**3 February – 12 March / 2021**

**İSTANBUL TECHNICAL UNIVERSITY**

**Faculty of Computer Science and Informatics**

**INTERNSHIP PROGRAM REPORT**

**Cihan GÜZEL**

**150140302**

**INTERNSHIP PROGRAM: Software**

**Development of Java Integration Services and Applications**

**İSTANBUL TECHNICAL UNIVERSITY**

**Faculty of Computer Science and Informatics**

**INTERNSHIP PROGRAM ACTIVITY REPORT**

Academic Year: 2020 - 2021

Period of Training: Spring

**Student Information**

|  |  |
| --- | --- |
| Name Surname: | Cihan Güzel |
| Student ID: | 150140302 |
| Department: | Computer Engineering |
| Program: | -%100 English |
| E-Mail: | guzelcih@itu.edu.tr |
| Mobile Phone: | +90 534 017 52 59 |
| Pursuing a Double Major? | No |
| Graduated? | No |
| Taking a class at Summer School? | No |

**Institution Information**

|  |  |
| --- | --- |
| Company Name: | Aktif Yatırım Bankası A.Ş. |
| Department: | Card Payment Systems |
| Web Address: | aktifbank.com.tr |
| Postal Address: | Aktif Bank Genel Müdürlük Esentepe Mahallesi Kore Şehitleri Caddesi No: 8/1 Şişli İSTANBUL |

**Authorized Person Information**

|  |  |
| --- | --- |
| Department: | Card Payment Systems |
| Title: | Senior Director |
| Name Surname: | Özgür Bilgin |
| Corporate E-Mail: | ozgur.bilgin@aktifbank.com.tr  No signatures or stamps are required on this page. |
| Corporate Phone: |  |

**Internship Program Information**

|  |  |
| --- | --- |
| Location | Turkey |
| Starting Date | 03.02.2021 |
| End Date | 12.03.2021 |
| Number of Days Worked | 27 |
| During your internship, did you have insurance? | Yes, I was insured by İTÜ. |

**TABLE OF CONTENTS**

1. INFORMATION ABOUT THE INSTITUTION …………………….…………………...1
2. INTRODUCTION ...……………………………………………………………………….3
3. DESCRIPTION AND ANALYSIS OF THE INTERNSHIP PROJECT ………………5
4. IMPRESSIONS ON INTERNSHIP EXPERIENCE……………………………………10
5. CONCLUSIONS …………………………………………………………………………10
6. REFERENCES .………………………………………………………………………….11
7. APPENDIX ……………………………………………………………………………….12
8. **INFORMATION ABOUT THE INSTITUTION**

Turkey's largest privately-owned investment bank, Aktif Bank in Turkey with innovative business models and technology investments that reinterprets the largest financial technology investment banking continues to operate as an ecosystem.

Aktif Bank, with products it offers physical and digital channels in Turkey and the global financial technology companies and business partners as well as competitors are moving. Having a customer-oriented approach in its genetic codes, the Bank, together with its subsidiaries and business partners, appeals to more than 10 million customers in more than 10 business lines from loans to insurance, transportation to card services, payment systems to entertainment world at every point.

Aktif Bank with 12 branches located in Turkey, a predominantly require a communication corporate banking, investment banking and private banking focus. Aktif Bank, which owns 18% of the retail customer market with its effective collaborations in retail banking, innovative business models and digital channel investments, today provides service with its subsidiaries and approximately one thousand employees. Aktif Bank also undertakes the name sponsorship of the Istanbul Marathon with its digital bank N Kolay, which makes life easier and responds quickly to the needs of individuals anytime, anywhere, without a branch. The dynamism and egalitarian, lean service model of the N Kolay brand; It turns into a harmonious cooperation by coming together with the spirit of the Istanbul Marathon, one of the most important marathons in the world.

Aktif Bank, another player of the digital world, provides easy access not only to matches but also to all events from concerts to theater, with its mobile application Passo. Continuously developing the Passo application, the Bank diversifies the advantages it provides in stadiums with opportunities such as 24/7 money transfer with QR, shopping and instant loans. With Passolig, the key to safety and comfort in the stands, Aktif Bank both supports football clubs and offers opportunities and privileges to its fans.

1. **INTRODUCTION**

Between the term break of 2020-2021 season, I performed my internship at Aktif Bank, which is the institution where I currently continue working as well. I had a chance to work on an software development team which belongs to a friendly and supportive environment and they helped me adopt the workplace and provided the informations that I needed during the projects. During the internship, I had opportunites to attend some seminars related to digital banking from experts of these subjects. I was the part of Integration Application Development Team of the Card Payment Systems Department. Each team member explained what their role is and what type of projects they were working on as well as what were my role and tasks in this group that I joined.

Througout the internship, I took part in various projects that I had no experience before and had a very educating chance to learn the technologies and tools used in software devolopment and production line. As a project management process, Jira which is a work management tool for all kind of software development cases, from requirements and test case management to Agile Scrum methodology is used for assigning tasks to developers for every week. I used Java programming language for developing J2EE applications, GUI’s with the help of Java SWING. I also developed web application and services over SOAP protocol, used database management system from Oracle Database and Graymound Framework which provides Service Oriented Architecture wise development and aims to give a acceptable interface for using enterprise components of Java.

1. **DESCRIPTION AND ANALYSIS OF THE INTERNSHIP PROJECT**

Througout my internship, I have been given many tasks which includes Desktop Applications using Graymound GUIML interface and for the business development I coded services on a unique structure provided by Graymound Framework. All the functionalities we developed were structured as a services so that it will be available for any channel on the network which has access. Swagger was the main tool used to define the structures and documentation of RESTful services and api’s that I coded which expressed as JSON. As an important part of development, I have experienced a software development workflow which includes Development-Test-UAT(User Acceptance Test)-Production line. For the database management system, I have coded procedures and functions on Oracle database that I used in GUI’s and services I developed.

1. **JAVA DESKTOP APPLICATION FOR MONITORING**

**BLOCKS ON PREPAID CARDS**

In this projects, I developed a java desktop application on Eclipse IDE to monitor blocks on prepaid cards for various reasons. On the monitoring screen, some required information has to be taken as an input and then listed all the prepaid cards filtered according to restrictions. A data field which has to be filled with customer number has to list all the customers if required as and mini window of List of Values. For the purpose of getting the data from database, Oracle JDBC functionalites is used to establish the connection.

* 1. **Technologies**
* **Java:** Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.
* **JDBC:** The Java Database Connectivity (JDBC) API provides universal data access from the Java programming language. Using the JDBC API, you can access virtually any data source, from relational databases to spreadsheets and flat files.
* **Swing Components under Graymound Framework:** Graymound GUIML is and interface that reduce designing time by providing an Eclipse plug-in for SWING. Basically, created designer provides a bug-less, simple, fast, user-friendly interface.
  1. **Project Details**

It is requested a monitoring screen that takes these filtering fields to get the proper list of customer’s blocks. Also, data field that takes the customer no has to be a list of values (LOV) which shows customer’s to choose from pp\_bloke\_koy table from database via the XML file created separately. Fields on the screen are;

1. Customer No
2. Card No
3. Reference No of Block
4. Date Range
5. Active/Passive Blocks

For the LOV file, a database operation has to be made also whose criteria are as following;

1. Name-Surname
2. Customer No
3. Reference No of Block

Blocks that are to be listed should include following information about customer on grid layout;

1. Card No
2. Customer No
3. Name-Surname
4. National Identity No
5. Description
6. Status
7. Reference No of Blocks
8. Card Balance
9. Reason Code of Block
10. Date of Blocking
11. Date of Unblocking
    1. **Design of the Project and Explanation of Codes**

Layout design of the monitoring screen made on Eclipse IDE by the help of Graymound GUIML Framework. These layouts have standard look for the all monitoringss applications that has to meet the requirements.

Criteria section of panel and List section were designed like following;

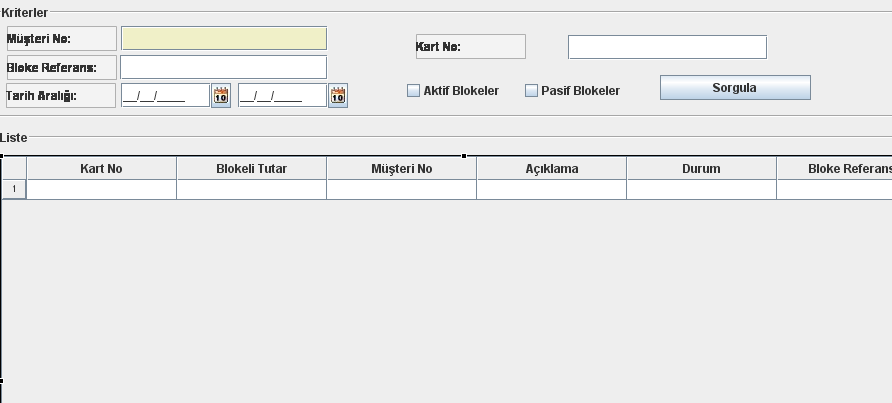


Figure 1: Layout Design of Monitoring Screen

Data Field that takes the Customer No has to show the list of customers from pp\_bloke\_koy as a pop-up window to chose from.

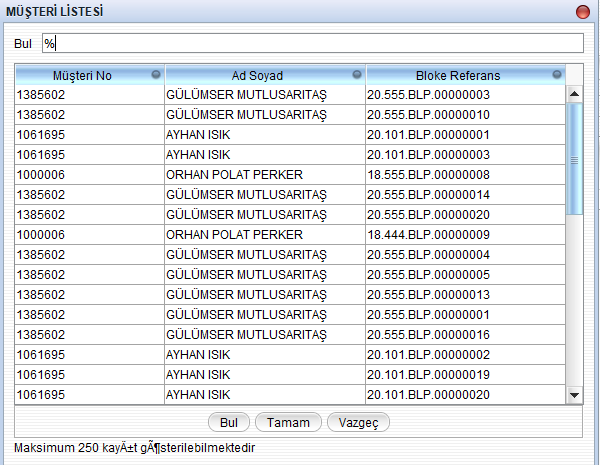


Figure 2: List of Values Shows Customer Information

Due to the service oriented architecture, all the functionalities used are made as services. For the purpose of query function, a query service is created to get the data from database and declared on the services section of application with the inputs.

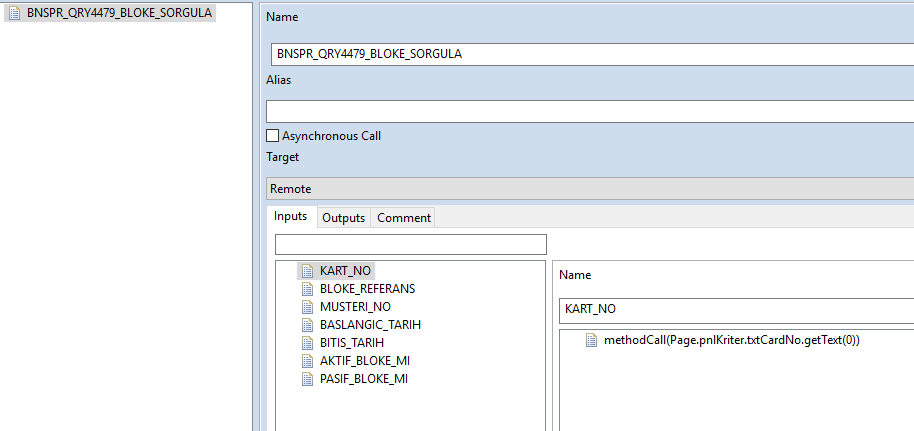


Figure 3: Services Used in Monitoring Screen

Actions of the application which have to meet the requirements of standardized design are determined and declared under the actions section of application.

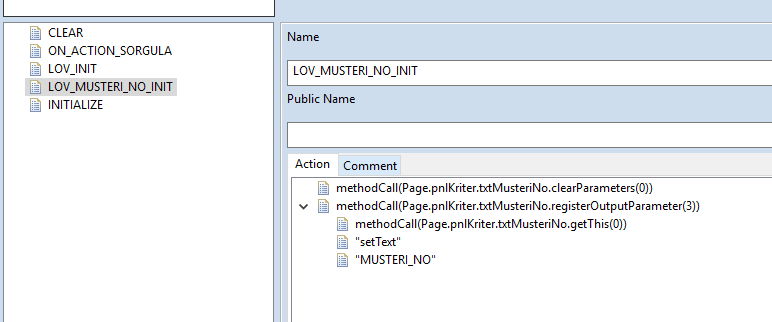


Figure 4: List of Actions in Monitoring Screen

BNSPR\_QRY4479\_BLOKE\_SORGULA is the main service that I implemented my query operation for this application. In Graymound Framework, all the services written take and give a specialized Hash Map extension as input and output which is called GMMap.

We have a project called BNSPRDAL and most common functions were declared under DalUtil class file which includes establishing JDBC connection under Graymound Framework.

This service and any other functionalities that will be added to this screen have to be developed under CreditCardQRY4479Services class I created.

Before the implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA, a function is required to check the input validation so that on the monitoring screen, at least one criteria is sent to this service because it is required for query operation.

ValidateRequest class gets the input map and checks whether any filtering field is sent to service and if not, a runtime exception is thrown saying “It is required at least one criteria.”

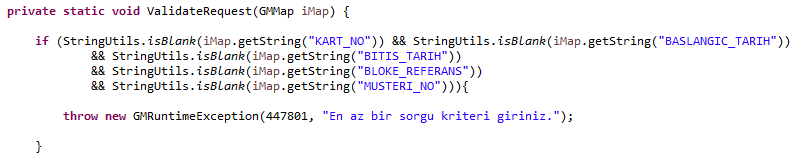


Figure 5: ValidateRequest Class

BNSPR\_QRY4479\_BLOKE\_SORGULA service first calls the ValidateRequest function for the purpose of input validation. We get the type of block type which is active or passive.

After that query statement is written as a string with the required parameters. Results of this query statement fills the table named TABLE\_LIST which we will add it to our GMMap as an output later on.

For the inputs of query function, a list of java object is created with the types of those inputs.

Then, I made the database connection, by using our standardized JDBC connection function.



Figure 6: Implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA Service-1

Database connections for common implementation is made by a function called callOracleRefCursorFunction under DALUtil class. Impementation of this class as follows;

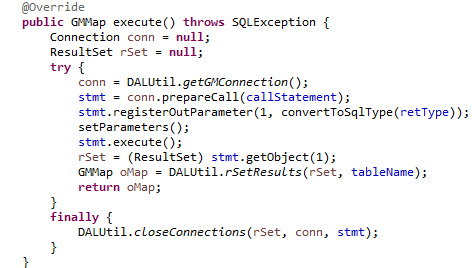


Figure 7: Implementation of callOracleRefCursorFunction Class

We first establish a connection, query the database operation and give output parameter we expect from SQL function. Resultset returning from function with the table name is assigned the GMMap hash map.

To list the blocks of customer, we called a query statement which is a SQL function under PKG\_TRN4478 package. If required any functions or procedures related to this monitoring screen is declared under this package. I declared my function as follows;

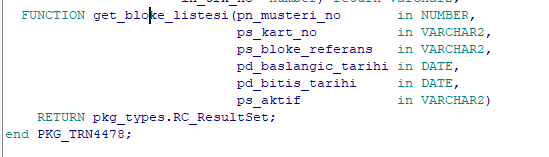


Figure 8: get\_bloke\_listesi Sql Function

Required database query for this project is implemented under this function on Oracle Database. It takes the required paramters and returns a resultset. I implemented package body as following;

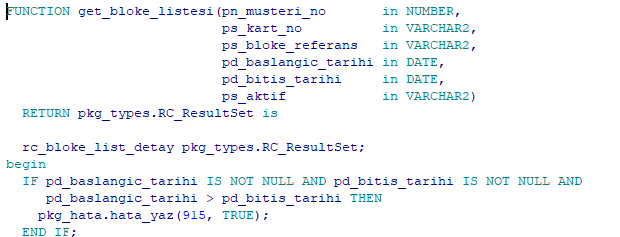


Figure 9: Function Body of get\_bloke\_listesi-1

Our return type is declared as a cursor called rc\_bloke\_list\_detay. Also, I add an if statement to check whether the range of dates are acceptable and not null, otherwise we return an error message.

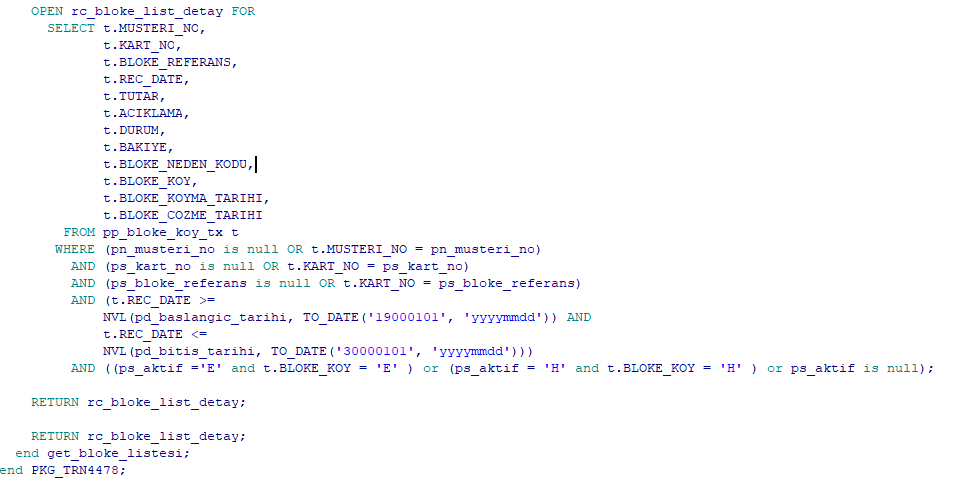


Figure 10: Function Body of get\_bloke\_listesi-2

After declaring our cursor, I added my main select statement which gets the columns from pp\_bloke\_koy\_tx table with same information as function parameters and with required restriction criteria. Then, we return our cursor as a resultset output.

BNSPR\_QRY4479\_BLOKE\_SORGULA service continues after getting a successful result set from our SQL function that we called as following;



Figure 11: Implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA Service-2

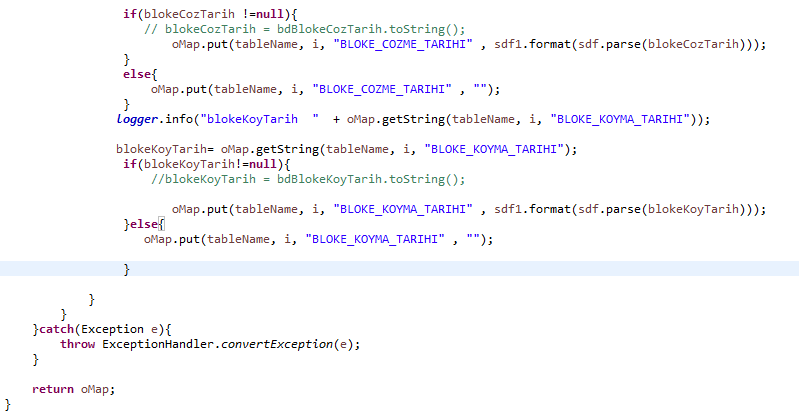


Figure 12: Implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA Service-3

If size of table returned in GMMap is empty, a runtime exception is thrown with the “There is no records.” message. If the “E” value is returned, we put “Bloke Konuldu” to the map, otherwise “Bloke Çözüldü”. Date formats are parsed into the format of expected output requirements. Then, we return the table in a GMMap to the screen as model data to fill the grid. An example output shown as follows with the given customer number.

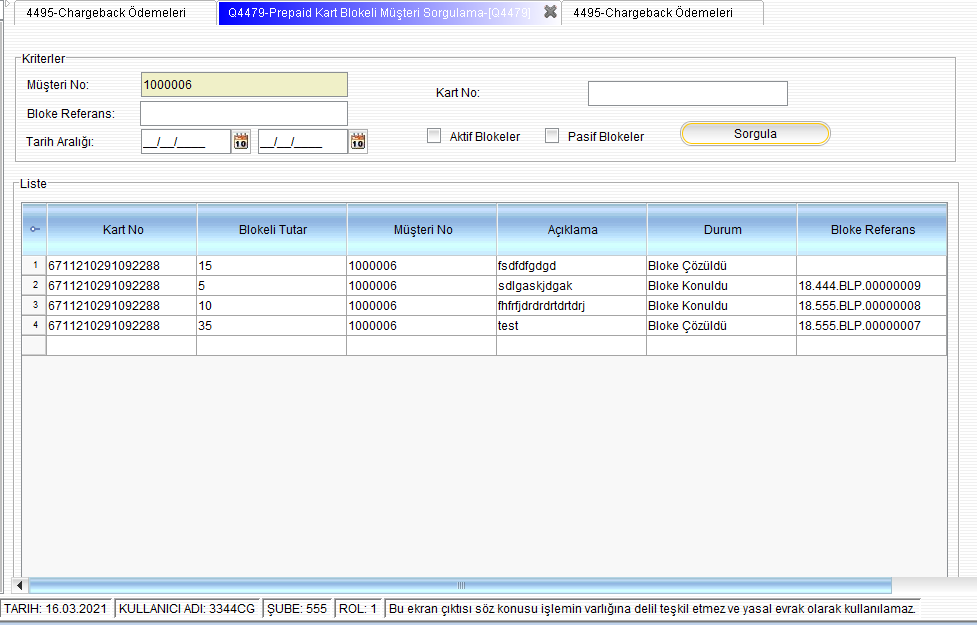


Figure 13: Sample Output of Monitoring Screen

Last thing I did was creating and XML file to show all the customers from pp\_bloke\_koy table as a LOV(List of Values). To do that, a query statement needed to list the customer and then I selected required columns to show on screen.

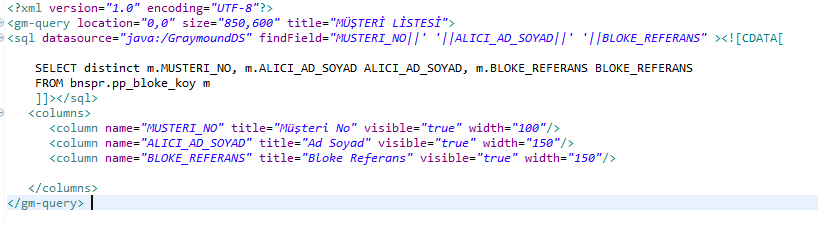


Figure 14: LOV(List of Values) File of Monitorinig Screen

Output screen of this XML file is shown below;

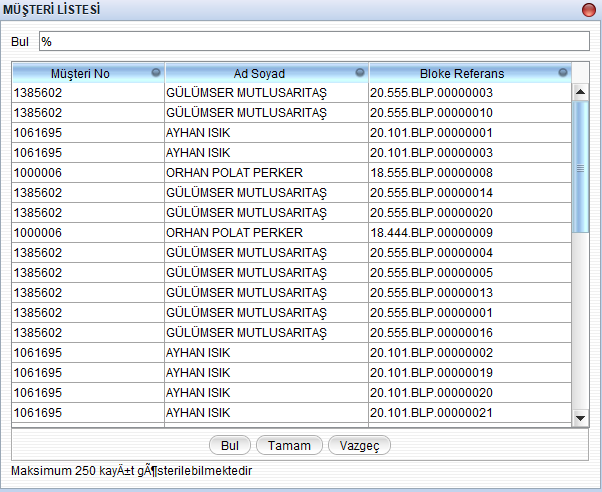


Figure 15: Sample LOV Output Screen

1. **JAVA DESKTOP APPLICATION TO MONITOR MANUEL INQUIRY RESULTS OF LKS ALLOCATION FOR FOREIGN CUSTOMERS**

In this projects, I developed another java desktop application on Eclipse IDE to monitor result of LKS inquiry for foreign customers’ allocation.

The Limit Control System (LKS) is a system where credit card limits can be followed by all banks in accordance with the Bank Cards and Credit Cards Law. Thanks to the system developed within the scope of the Regulation on Bank Cards and Credit Cards issued by the Banking Regulation and Supervision Agency, the total of credit card limits is monitored by the Limit Control system among banks. With LKS, credit card limits collected from all credit card issuing institutions in the sector are combined on a customer basis, enabling the single limit application to be managed by card issuing institutions.

On the monitoring screen, some required information has to be taken as an input and then we have to monitor the return code and return description of inquiry of LKS allocation for given customer. Result of LKS inquiry is returned from LKS system which KKB( Credit Registry Office) provided to banks.

* 1. **Technologies**
* **Java:** Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.
* **Swing Components under Graymound Framework:** Graymound GUIML is and interface that reduce designing time by providing an Eclipse plug-in for SWING. Basically, created designer provides a bug-less, simple, fast, user-friendly interface.
  1. **Project Details**

I was supposed to develop a monitoring screen that takes information of foreign customers and return the proper output of inquiry result coming from LKS system from KKB. Because inputs will be sent to LKS system, very detailed explanation of input and output information are provided below.

Detailed input criteria is defined as follows;

1. TCKN: the screen label should write TKCN / YKN / VKN. String Type. 10 digits or 11 digits load for tckn. PropName: TCKN. Compulsory.
2. Customer No: BigDecimal Type. PropName: MUSTERI\_NO. Compulsory.
3. First Name: String Type, PropName: FIRST\_NAME. Compulsory.
4. Second Name: String Type, PropName: SECOND\_NAME. Not Compulsory
5. Surname: String Type, PropName: SURNAME. Compulsory
6. Date of Birth: Date must be in the format YYYYMMDD, String Type. PropName: DOGUM\_TARIHI. Not Compulsory.
7. Father Name: Name only and String Type. PropName: BABA\_NAME. Not Compulsory.
8. Allocation Date: Must be in the format YYYYMMDD, String Type. PropName: TAHSIS\_TARIHI. Not Compulsory.
9. Current Limit: String Type. PropName: GUNCEL\_LIMIT. Not Compulsory.
10. Branch Code: String Type. PropName: SUBE. Compulsory
11. Manuel Inquiry: String Type. Radio Button: E. PropName: MANUEL\_SORGU. Compulsory.
12. Inquiry Type: 1 digit. String Type. PropName: SORGU\_TIPI. Can be ComboBox. Compulsory. Entry Examples: 01 - Pre-Allocation Inquiry, 2 - Additional Pre-Allocation Inquiry, 3 - LKS Allocation, 4 - LKS Additional Allocation

Detailed return parameters are defined as follows;

1. Return Code: String Type. PropName: RETURN\_CODE.
2. Return Description: String Type. PropName: RETURN\_DESCRIPTION.

Monitoring Screen has to provide features below for Inquiry Type and has to be designed as ComboBox with following additional requirements,

* The allocation date cannot be sent for query types 01 (Pre-Allocation Query) and 02 (Additional Pre-Allocation Query).
* Current Limit Value 01 (Pre-Allocation Query) and 02 (Additional Pre-Allocation Query) cannot be sent different from 0 for query types.
* Allocation date 03 (Online Allocation Operation) and 04 (Online Additional Allocation Operation) is a mandatory field for query types.
  1. **Design of the Project and Explanation of Codes**

Layout design of application is made on Eclipse IDE and with Swing Components of Graymound Framework. As the all other screens on this project, it has a standardized design pattern.

GUIML file of this project that contains visual layout is created under BNSPRCreditCard project.

I implemented my layout for this desktop application as below;

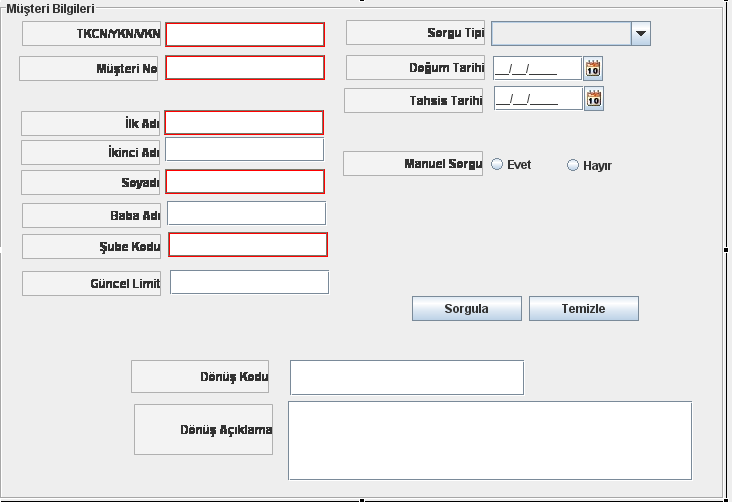


Figure 16: Monitoring Screen for LKS Manuel Inquiry of Foreign Customers

Inquiry Type variable is a ComboBox, so in the actions section, ComboBox is filled with the parameters as described in project details.

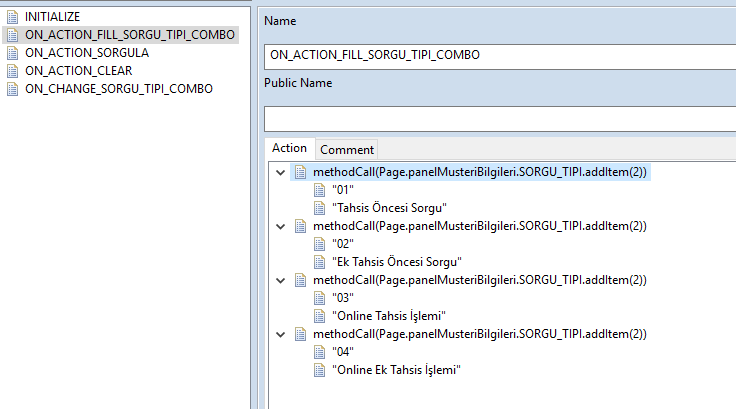


Figure 17: Filling The ComboBox with Required Parameter Values

Most important action on this screen is ON\_CHANGE\_SORGU\_TIPI\_COMBO action that has has to provide features and restrisctions of Inquiry Type variable. For all the conditions,

Following statements are designed to meet requirements as follows;

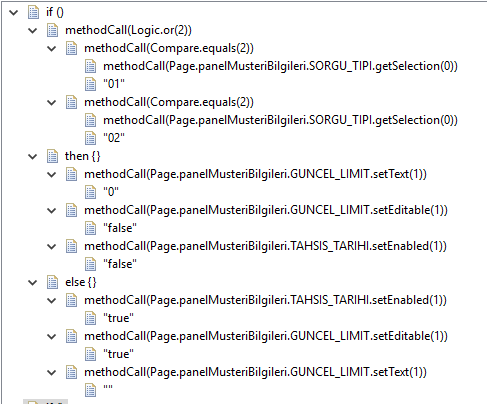


Figure 18: Inquiry Type Variable Conditon Implementations-1

All the conditions that are mentioned in details of Inquiry Type ComboBox are met with following conditon statements;

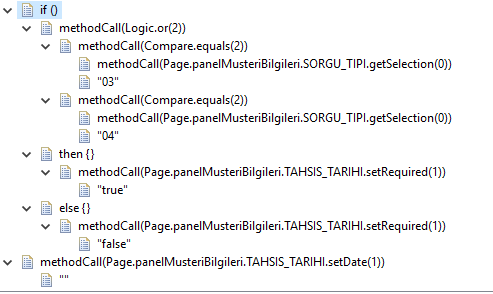


Figure 19: Inquiry Type Variable Conditon Implementations-2

BNSPR\_QRY3893\_LKS\_YKN\_MANUEL\_TAHSIS\_SORGU\_YAP is the service that implemented for the monitoring screen of inquiry result. Firstly, all the values that screen sent to the this service are assigned to variable so that in the following if statements, their validity and compatibility will be checked. If any of compulsory variables is empty or not in compatible data type expected, output to be returned will give spesific return description with error code.

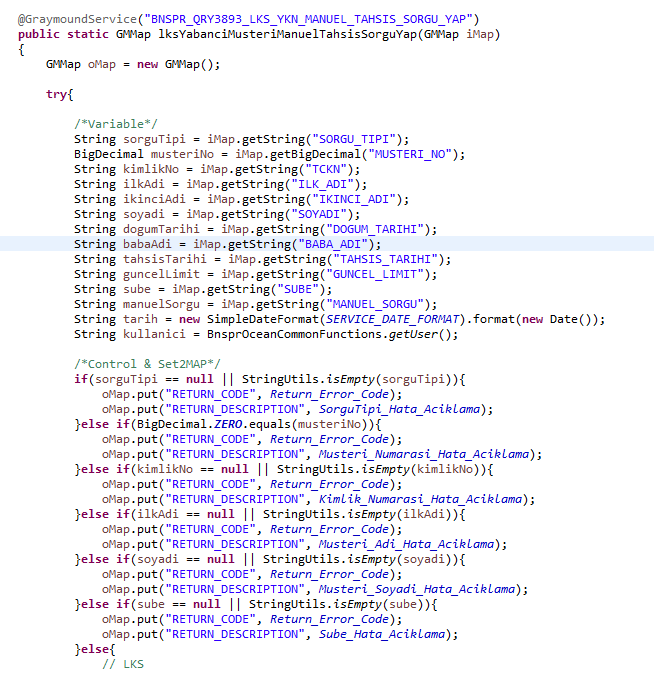


Figure 20: Implementation of BNSPR\_QRY3893\_LKS\_YKN\_MANUEL\_TAHSIS\_SORGU\_YAP Service-1

TCKN data field on the screen can take both TCKN or VKN but we have to decide which one it is and change the value and key of our input map according to this result. If length of String entered is not 11, it means it is not a TCKN and we know it is VKN. So, we make necessary change on our map.



Figure 21: Implementation of BNSPR\_QRY3893\_LKS\_YKN\_MANUEL\_TAHSIS\_SORGU\_YAP Service-2

After validation and compatibilty check of key and values of input map, we call the common service of LKS inquiry via GMServiceExecuter.call function. This function is a Graymound Framework function that is used by all the developers and it takes the service name and a map as a parameter and returns a GMMap of that service’s output.

At the end, we return RETURN\_DESCRIPTION and RETURN\_CODE of LKS inquiry and put these result to our output GMMap that our screen needs for monitoring.

1. **JAVA INTEGRETION SERVICES OF TR KAREKOD QR PROJECT**

In this projects, I coded the integration services of Tr Karekod Qr Project that is required on the side of integration team. Implementing of any service on integration side has a standardized way of coding. All the information of required input values with types and expected output values documented on Swagger as JSON objects. Required end points that are needed also is declared clearly for successful development process.

* 1. **Technologies**
* **Java:** Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.
* **Swagger:** It is an Interface Description Language for describing RESTful APIs expressed using JSON. Swagger is used together with a set of open-source software tools to design, build, document, and use RESTful web services.
  1. **Project Details**

I was supposed to implement four services that are to be called on integration side with following detailed information;

1. Banking Service Name: BNSPR\_ATLAS\_CLOSE\_QR\_TRANSACTION

Method: POST

EndPoint: /api/integration/QrIntegration/CloseQrTransactionBank

Request:

{

"qrReferansNumber": "string"

}

Responses: No response object

Code: 200

Description: Success

1. Banking Service Name: BNSPR\_ATLAS\_GET\_QR\_INFO\_ISSUER

Method: POST

EndPoint: /api/integration/QrIntegration/GetQrInfoIssuer

Request:

{

"qrTerminalType": 0,

"qrData": "string"

}

Responses:

{

"qrFormatInd": "string",

"qrProducerCode": "string",

"qrReferansNumber": "string",

"hashValue": "string",

"crc": "string",

"messageReferenceNumber": "string",

"qrType": "string",

"date": "2021-02-16T10:52:00.546Z",

"countryCode": "string",

"amountAvailable": 0,

"transactionType": 0,

"terminalType": 0,

"supportedSchema": "string",

"supportedBrand": "string",

"installmentNumber": "string",

"rrn": "string",

"amount": 0,

"mcc": "string",

"currencyCode": 0,

"merchantName": "string",

"merchantCity": "string",

"postalCode": "string",

"qrGenerationDate": "2021-02-16T10:52:00.546Z",

"qrExpireDate": "2021-02-16T10:52:00.546Z",

"location": "string",

"merchantId": 0,

"terminalId": "string",

"merchantIban": "string",

"billNumber": "string",

"customerPhone": "string",

"merchantLabel": "string",

"loyaltyId": "string",

"customerId": "string",

"paymentType": "string",

"responseMac": "string"

}

1. Banking Service Name: BNSPR\_ATLAS\_START\_QR\_TRANSACTION\_ISSUER

Method: POST

EndPoint: /api/integration/QrIntegration/StartQrTransactionIssuer

{

"acquirerId": "string",

"channelCode": "string",

"cardNo": "string",

"qrData": "string",

"cvm": 0,

"maskedCardholderName": "string",

"amount": "string",

"pointAmount": "string",

"installmentNumber": "string",

"eci": "string",

"walletId": "string",

"issuerExtra": "string",

"locationData": "string",

"iban": "string",

"processingCode": "string",

"trxType": 0

}

Responses: No response object

Code: 200

Description: Success

1. Banking Service Name: BNSPR\_ATLAS\_START\_TRANSACTION\_ATM

Method: POST

EndPoint: /api/integration/QrIntegration/StartQrTransactionAtm

Request:

{

"cardNo": "string",

"qrData": "string",

"maskedCardholderName": "string",

"amount": "string",

"locationData": "string",

"walletId": "string",

"iban": "string"

}

Responses: No response object

Code: 200

Description: Success

* 1. **Design of the Project and Explanation of Codes**

I developed services that has given details by creating request object, creating response object and implementing the integration service. This is standardized method of development workflow. After development, these services are tested on UAT (User Acceptance Test) environment with example request inputs on Service Tester to check whether services are called successfully and give expected response data.

Because implementing of these four services are very similar, an example of BNSPR\_ATLAS\_GET\_QR\_INFO\_ISSUER service will be given for simplicity of this report.

Request object of this service implemented as follows;

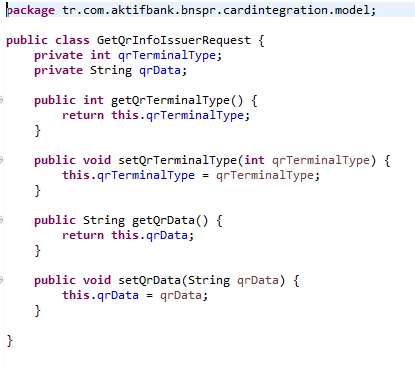


Figure 22: Request POJO of BNSPR\_ATLAS\_GET\_QR\_ISSUER Integration Service

All the request POJOs (Plain Old Java Object) have to include simple getter and setter methods for the variables they contain.

After implementing request object, response object is created same as request POJO with getter and setter methods of variables it contain.

After request and response POJOs, BNSPR\_ATLAS\_GET\_QR\_INFO\_ISSUER service is coded like following;

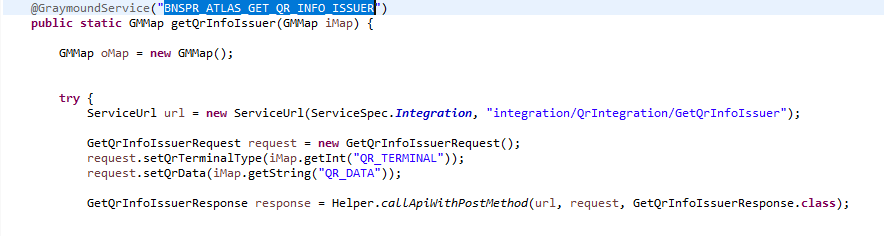


Figure 23: Body of BNSPR\_ATLAS\_GET\_QR\_ISSUER Integration Service-1

Firstly, endpoint as a url is taken from input GMMap and a request object created.

Request object’s fields set with the information taken from input map and after that we make call for a response object wiht our request object and given enpoint by callApiWithPostMethod function which is under Graymound Framework and is stardardized Http Post method used for all projects.

If we successfully get a response from our call, then we put the response information to our output map and we return a success for return.

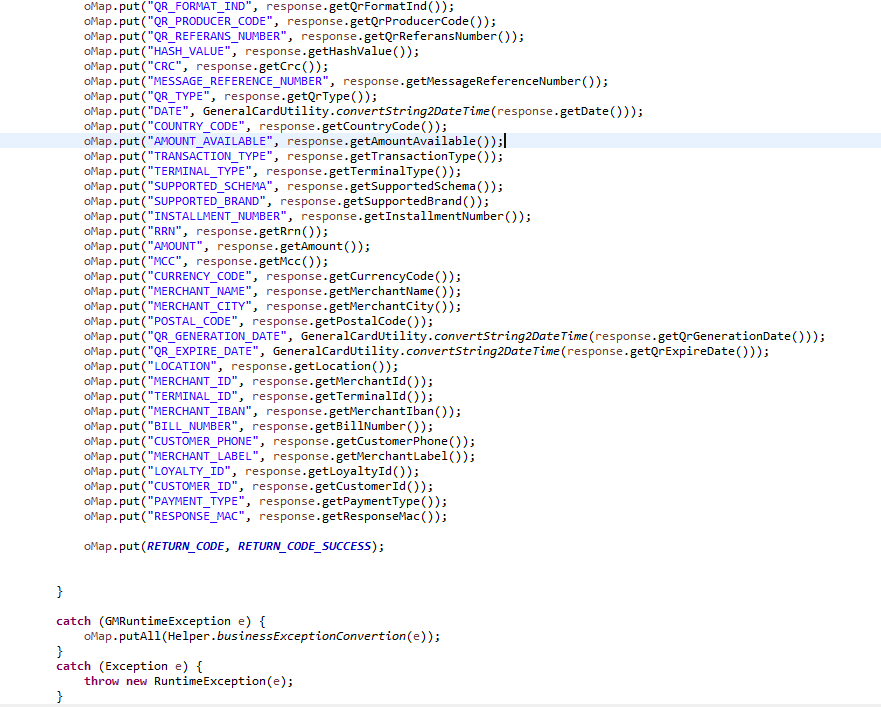


Figure 24: Body of BNSPR\_ATLAS\_GET\_QR\_ISSUER Integration Service-2

1. **IMPRESSIONS ON INTERNSHIP EXPERIENCE**

This internship was my first experience on working in an IT department of a corporation. Firstly, I had no prior knowledge about the methods used or workflow of software production line and it was very brightening experience for me when I had a chance to be part of it. I became a member of a team, which everybody has their own responsibility thay they have to carry out. Being responsible of given tasks, deadlines and experiensing flow of information transfer from one person to other everyday was priceless for me.

I had a chance to work under agile/scrum methodology in a software development team. Everyday, we had meetings that every team member informs others about what they do currently, what is their progress and has change to ask questions to each other. In addition to daily scrums, I attended the weekly meetings that we actualy inform our seniour director about the tasks we work on and our progres also new task assignments were made during this meeting.

Before my internship, I had not developed a project using Java or Swing and did not used Oracle Database systems. This internship helped my programming skills become sharper and learned the technologies used in card payments systems also the business logic of this sector.

After completing my internship, I wanted to continue being part of this friendly and educational environment. Currently, I continue working in Integration Team of Card Payment Systems Department of this company.

1. **CONCLUSION**

In this internship, I had chance to work in an environment in the business world. I met new people who has many experience on programing and software development who helped me a lot during my internship. I learned new technologies that I do not know before and I became a part of Integration Application Development team that I want to become a part of this team. Not only programing knowledge but also I learned business login of Card Payments Systems during development of my projects.

1. **REFERENCES**

* AktifBank Web Page: www.aktifbank.com.tr
* Swagger Web Page: <https://swagger.io>
* Oracle Docs Page: <https://docs.oracle.com/>
* Graymound Web Page: <https://www.graymound.com/graymound2/about>
* KKB Web Page: <https://www.kkb.com.tr/urunler/limit-kontrol-sistem>
* Swingx Web Page: https://swingx.java.net/

1. **APPENDIX**

[Figure 1: Layout Design of Monitoring Screen 7](#_Toc66830683)

[Figure 2: List of Values Shows Customer Information 7](#_Toc66830684)

[Figure 3: Services Used in Monitoring Screen 8](#_Toc66830685)

[Figure 4: List of Actions in Monitoring Screen 8](#_Toc66830686)

[Figure 5: ValidateRequest Class 9](#_Toc66830687)

[Figure 6: Implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA Service-1 9](#_Toc66830688)

[Figure 7: Implementation of callOracleRefCursorFunction Class 10](#_Toc66830689)

[Figure 8: get\_bloke\_listesi Sql Function 10](#_Toc66830690)

[Figure 9: Function Body of get\_bloke\_listesi-1 10](#_Toc66830691)

[Figure 10: Function Body of get\_bloke\_listesi-2 11](#_Toc66830692)

[Figure 11: Implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA Service-2 11](#_Toc66830693)

[Figure 12: Implementation of BNSPR\_QRY4479\_BLOKE\_SORGULA Service-3 12](#_Toc66830694)

[Figure 13: Sample Output of Monitoring Screen 12](#_Toc66830695)

[Figure 14: LOV(List of Values) File of Monitorinig Screen 13](#_Toc66830696)

[Figure 15: Sample LOV Output Screen 13](#_Toc66830697)

[Figure 16: Monitoring Screen for LKS Manuel Inquiry of Foreign Customers 16](#_Toc66830698)

[Figure 17: Filling The ComboBox with Required Parameter Values 16](#_Toc66830699)

[Figure 18: Inquiry Type Variable Conditon Implementations-1 17](#_Toc66830700)

[Figure 19: Inquiry Type Variable Conditon Implementations-2 17](#_Toc66830701)

[Figure 20: Implementation of BNSPR\_QRY3893\_LKS\_YKN\_MANUEL\_TAHSIS\_SORGU\_YAP Service-1 18](#_Toc66830702)

[Figure 21: Implementation of BNSPR\_QRY3893\_LKS\_YKN\_MANUEL\_TAHSIS\_SORGU\_YAP Service-2 19](#_Toc66830703)

[Figure 22: Request POJO of BNSPR\_ATLAS\_GET\_QR\_ISSUER Integration Service 23](#_Toc66830704)

[Figure 23: Body of BNSPR\_ATLAS\_GET\_QR\_ISSUER Integration Service-1 23](#_Toc66830705)

[Figure 24: Body of BNSPR\_ATLAS\_GET\_QR\_ISSUER Integration Service-2 24](#_Toc66830706)