArithmeticException.class)

public ResponseEntity<ApiError> handleArithmeticException(ArithmeticException e)

{

ApiError error = new ApiError();

error.setException("Arithmetic exception: " + e.getMessage());

HttpStatus status = HttpStatus.BAD\_REQUEST;

return new ResponseEntity<ApiError>(error, status);

}

@ExceptionHandler(value = NullPointerException.class)

public ResponseEntity<ApiError> handleNullPointerException(NullPointerException e)

{

ApiError error = new ApiError();

error.setException("Null pointer exception: " + e.getMessage());

HttpStatus status = HttpStatus.NOT\_ACCEPTABLE;

return new ResponseEntity<ApiError>(error, status);

}

@ExceptionHandler(value = Exception.class)

public ResponseEntity<ApiError> handleNullPointerException(Exception e)

{

ApiError error = new ApiError();

error.setException("Exception: " + e.getMessage());

HttpStatus status = HttpStatus.INTERNAL\_SERVER\_ERROR;

return new ResponseEntity<ApiError>(error, status);

}

}

Now, @ControllerAdvice is intercepting all the exceptions occurring in the