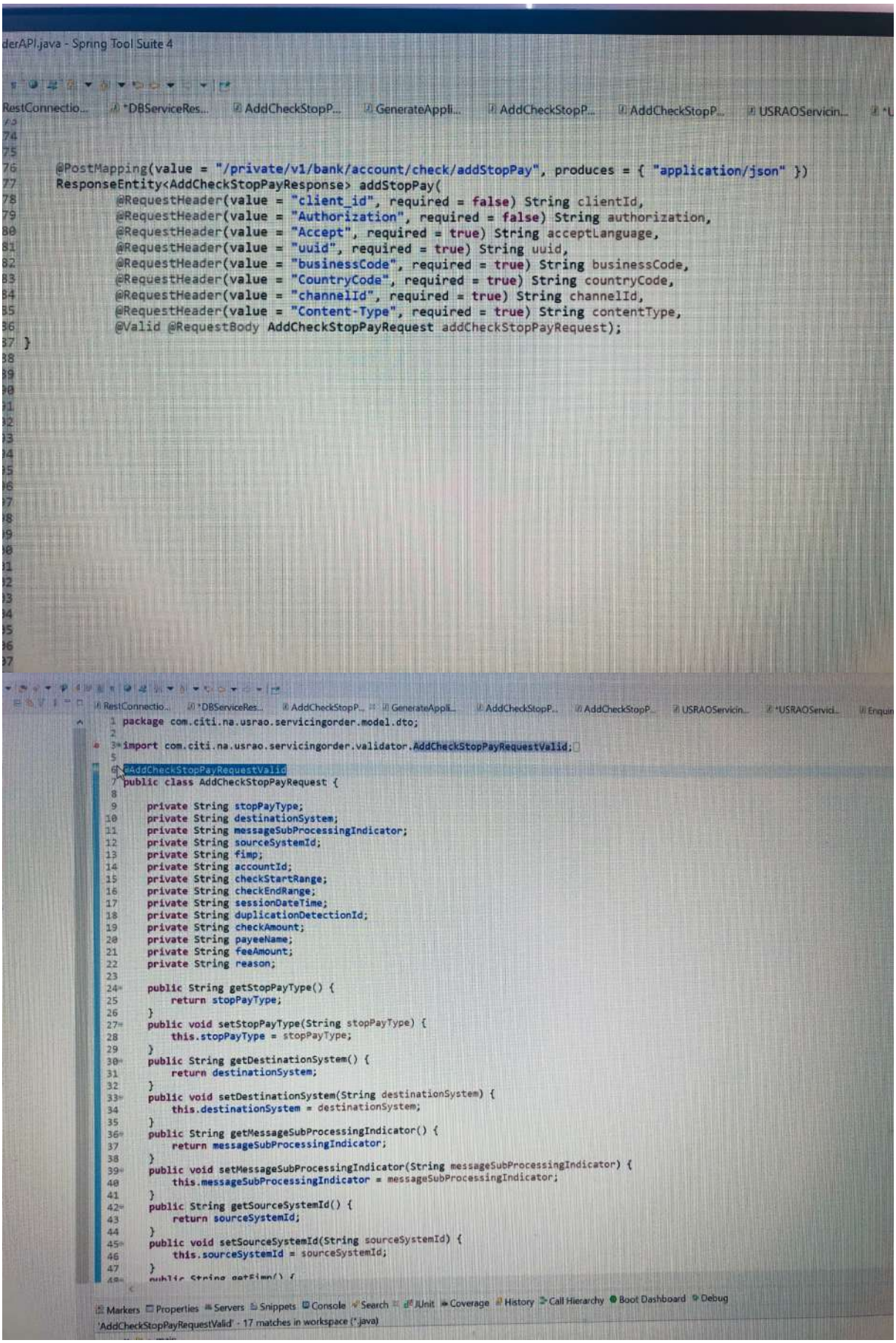
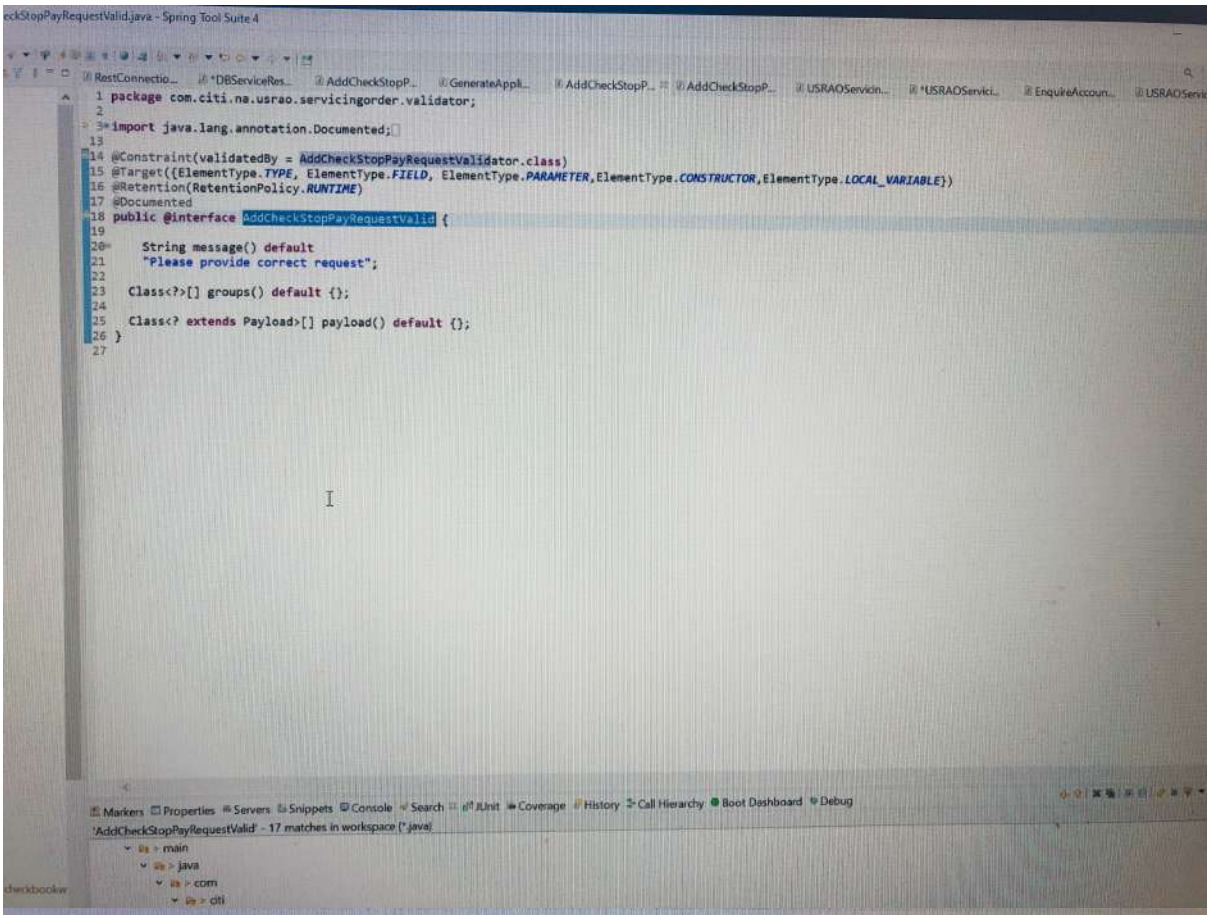
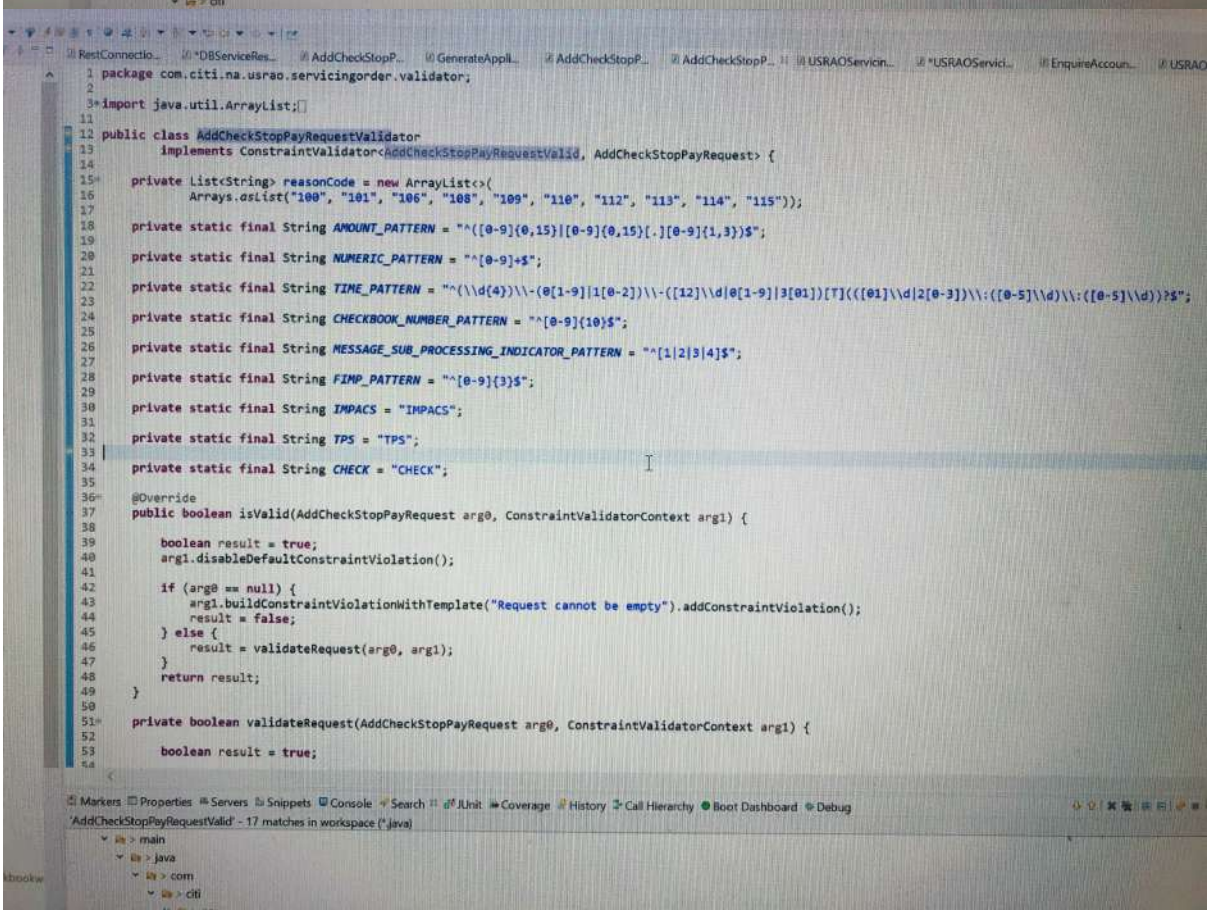
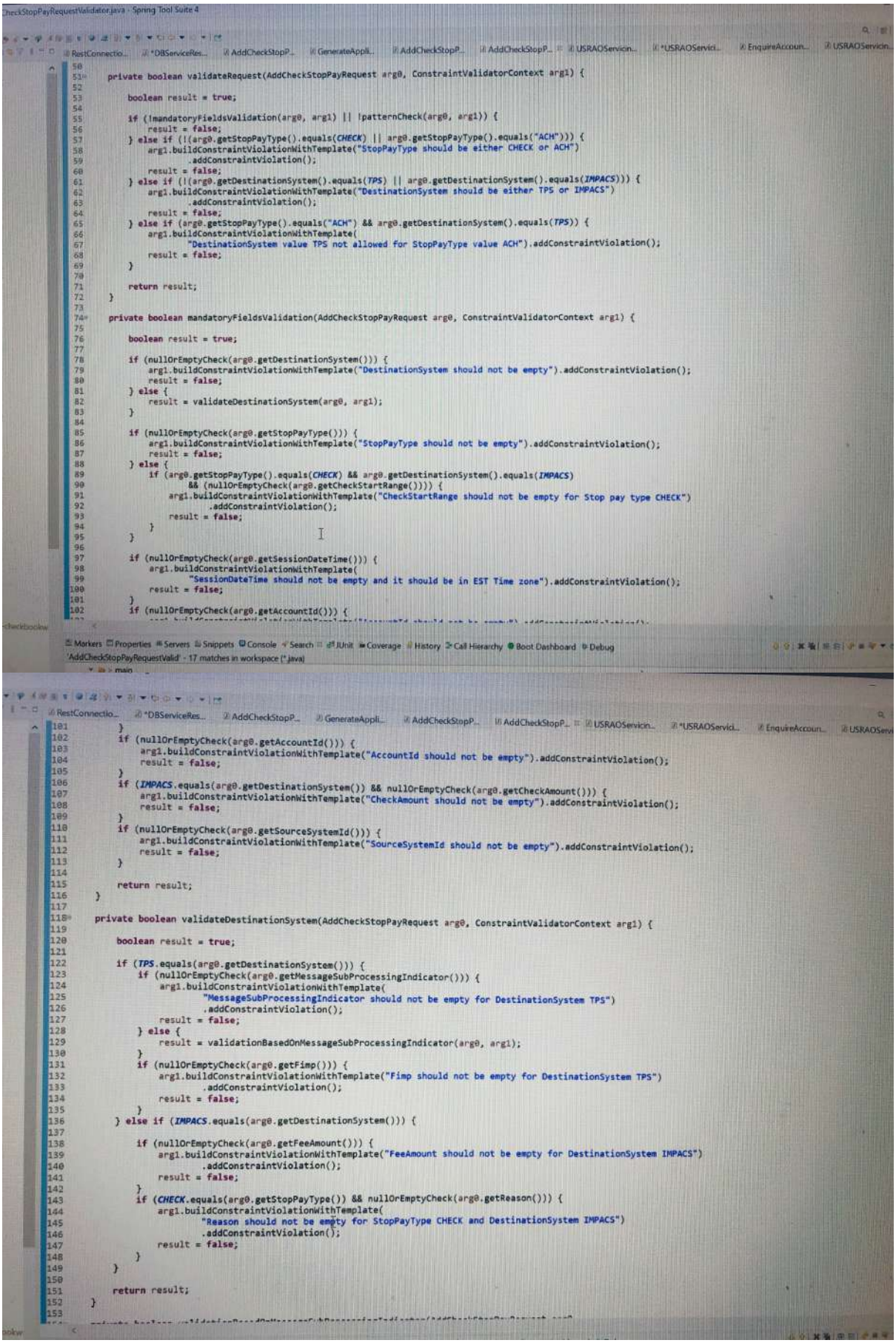

derAPI.java - Spring Tool Suite 4
RestConnectio...
*DBServiceRes...
AddCheckStopP...
GenerateAppli...
AddCheckStopP...
AddCheckStopP...
USRAOServicin...
@PostMapping(value = "/private/v1/bank/account/check/addStopPay", produces = { "application/json" })
ResponseEntity<AddCheckStopPayResponse> addStopPay(
@RequestHeader(value = "client_id", required = false) String clientId,
@RequestHeader(value = "Authorization", required = false) String authorization,
@RequestHeader(value = "Accept", required = true) String acceptLanguage,
@RequestHeader(value = "uuid", required = true) String uuid,
@RequestHeader(value = "businessCode", required = true) String businessCode,
@RequestHeader(value = "CountryCode", required = true) String countryCode,
@RequestHeader(value = "channelId", required = true) String channelId,
@RequestHeader(value = "Content-Type", required = true) String contentType,
@Valid @RequestBody AddCheckStopPayRequest addCheckStopPayRequest);
}
RestConnectio...
*DBServiceRes...
AddCheckStopP...
GenerateAppli...
AddCheckStopP...
AddCheckStopP...
USRAOServicin...
*USRAOServici...
Enquir
package com.citi.na.usrao.servicingorder.model.dto;
import com.citi.na.usrao.servicingorder.validator.AddCheckStopPayRequestValid;
@AddCheckStopPayRequestValid
public class AddCheckStopPayRequest {
private String stopPayType;
private String destinationSystem;
private String messageSubProcessingIndicator;
private String sourceSystemId;
private String fimp;
private String accountId;
private String checkStartRange;
private String checkEndRange;
private String sessionDateTime;
private String duplicationDetectionId;
private String checkAmount;
private String payeeName;
private String feeAmount;
private String reason;
public String getStopPayType() {
return stopPayType;
}
public void setStopPayType(String stopPayType) {
this.stopPayType = stopPayType;
}
public String getDestinationSystem() {
return destinationSystem;
}
public void setDestinationSystem(String destinationSystem) {
this.destinationSystem = destinationSystem;
}
public String getMessageSubProcessingIndicator() {
return messageSubProcessingIndicator;
}
public void setMessageSubProcessingIndicator(String messageSubProcessingIndicator) {
this.messageSubProcessingIndicator = messageSubProcessingIndicator;
}
public String getSourceSystemId() {
return sourceSystemId;
}
public void setSourceSystemId(String sourceSystemId) {
this.sourceSystemId = sourceSystemId;
}
Markers
Properties
Servers
Snippets
Console
Search
JUnit
Coverage
History
Call Hierarchy
Boot Dashboard
Debug
'AddCheckStopPayRequestValid' - 17 matches in workspace (*.java)



```java
package com.citi.na.usrao.servicingorder.validator;

import java.lang.annotation.Documented;

@Constraint(validatedBy = AddCheckStopPayRequestValidator.class)
@Target({ElementType.TYPE, ElementType.FIELD, ElementType.PARAMETER,ElementType.CONSTRUCTOR,ElementType.LOCAL_VARIABLE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface AddCheckStopPayRequestValid {

    String message() default
    "Please provide correct request";

  Class<?>[] groups() default {};

  Class<? extends Payload>[] payload() default {};
}
```

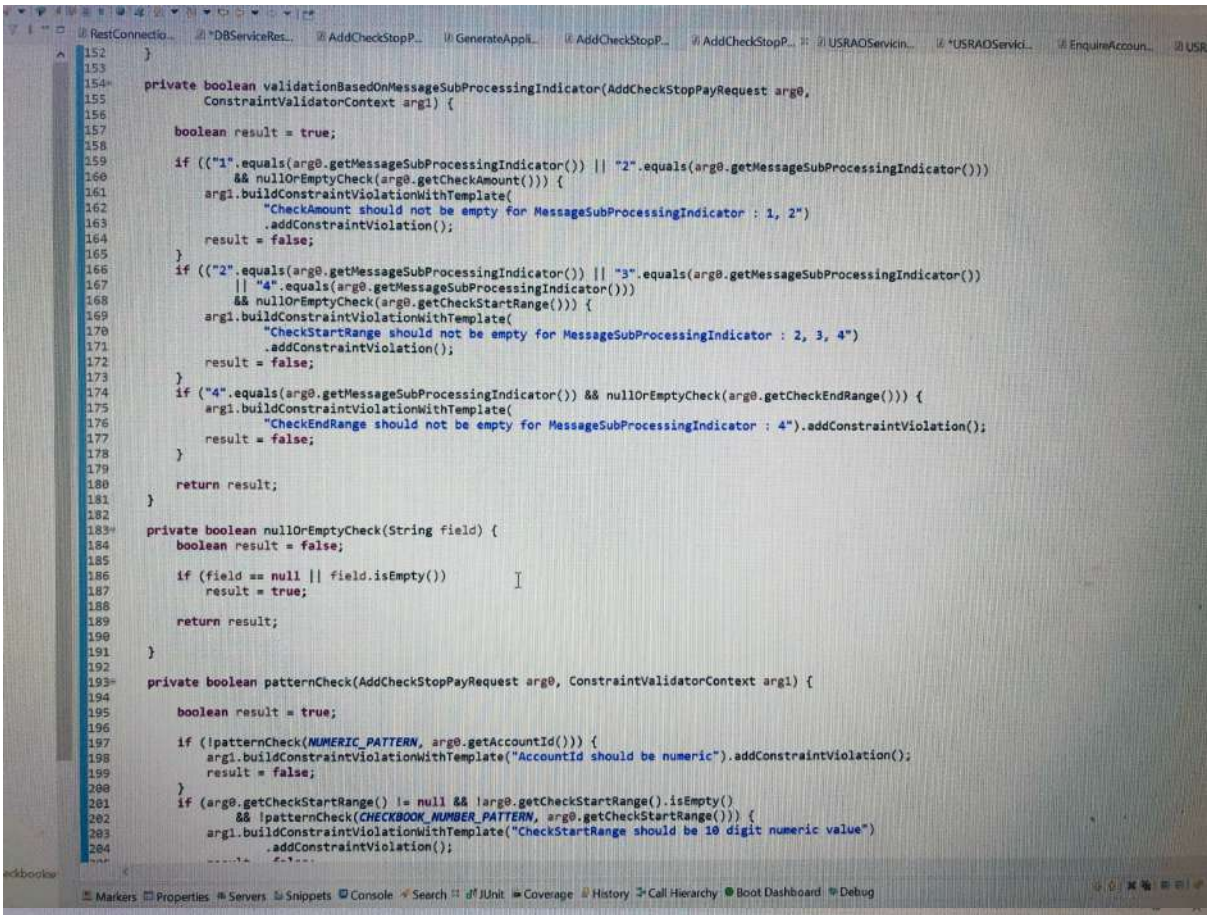


```java
package com.citi.na.usrao.servicingorder.validator;

import java.util.ArrayList;

public class AddCheckStopPayRequestValidator
        implements ConstraintValidator<AddCheckStopPayRequestValid, AddCheckStopPayRequest> {

    private List<String> reasonCode = new ArrayList<>(
            Arrays.asList("100", "101", "106", "108", "109", "110", "112", "113", "114", "115"));

    private static final String AMOUNT_PATTERN = "^([0-9]{0,15}|[0-9]{0,15}[.][0-9]{1,3})$";

    private static final String NUMERIC_PATTERN = "^[0-9]+$";

    private static final String TIME_PATTERN = "^(\\d{4})\\-(0[1-9]|1[0-2])\\-([12]\\d|0[1-9]|3[01])[T](([01]\\d|2[0-3])\\:([0-5]\\d)\\:([0-5]\\d))?$";

    private static final String CHECKBOOK_NUMBER_PATTERN = "^[0-9]{10}$";

    private static final String MESSAGE_SUB_PROCESSING_INDICATOR_PATTERN = "^[1|2|3|4]$";

    private static final String FIMP_PATTERN = "^[0-9]{3}$";

    private static final String IMPACS = "IMPACS";

    private static final String TPS = "TPS";

    private static final String CHECK = "CHECK";

    @Override
    public boolean isValid(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {

        boolean result = true;
        arg1.disableDefaultConstraintViolation();

        if (arg0 == null) {
            arg1.buildConstraintViolationWithTemplate("Request cannot be empty").addConstraintViolation();
            result = false;
        } else {
            result = validateRequest(arg0, arg1);
        }
        return result;
    }

    private boolean validateRequest(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {

        boolean result = true;
```

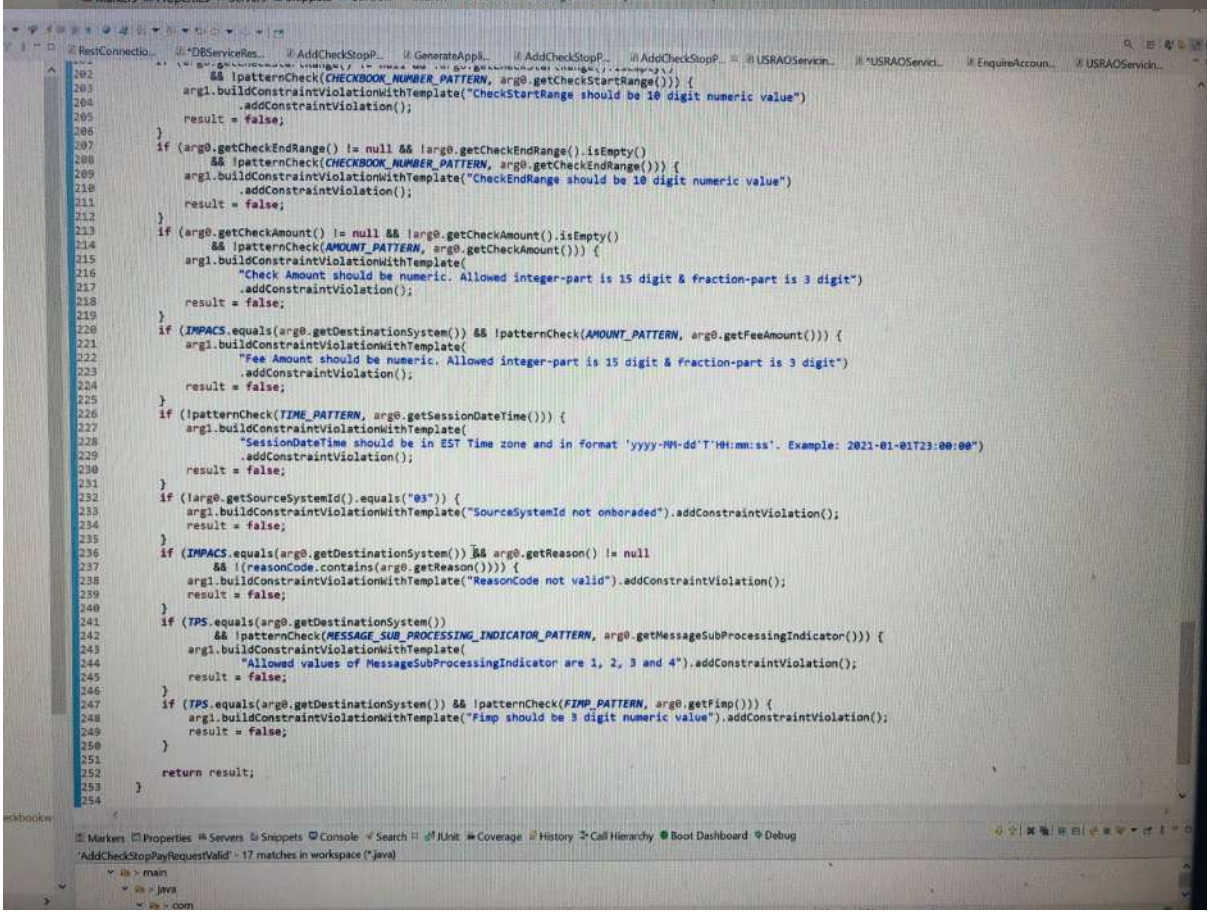


```
    private boolean validateRequest(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {

        boolean result = true;

        if (!mandatoryFieldsValidation(arg0, arg1) || !patternCheck(arg0, arg1)) {
            result = false;
        } else if (!(arg0.getStopPayType().equals(CHECK) || arg0.getStopPayType().equals("ACH"))) {
            arg1.buildConstraintViolationWithTemplate("StopPayType should be either CHECK or ACH")
                    .addConstraintViolation();
            result = false;
        } else if (!(arg0.getDestinationSystem().equals(TPS) || arg0.getDestinationSystem().equals(IMPACS))) {
            arg1.buildConstraintViolationWithTemplate("DestinationSystem should be either TPS or IMPACS")
                    .addConstraintViolation();
            result = false;
        } else if (arg0.getStopPayType().equals("ACH") && arg0.getDestinationSystem().equals(TPS)) {
            arg1.buildConstraintViolationWithTemplate(
                    "DestinationSystem value TPS not allowed for StopPayType value ACH").addConstraintViolation();
            result = false;
        }

        return result;
    }

    private boolean mandatoryFieldsValidation(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {

        boolean result = true;

        if (nullOrEmptyCheck(arg0.getDestinationSystem())) {
            arg1.buildConstraintViolationWithTemplate("DestinationSystem should not be empty").addConstraintViolation();
            result = false;
        } else {
            result = validateDestinationSystem(arg0, arg1);
        }

        if (nullOrEmptyCheck(arg0.getStopPayType())) {
            arg1.buildConstraintViolationWithTemplate("StopPayType should not be empty").addConstraintViolation();
            result = false;
        } else {
            if (arg0.getStopPayType().equals(CHECK) && arg0.getDestinationSystem().equals(IMPACS)
                    && (nullOrEmptyCheck(arg0.getCheckStartRange()))) {
                arg1.buildConstraintViolationWithTemplate("CheckStartRange should not be empty for Stop pay type CHECK")
                        .addConstraintViolation();
                result = false;
            }
        }

        if (nullOrEmptyCheck(arg0.getSessionDateTime())) {
            arg1.buildConstraintViolationWithTemplate(
                    "SessionDateTime should not be empty and it should be in EST Time zone").addConstraintViolation();
            result = false;
        }
        if (nullOrEmptyCheck(arg0.getAccountId())) {
```

```
        }
        if (nullOrEmptyCheck(arg0.getAccountId())) {
            arg1.buildConstraintViolationWithTemplate("AccountId should not be empty").addConstraintViolation();
            result = false;
        }
        if (IMPACS.equals(arg0.getDestinationSystem()) && nullOrEmptyCheck(arg0.getCheckAmount())) {
            arg1.buildConstraintViolationWithTemplate("CheckAmount should not be empty").addConstraintViolation();
            result = false;
        }
        if (nullOrEmptyCheck(arg0.getSourceSystemId())) {
            arg1.buildConstraintViolationWithTemplate("SourceSystemId should not be empty").addConstraintViolation();
            result = false;
        }

        return result;
    }

    private boolean validateDestinationSystem(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {

        boolean result = true;

        if (TPS.equals(arg0.getDestinationSystem())) {
            if (nullOrEmptyCheck(arg0.getMessageSubProcessingIndicator())) {
                arg1.buildConstraintViolationWithTemplate(
                        "MessageSubProcessingIndicator should not be empty for DestinationSystem TPS")
                        .addConstraintViolation();
                result = false;
            } else {
                result = validationBasedOnMessageSubProcessingIndicator(arg0, arg1);
            }
            if (nullOrEmptyCheck(arg0.getFimp())) {
                arg1.buildConstraintViolationWithTemplate("Fimp should not be empty for DestinationSystem TPS")
                        .addConstraintViolation();
                result = false;
            }
        } else if (IMPACS.equals(arg0.getDestinationSystem())) {

            if (nullOrEmptyCheck(arg0.getFeeAmount())) {
                arg1.buildConstraintViolationWithTemplate("FeeAmount should not be empty for DestinationSystem IMPACS")
                        .addConstraintViolation();
                result = false;
            }
            if (CHECK.equals(arg0.getStopPayType()) && nullOrEmptyCheck(arg0.getReason())) {
                arg1.buildConstraintViolationWithTemplate(
                        "Reason should not be empty for StopPayType CHECK and DestinationSystem IMPACS")
                        .addConstraintViolation();
                result = false;
            }
        }

        return result;
    }
```



```
    }

    private boolean validationBasedOnMessageSubProcessingIndicator(AddCheckStopPayRequest arg0,
            ConstraintValidatorContext arg1) {

        boolean result = true;

        if (("1".equals(arg0.getMessageSubProcessingIndicator()) || "2".equals(arg0.getMessageSubProcessingIndicator()))
                && nullOrEmptyCheck(arg0.getCheckAmount())) {
            arg1.buildConstraintViolationWithTemplate(
                    "CheckAmount should not be empty for MessageSubProcessingIndicator : 1, 2")
                    .addConstraintViolation();
            result = false;
        }
        if (("2".equals(arg0.getMessageSubProcessingIndicator()) || "3".equals(arg0.getMessageSubProcessingIndicator())
                || "4".equals(arg0.getMessageSubProcessingIndicator()))
                && nullOrEmptyCheck(arg0.getCheckStartRange())) {
            arg1.buildConstraintViolationWithTemplate(
                    "CheckStartRange should not be empty for MessageSubProcessingIndicator : 2, 3, 4")
                    .addConstraintViolation();
            result = false;
        }
        if ("4".equals(arg0.getMessageSubProcessingIndicator()) && nullOrEmptyCheck(arg0.getCheckEndRange())) {
            arg1.buildConstraintViolationWithTemplate(
                    "CheckEndRange should not be empty for MessageSubProcessingIndicator : 4").addConstraintViolation();
            result = false;
        }

        return result;
    }

    private boolean nullOrEmptyCheck(String field) {
        boolean result = false;

        if (field == null || field.isEmpty())
            result = true;

        return result;

    }

    private boolean patternCheck(AddCheckStopPayRequest arg0, ConstraintValidatorContext arg1) {

        boolean result = true;

        if (!patternCheck(NUMERIC_PATTERN, arg0.getAccountId())) {
            arg1.buildConstraintViolationWithTemplate("AccountId should be numeric").addConstraintViolation();
            result = false;
        }
        if (arg0.getCheckStartRange() != null && !arg0.getCheckStartRange().isEmpty()
                && !patternCheck(CHECKBOOK_NUMBER_PATTERN, arg0.getCheckStartRange())) {
            arg1.buildConstraintViolationWithTemplate("CheckStartRange should be 10 digit numeric value")
                    .addConstraintViolation();
```



```
                && !patternCheck(CHECKBOOK_NUMBER_PATTERN, arg0.getCheckStartRange())) {
            arg1.buildConstraintViolationWithTemplate("CheckStartRange should be 10 digit numeric value")
                    .addConstraintViolation();
            result = false;
        }
        if (arg0.getCheckEndRange() != null && !arg0.getCheckEndRange().isEmpty()
                && !patternCheck(CHECKBOOK_NUMBER_PATTERN, arg0.getCheckEndRange())) {
            arg1.buildConstraintViolationWithTemplate("CheckEndRange should be 10 digit numeric value")
                    .addConstraintViolation();
            result = false;
        }
        if (arg0.getCheckAmount() != null && !arg0.getCheckAmount().isEmpty()
                && !patternCheck(AMOUNT_PATTERN, arg0.getCheckAmount())) {
            arg1.buildConstraintViolationWithTemplate(
                    "Check Amount should be numeric. Allowed integer-part is 15 digit & fraction-part is 3 digit")
                    .addConstraintViolation();
            result = false;
        }
        if (IMPACS.equals(arg0.getDestinationSystem()) && !patternCheck(AMOUNT_PATTERN, arg0.getFeeAmount())) {
            arg1.buildConstraintViolationWithTemplate(
                    "Fee Amount should be numeric. Allowed integer-part is 15 digit & fraction-part is 3 digit")
                    .addConstraintViolation();
            result = false;
        }
        if (!patternCheck(TIME_PATTERN, arg0.getSessionDateTime())) {
            arg1.buildConstraintViolationWithTemplate(
                    "SessionDateTime should be in EST Time zone and in format 'yyyy-MM-dd'T'HH:mm:ss'. Example: 2021-01-01T23:00:00")
                    .addConstraintViolation();
            result = false;
        }
        if (!arg0.getSourceSystemId().equals("03")) {
            arg1.buildConstraintViolationWithTemplate("SourceSystemId not onboraded").addConstraintViolation();
            result = false;
        }
        if (IMPACS.equals(arg0.getDestinationSystem()) && arg0.getReason() != null
                && !(reasonCode.contains(arg0.getReason()))) {
            arg1.buildConstraintViolationWithTemplate("ReasonCode not valid").addConstraintViolation();
            result = false;
        }
        if (TPS.equals(arg0.getDestinationSystem())
                && !patternCheck(MESSAGE_SUB_PROCESSING_INDICATOR_PATTERN, arg0.getMessageSubProcessingIndicator())) {
            arg1.buildConstraintViolationWithTemplate(
                    "Allowed values of MessageSubProcessingIndicator are 1, 2, 3 and 4").addConstraintViolation();
            result = false;
        }
        if (TPS.equals(arg0.getDestinationSystem()) && !patternCheck(FIMP_PATTERN, arg0.getFimp())) {
            arg1.buildConstraintViolationWithTemplate("Fimp should be 3 digit numeric value").addConstraintViolation();
            result = false;
        }

        return result;
    }
```