Uppgiften var att kodeen skulle kunna spara ner meddelanden i olika format i en fil. Dem formaten som skulle finnas med var Rubbery language, Base64 och en super implementation där jag själv fick välja formatet.

För att lösa uppgiften följde jag mönstret decoration pattern. Decorator pattern är ett mönster som tillåter utvecklaren att lägga till nya funktionaliteter till ett objekt. Detta görs genom att objektet läggs i något som kallas för wrapper objects som inehåller dessa funktioner. I andra ord används decorator pattern för att implementera nya funktioner till ett objekt dynamsikt utan att det stör funktionerna av andra objekt i samma klass.

Ideen bakom Open-Closed Principal (OCP) är att man ska kunna behålla koden som redan finns och kunna implementera ny kod utan att det ändrar koden som man tidigare skrivit. En klass kan vara både öppen och stängd på det viset att den samtidigt är öppen för extentions men stängd för ändringar.

Med allt deet sagt har jag lyckats att använda mig av decorator pattern i min lösning men min lösning bryter mot OCP. I min handleCbChange() metod var jag tvungen att modifiera koden genom att bygga in if-satser för att det hela skulle fungera. Jag tänker också att om man i framtiden skulle vilja utöka programmet vidare skulle man fortfarande behöva göra ändringar i handleCbChange() motoden för att lägga till knappar och en till if-sats.

```
@FXML
private void handleCbChange() throws Exception {
    boolean rubberyChecked = cbRubberyLang.isSelected();
    boolean base64Checked = cbBase64.isSelected();
    boolean superImpChecked = cbSuperImp.isSelected();

    System.out.println("Rubbery selected: " + rubberyChecked);
    System.out.println("Base64 selected: " + base64Checked);
    System.out.println("Super implementation selected: " + superImpChecked);

    messageSaver = new SaveMessageToFile();

    if(rubberyChecked){
        this.messageSaver = new RubberLanguage(messageSaver);
    }
    if(base64Checked){
        this.messageSaver = new Base64Lang(messageSaver);
    }
    if(superImpChecked){
        this.messageSaver = new SuperImplementation(messageSaver);
    }
}
```