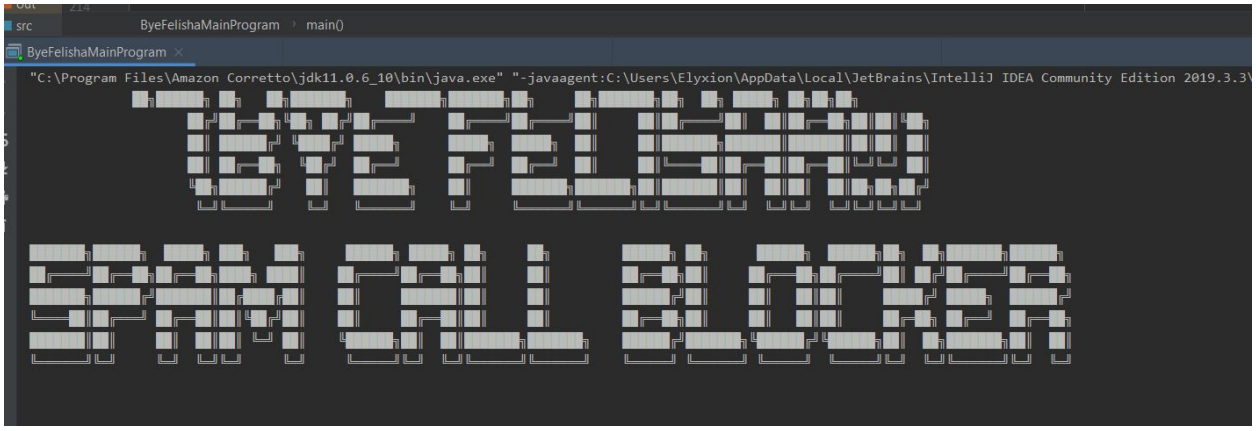
Design Patterns Chosen:

- **Model-View-Controller**
  - **Description::** The Model-View-Controller design pattern implements the important feature of hiding some of the program implementation of the user. The user uses the controller, the controller manipulates the model and the model updates the view. Then the user in the end sees the view.
  - **Reasoning for Choice::** This design pattern works well with our program because we have the background implementation of loading our program that the user should not be able to view or manipulate. The user does view the message notification that of “Loading Program” and “Proceeding to Main Menu”, with an additional function that provides a slight lag, showing that the program will start in a second.
  - **Location in Code::** This design implementation is located in lines 39-52, 128-132, 257-307, in a variety of methods and classes as per the MVC design pattern.

- **Singleton**
  - **Description::** This design pattern ensures only one object of the selected type is created. This pattern is a type of creational design pattern, the patterns that deal with object creation and mechanisms. The creational design patterns tailor the object-creation to the needs of the particular program.
  - **Reasoning for Choice::** This design pattern works well with our program because we have an object we need to ensure there is only one of. The single object we need is the opening logo/ banner at the beginning of the “Bye, Felisha” program.
- **Location in Code::** This design implementation is located in line 217-253, in the class SingleObject.

SINGLETON OUTPUT IMPLEMENTATION:



SINGLETON CODE SNAPSHOT:

MODEL-VIEW-CONTROLLER OUTPUT Implementation

```
Loading Program

Proceeding to Main Menu...
Please enter number corresponding to your choice from the following menu:
1) Register to ByeFelisha Spam Blocker
2) Report Spam Number
3) Display Blocked Calls
4) Check for Scam

Enter 5 to Exit
```

MODEL-VIEW-CONTROLLER CODE SNAPSHOT:

```
/// Model View control implementation

class Model{
    private String UserLicenseAgreement;









    public String GetUserAgreement() {
        return UserLicenseAgreement;
    }
    public void SetUserAgreement(String s) {
         UserLicenseAgreement = s;
    }

}

class Controller{
    private Model model;
    private View view;

    public Controller(Model model , View view) {
        this.model = model;
        this.view = view;

    }

    public void SetUserAgreement(String s) {
        model.SetUserAgreement(s);
```

References:

https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm
https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm

Github Project Link - With Design Patterns Implemented

https://github.com/nn200192/Bye-Felisha-Spam-Blocker/blob/master/FinalDesignPattern3