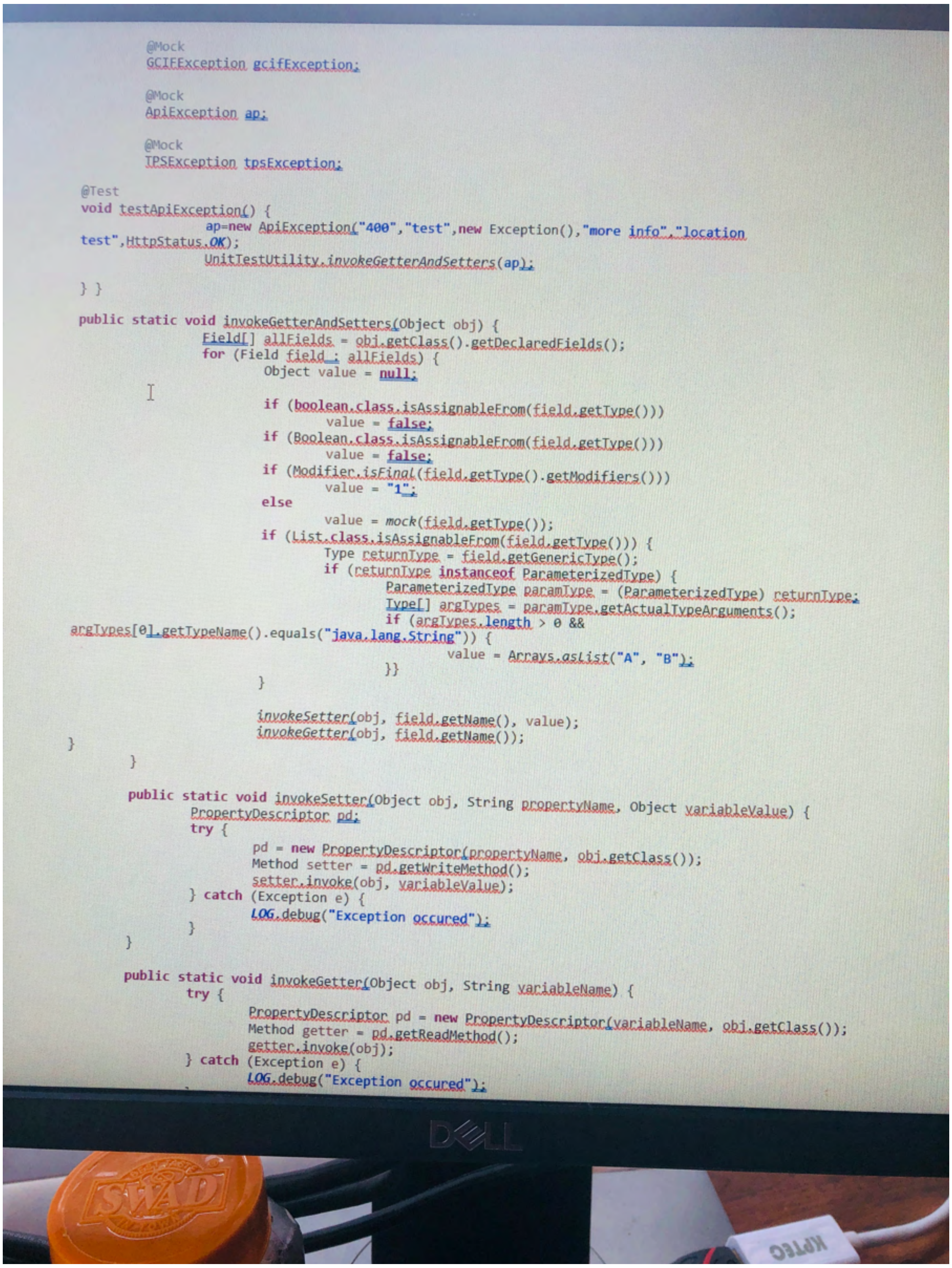
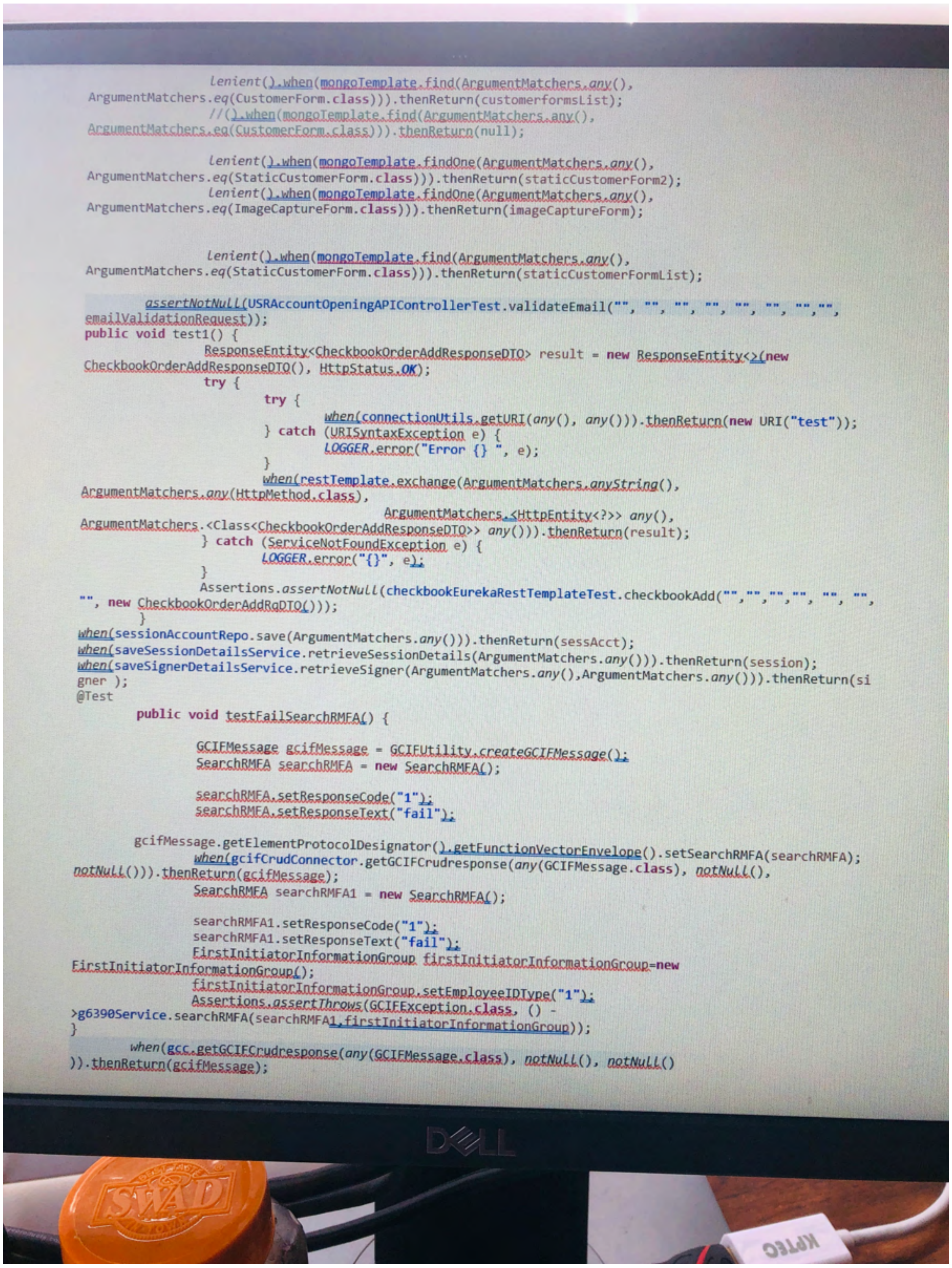
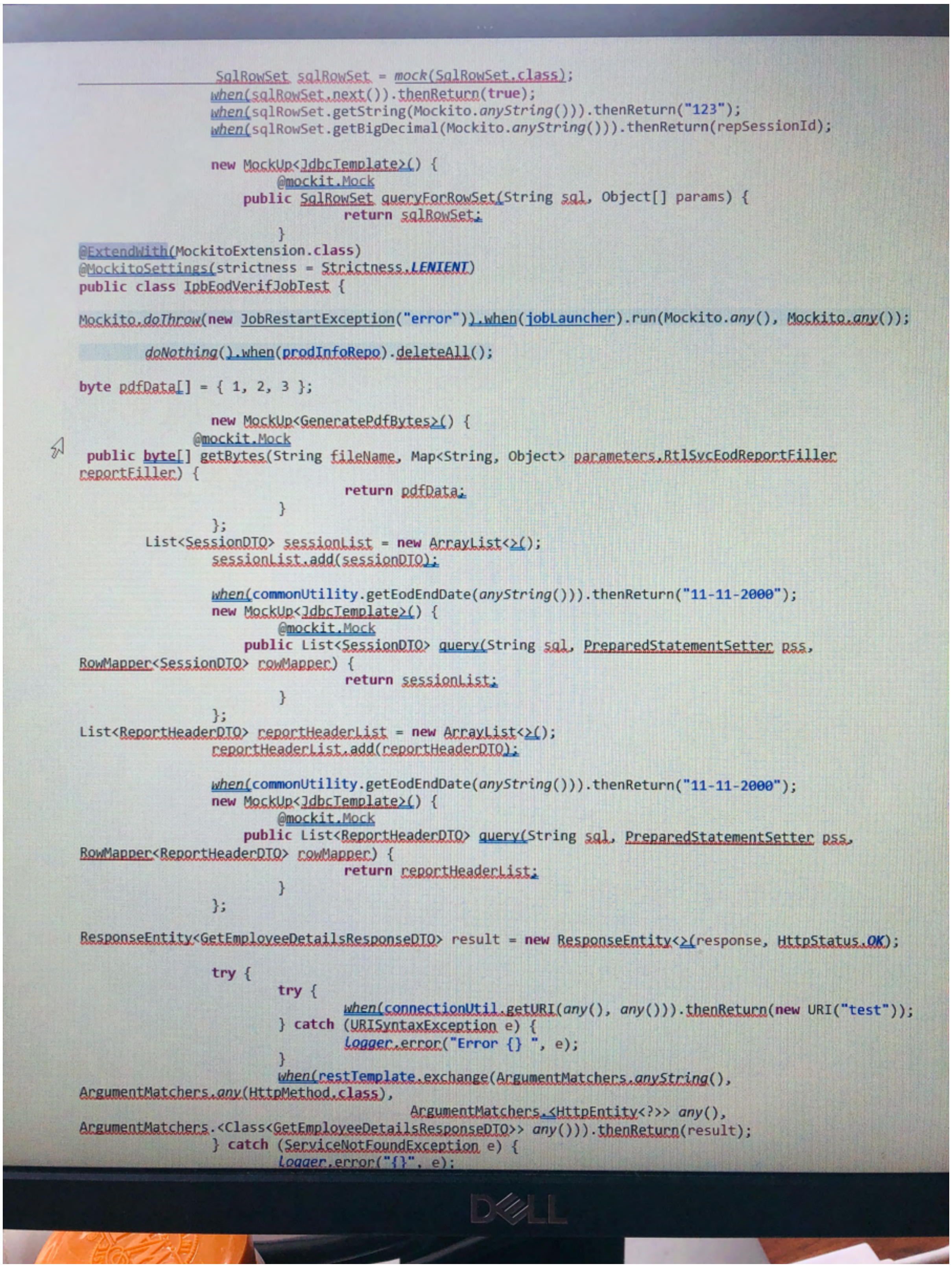


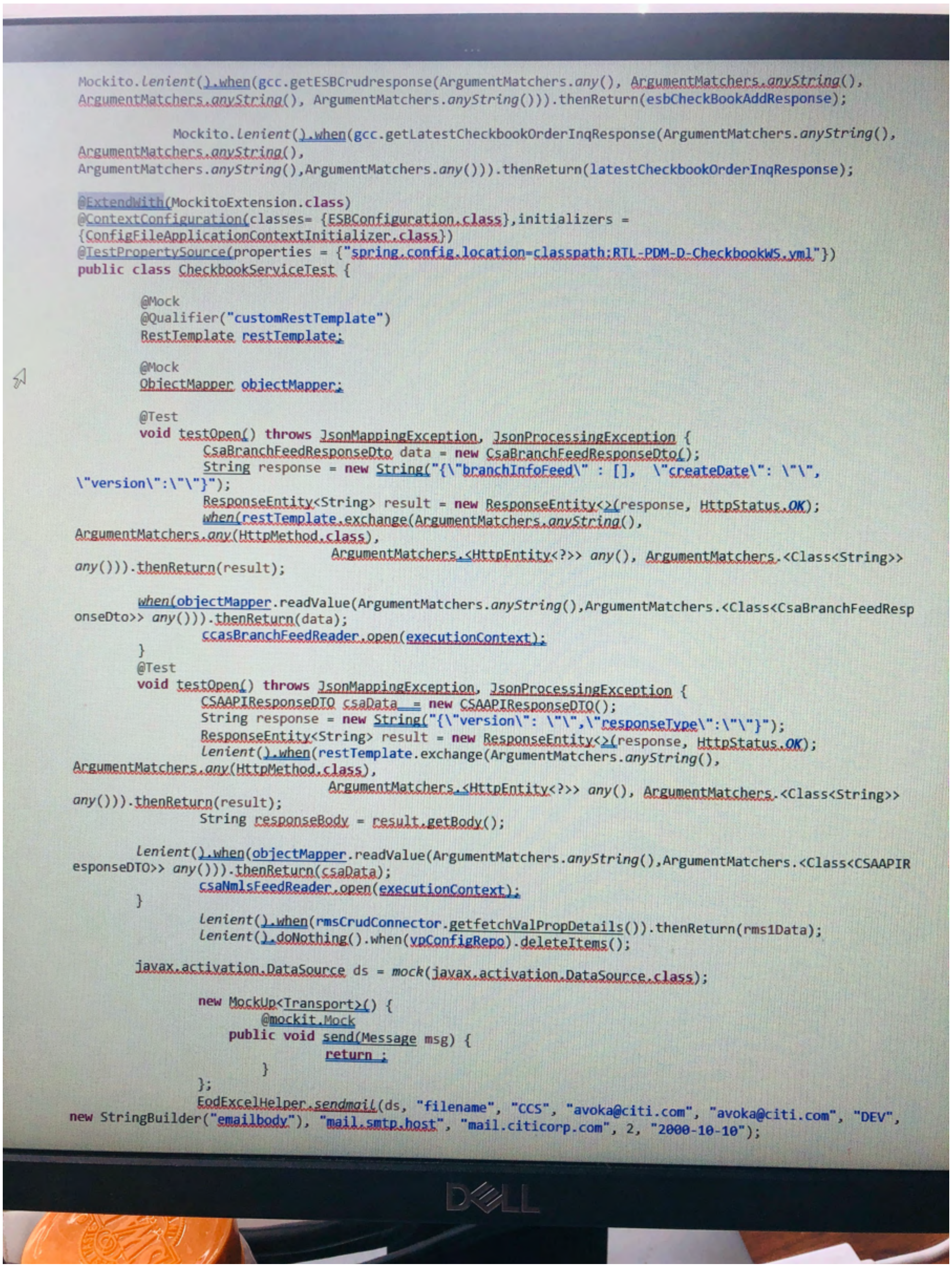
```
        @Mock
        GCIFException gcifException;

        @Mock
        ApiException ap;

        @Mock
        TPSException tpsException;

    @Test
    void testApiException() {
                ap=new ApiException("400","test",new Exception(),"more info","location
    test",HttpStatus.OK);
                UnitTestUtility.invokeGetterAndSetters(ap);

    } }

    public static void invokeGetterAndSetters(Object obj) {
                Field[] allFields = obj.getClass().getDeclaredFields();
                for (Field field : allFields) {
                        Object value = null;

                        if (boolean.class.isAssignableFrom(field.getType()))
                                value = false;
                        if (Boolean.class.isAssignableFrom(field.getType()))
                                value = false;
                        if (Modifier.isFinal(field.getType().getModifiers()))
                                value = "1";
                        else
                                value = mock(field.getType());
                        if (List.class.isAssignableFrom(field.getType())) {
                                Type returnType = field.getGenericType();
                                if (returnType instanceof ParameterizedType) {
                                        ParameterizedType paramType = (ParameterizedType) returnType;
                                        Type[] argTypes = paramType.getActualTypeArguments();
                                        if (argTypes.length > 0 &&
    argTypes[0].getTypeName().equals("java.lang.String")) {
                                                value = Arrays.asList("A", "B");
                                        }}
                        }

                        invokeSetter(obj, field.getName(), value);
                        invokeGetter(obj, field.getName());
    }
        }

        public static void invokeSetter(Object obj, String propertyName, Object variableValue) {
                PropertyDescriptor pd;
                try {
                        pd = new PropertyDescriptor(propertyName, obj.getClass());
                        Method setter = pd.getWriteMethod();
                        setter.invoke(obj, variableValue);
                } catch (Exception e) {
                        LOG.debug("Exception occured");
                }
        }

        public static void invokeGetter(Object obj, String variableName) {
                try {
                        PropertyDescriptor pd = new PropertyDescriptor(variableName, obj.getClass());
                        Method getter = pd.getReadMethod();
                        getter.invoke(obj);
                } catch (Exception e) {
                        LOG.debug("Exception occured");
```



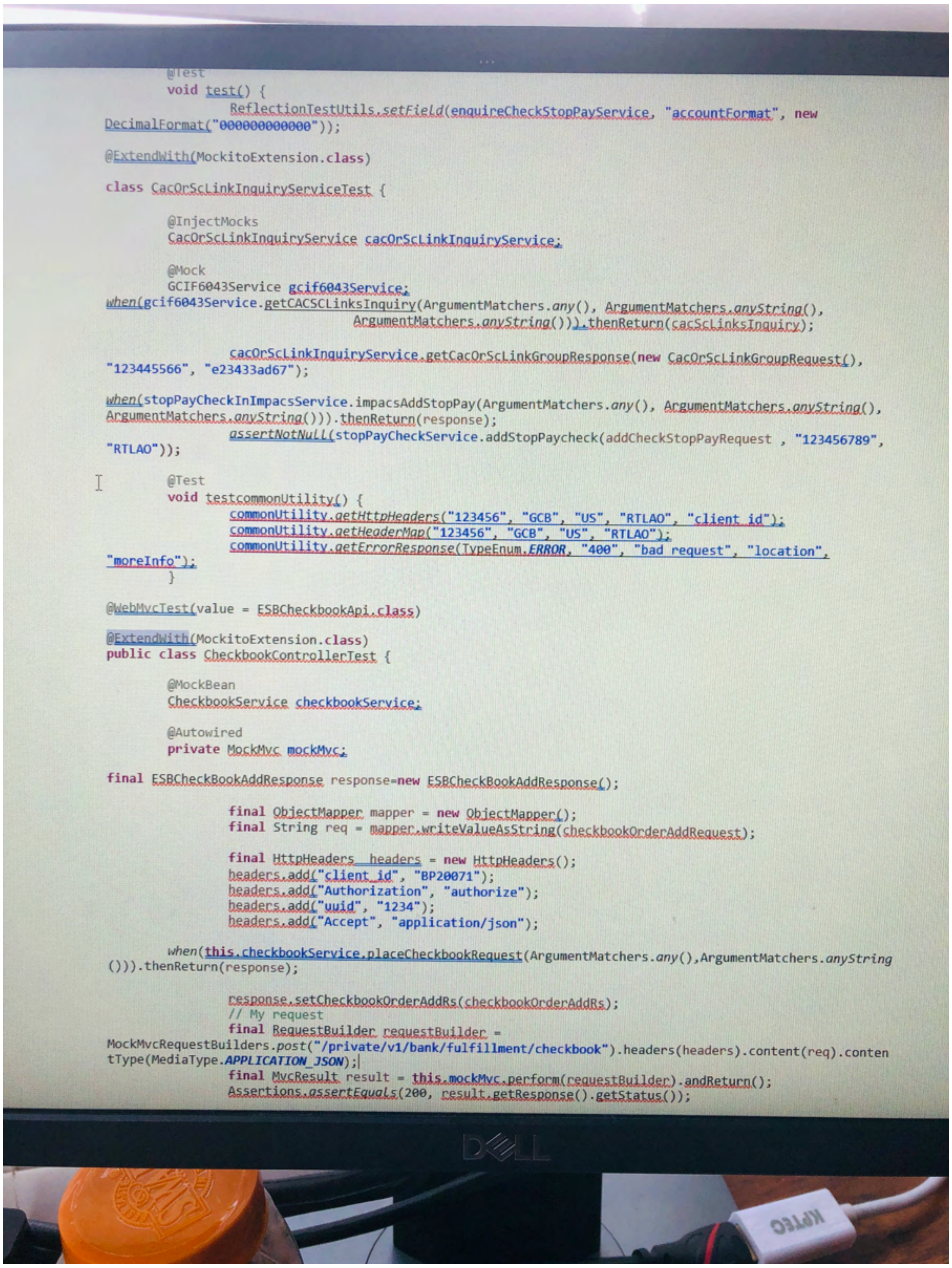
```
                    lenient().when(mongoTemplate.find(ArgumentMatchers.any(),
ArgumentMatchers.eq(CustomerForm.class))).thenReturn(customerformsList);
                    //().when(mongoTemplate.find(ArgumentMatchers.any(),
ArgumentMatchers.eq(CustomerForm.class))).thenReturn(null);

                    lenient().when(mongoTemplate.findOne(ArgumentMatchers.any(),
ArgumentMatchers.eq(StaticCustomerForm.class))).thenReturn(staticCustomerForm2);
                    lenient().when(mongoTemplate.findOne(ArgumentMatchers.any(),
ArgumentMatchers.eq(ImageCaptureForm.class))).thenReturn(imageCaptureForm);


                    lenient().when(mongoTemplate.find(ArgumentMatchers.any(),
ArgumentMatchers.eq(StaticCustomerForm.class))).thenReturn(staticCustomerFormList);

          assertNotNull(USRAccountOpeningAPIControllerTest.validateEmail("", "", "", "", "", "", "","",
emailValidationRequest));
public void test1() {
                    ResponseEntity<CheckbookOrderAddResponseDTO> result = new ResponseEntity<>(new
CheckbookOrderAddResponseDTO(), HttpStatus.OK);
                    try {
                              try {
                                        when(connectionUtils.getURI(any(), any())).thenReturn(new URI("test"));
                              } catch (URISyntaxException e) {
                                        LOGGER.error("Error {} ", e);
                              }
                              when(restTemplate.exchange(ArgumentMatchers.anyString(),
ArgumentMatchers.any(HttpMethod.class),
                                                  ArgumentMatchers.<HttpEntity<?>> any(),
ArgumentMatchers.<Class<CheckbookOrderAddResponseDTO>> any())).thenReturn(result);
                    } catch (ServiceNotFoundException e) {
                              LOGGER.error("{}", e);
                    }
                    Assertions.assertNotNull(checkbookEurekaRestTemplateTest.checkbookAdd("","","","", "", "",
"", new CheckbookOrderAddRqDTO()));
          }
when(sessionAccountRepo.save(ArgumentMatchers.any())).thenReturn(sessAcct);
when(saveSessionDetailsService.retrieveSessionDetails(ArgumentMatchers.any())).thenReturn(session);
when(saveSignerDetailsService.retrieveSigner(ArgumentMatchers.any(),ArgumentMatchers.any())).thenReturn(si
gner );
@Test
          public void testFailSearchRMFA() {

                    GCIFMessage gcifMessage = GCIFUtility.createGCIFMessage();
                    SearchRMFA searchRMFA = new SearchRMFA();

                    searchRMFA.setResponseCode("1");
                    searchRMFA.setResponseText("fail");

          gcifMessage.getElementProtocolDesignator().getFunctionVectorEnvelope().setSearchRMFA(searchRMFA);
                    when(gcifCrudConnector.getGCIFCrudresponse(any(GCIFMessage.class), notNull(),
notNull())).thenReturn(gcifMessage);
                    SearchRMFA searchRMFA1 = new SearchRMFA();

                    searchRMFA1.setResponseCode("1");
                    searchRMFA1.setResponseText("fail");
                    FirstInitiatorInformationGroup firstInitiatorInformationGroup=new
FirstInitiatorInformationGroup();
                    firstInitiatorInformationGroup.setEmployeeIDType("1");
                    Assertions.assertThrows(GCIFException.class, () -
>g6390Service.searchRMFA(searchRMFA1,firstInitiatorInformationGroup));
}
          when(gcc.getGCIFCrudresponse(any(GCIFMessage.class), notNull(), notNull()
)).thenReturn(gcifMessage);
```



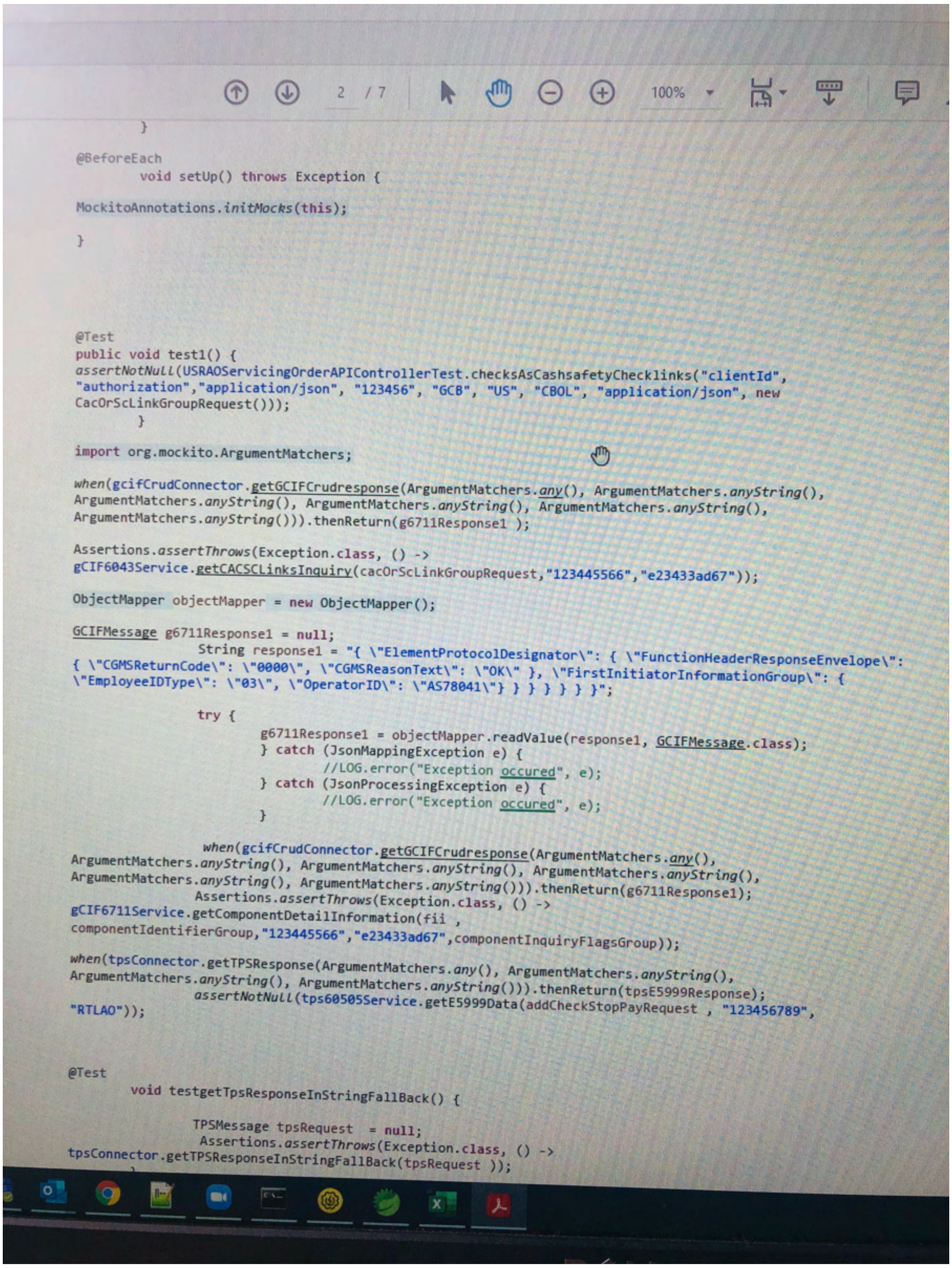
```
                 SqlRowSet sqlRowSet = mock(SqlRowSet.class);
                when(sqlRowSet.next()).thenReturn(true);
                when(sqlRowSet.getString(Mockito.anyString())).thenReturn("123");
                when(sqlRowSet.getBigDecimal(Mockito.anyString())).thenReturn(repSessionId);

                new MockUp<JdbcTemplate>() {
                        @mockit.Mock
                    public SqlRowSet queryForRowSet(String sql, Object[] params) {
                                return sqlRowSet;
                        }
@ExtendWith(MockitoExtension.class)
@MockitoSettings(strictness = Strictness.LENIENT)
public class IpbEodVerifJobTest {

Mockito.doThrow(new JobRestartException("error")).when(jobLauncher).run(Mockito.any(), Mockito.any());

        doNothing().when(prodInfoRepo).deleteAll();

byte pdfData[] = { 1, 2, 3 };

                new MockUp<GeneratePdfBytes>() {
              @mockit.Mock
 public byte[] getBytes(String fileName, Map<String, Object> parameters,RtlSvcEodReportFiller
reportFiller) {
                                return pdfData;
                        }
                };
        List<SessionDTO> sessionList = new ArrayList<>();
                sessionList.add(sessionDTO);

                when(commonUtility.getEodEndDate(anyString())).thenReturn("11-11-2000");
                new MockUp<JdbcTemplate>() {
                        @mockit.Mock
                    public List<SessionDTO> query(String sql, PreparedStatementSetter pss,
RowMapper<SessionDTO> rowMapper) {
                                return sessionList;
                        }
                };
List<ReportHeaderDTO> reportHeaderList = new ArrayList<>();
                reportHeaderList.add(reportHeaderDTO);

                when(commonUtility.getEodEndDate(anyString())).thenReturn("11-11-2000");
                new MockUp<JdbcTemplate>() {
                        @mockit.Mock
                    public List<ReportHeaderDTO> query(String sql, PreparedStatementSetter pss,
RowMapper<ReportHeaderDTO> rowMapper) {
                                return reportHeaderList;
                        }
                };

ResponseEntity<GetEmployeeDetailsResponseDTO> result = new ResponseEntity<>(response, HttpStatus.OK);

                try {
                        try {
                                when(connectionUtil.getURI(any(), any())).thenReturn(new URI("test"));
                        } catch (URISyntaxException e) {
                                Logger.error("Error {} ", e);
                        }
                        when(restTemplate.exchange(ArgumentMatchers.anyString(),
ArgumentMatchers.any(HttpMethod.class),
                                        ArgumentMatchers.<HttpEntity<?>> any(),
ArgumentMatchers.<Class<GetEmployeeDetailsResponseDTO>> any())).thenReturn(result);
                } catch (ServiceNotFoundException e) {
                        Logger.error("{}", e);
```



```java
Mockito.lenient().when(gcc.getESBCrudresponse(ArgumentMatchers.any(), ArgumentMatchers.anyString(),
ArgumentMatchers.anyString(), ArgumentMatchers.anyString())).thenReturn(esbCheckBookAddResponse);

            Mockito.lenient().when(gcc.getLatestCheckbookOrderInqResponse(ArgumentMatchers.anyString(),
ArgumentMatchers.anyString(),
ArgumentMatchers.anyString(),ArgumentMatchers.any())).thenReturn(latestCheckbookOrderInqResponse);

@ExtendWith(MockitoExtension.class)
@ContextConfiguration(classes= {ESBConfiguration.class},initializers =
{ConfigFileApplicationContextInitializer.class})
@TestPropertySource(properties = {"spring.config.location=classpath:RTL-PDM-D-CheckbookWS.yml"})
public class CheckbookServiceTest {

        @Mock
        @Qualifier("customRestTemplate")
        RestTemplate restTemplate;

        @Mock
        ObjectMapper objectMapper;

        @Test
        void testOpen() throws JsonMappingException, JsonProcessingException {
                CsaBranchFeedResponseDto data = new CsaBranchFeedResponseDto();
                String response = new String("{\"branchInfoFeed\" : [],  \"createDate\": \"\",
\"version\":\"\"}");
                ResponseEntity<String> result = new ResponseEntity<>(response, HttpStatus.OK);
                when(restTemplate.exchange(ArgumentMatchers.anyString(),
ArgumentMatchers.any(HttpMethod.class),
                                ArgumentMatchers.<HttpEntity<?>> any(), ArgumentMatchers.<Class<String>>
any())).thenReturn(result);

        when(objectMapper.readValue(ArgumentMatchers.anyString(),ArgumentMatchers.<Class<CsaBranchFeedResp
onseDto>> any())).thenReturn(data);
                ccasBranchFeedReader.open(executionContext);
        }
        @Test
        void testOpen() throws JsonMappingException, JsonProcessingException {
                CSAAPIResponseDTO csaData  = new CSAAPIResponseDTO();
                String response = new String("{\"version\": \"\",\"responseType\":\"\"}");
                ResponseEntity<String> result = new ResponseEntity<>(response, HttpStatus.OK);
                lenient().when(restTemplate.exchange(ArgumentMatchers.anyString(),
ArgumentMatchers.any(HttpMethod.class),
                                ArgumentMatchers.<HttpEntity<?>> any(), ArgumentMatchers.<Class<String>>
any())).thenReturn(result);
                String responseBody = result.getBody();

        lenient().when(objectMapper.readValue(ArgumentMatchers.anyString(),ArgumentMatchers.<Class<CSAAPIR
esponseDTO>> any())).thenReturn(csaData);
                csaNmlsFeedReader.open(executionContext);
        }

                lenient().when(rmsCrudConnector.getfetchValPropDetails()).thenReturn(rms1Data);
                lenient().doNothing().when(vpConfigRepo).deleteItems();

        javax.activation.DataSource ds = mock(javax.activation.DataSource.class);

                new MockUp<Transport>() {
                        @mockit.Mock
                    public void send(Message msg) {
                                return ;
                        }
                };
                EodExcelHelper.sendmail(ds, "filename", "CCS", "avoka@citi.com", "avoka@citi.com", "DEV",
new StringBuilder("emailbody"), "mail.smtp.host", "mail.citicorp.com", 2, "2000-10-10");
```



```java
        @Test
        void test() {
                ReflectionTestUtils.setField(enquireCheckStopPayService, "accountFormat", new
DecimalFormat("000000000000"));

@ExtendWith(MockitoExtension.class)

class CacOrScLinkInquiryServiceTest {

        @InjectMocks
        CacOrScLinkInquiryService cacOrScLinkInquiryService;

        @Mock
        GCIF6043Service gcif6043Service;
when(gcif6043Service.getCACSCLinksInquiry(ArgumentMatchers.any(), ArgumentMatchers.anyString(),
                                ArgumentMatchers.anyString())).thenReturn(cacScLinksInquiry);

                cacOrScLinkInquiryService.getCacOrScLinkGroupResponse(new CacOrScLinkGroupRequest(),
"123445566", "e23433ad67");

when(stopPayCheckInImpacsService.impacsAddStopPay(ArgumentMatchers.any(), ArgumentMatchers.anyString(),
ArgumentMatchers.anyString())).thenReturn(response);
                assertNotNull(stopPayCheckService.addStopPaycheck(addCheckStopPayRequest , "123456789",
"RTLAO"));

        @Test
        void testcommonUtility() {
                commonUtility.getHttpHeaders("123456", "GCB", "US", "RTLAO", "client id");
                commonUtility.getHeaderMap("123456", "GCB", "US", "RTLAO");
                commonUtility.getErrorResponse(TypeEnum.ERROR, "400", "bad request", "location",
"moreInfo");
        }

@WebMvcTest(value = ESBCheckbookApi.class)

@ExtendWith(MockitoExtension.class)
public class CheckbookControllerTest {

        @MockBean
        CheckbookService checkbookService;

        @Autowired
        private MockMvc mockMvc;

final ESBCheckBookAddResponse response=new ESBCheckBookAddResponse();

                final ObjectMapper mapper = new ObjectMapper();
                final String req = mapper.writeValueAsString(checkbookOrderAddRequest);

                final HttpHeaders  headers = new HttpHeaders();
                headers.add("client_id", "BP20071");
                headers.add("Authorization", "authorize");
                headers.add("uuid", "1234");
                headers.add("Accept", "application/json");

        when(this.checkbookService.placeCheckbookRequest(ArgumentMatchers.any(),ArgumentMatchers.anyString
())).thenReturn(response);

                response.setCheckbookOrderAddRs(checkbookOrderAddRs);
                // My request
                final RequestBuilder requestBuilder =
MockMvcRequestBuilders.post("/private/v1/bank/fulfillment/checkbook").headers(headers).content(req).conten
tType(MediaType.APPLICATION_JSON);
                final MvcResult result = this.mockMvc.perform(requestBuilder).andReturn();
                Assertions.assertEquals(200, result.getResponse().getStatus());
```



```java
        }
@BeforeEach
        void setUp() throws Exception {

MockitoAnnotations.initMocks(this);

}



@Test
public void test1() {
assertNotNull(USRAOServicingOrderAPIControllerTest.checksAsCashsafetyChecklinks("clientId",
"authorization","application/json", "123456", "GCB", "US", "CBOL", "application/json", new
CacOrScLinkGroupRequest()));
        }

import org.mockito.ArgumentMatchers;

when(gcifCrudConnector.getGCIFCrudresponse(ArgumentMatchers.any(), ArgumentMatchers.anyString(),
ArgumentMatchers.anyString(), ArgumentMatchers.anyString(), ArgumentMatchers.anyString(),
ArgumentMatchers.anyString())).thenReturn(g6711Response1 );

Assertions.assertThrows(Exception.class, () ->
gCIF6043Service.getCACSCLinksInquiry(cacOrScLinkGroupRequest,"123445566","e23433ad67"));

ObjectMapper objectMapper = new ObjectMapper();

GCIFMessage g6711Response1 = null;
                String response1 = "{ \"ElementProtocolDesignator\": { \"FunctionHeaderResponseEnvelope\":
{ \"CGMSReturnCode\": \"0000\", \"CGMSReasonText\": \"OK\" }, \"FirstInitiatorInformationGroup\": {
\"EmployeeIDType\": \"03\", \"OperatorID\": \"AS78041\"} } } } } }";

                try {
                        g6711Response1 = objectMapper.readValue(response1, GCIFMessage.class);
                        } catch (JsonMappingException e) {
                                //LOG.error("Exception occured", e);
                        } catch (JsonProcessingException e) {
                                //LOG.error("Exception occured", e);
                        }

                when(gcifCrudConnector.getGCIFCrudresponse(ArgumentMatchers.any(),
ArgumentMatchers.anyString(), ArgumentMatchers.anyString(), ArgumentMatchers.anyString(),
ArgumentMatchers.anyString(), ArgumentMatchers.anyString())).thenReturn(g6711Response1);
                Assertions.assertThrows(Exception.class, () ->
gCIF6711Service.getComponentDetailInformation(fii ,
componentIdentifierGroup,"123445566","e23433ad67",componentInquiryFlagsGroup));

when(tpsConnector.getTPSResponse(ArgumentMatchers.any(), ArgumentMatchers.anyString(),
ArgumentMatchers.anyString(), ArgumentMatchers.anyString())).thenReturn(tpsE5999Response);
                assertNotNull(tps60505Service.getE5999Data(addCheckStopPayRequest , "123456789",
"RTLAO"));


@Test
        void testgetTpsResponseInStringFallBack() {

                TPSMessage tpsRequest  = null;
                 Assertions.assertThrows(Exception.class, () ->
tpsConnector.getTPSResponseInStringFallBack(tpsRequest ));
```