UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING

SCHOOL OF COMPUTING

SESSION 2020/2021

SEMESTER 2

SECJ2154 OBJECT-ORIENTED PROGRAMMING

SECTION 11

PROJECT REPORT

LECTURER : DR NORSHAM BINTI IDRIS

INVINCIBLE WARRIOR

TITLE: CROSS-STATE SYSTEM

NAME	NO. MATRIC
ASWIND SARVANESH VARMAN A/L SARAVANAN	A19EC0025
NUR KHUZAIMAH BINTI ISKANDAR	A19EC0133
WAN NUR ATIQAH BINTI JUNAIDI	A19EC0176
SHAFIA CARINA SAPTOMO	A19EC0286

Contents

1. Project Description

A. Introduction

For the final project, we had to create program code in Java language with themes in our system related to Covid-19. As we know, the COVID-19 pandemic has affected the world since early 2020 and has brought many new digital needs. There are many sectors affected, including education, tourism, and health. To reduce the risk of COVID-19 infection, the state has issued a travel ban. We decided to develop a program that would help police determine which applicants might apply to cross-government organizations because of the urgent need to monitor cross-government travel and consent.

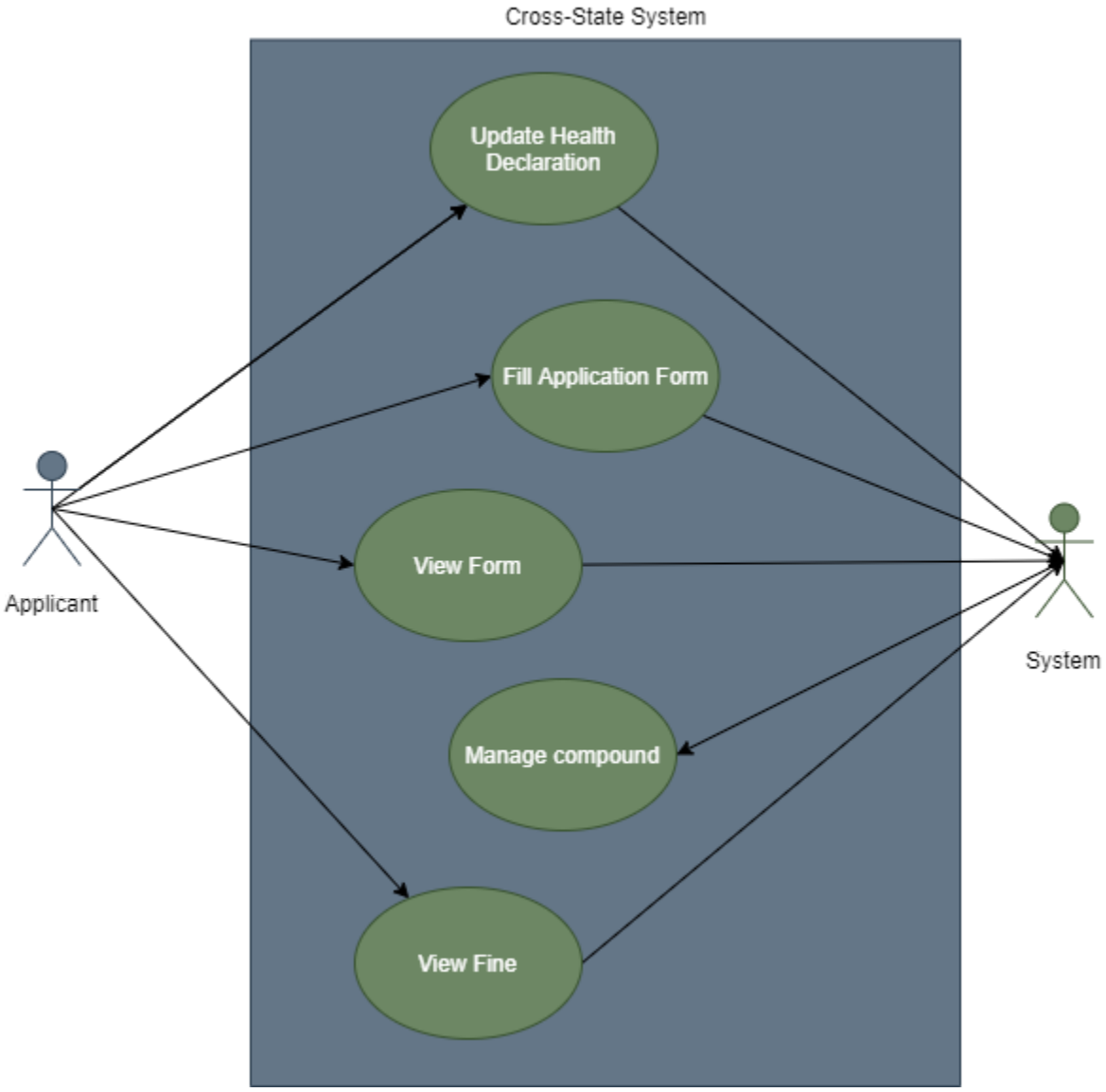
B. Objectives

- To determine the health declaration of the applicant because there will only be low risk and no symptoms.
- To obtain applicant information before requesting cross-border travel.
- To ensure that the application form is completed by the applicant.
- To give a penalty to applicants who have problems with their application.

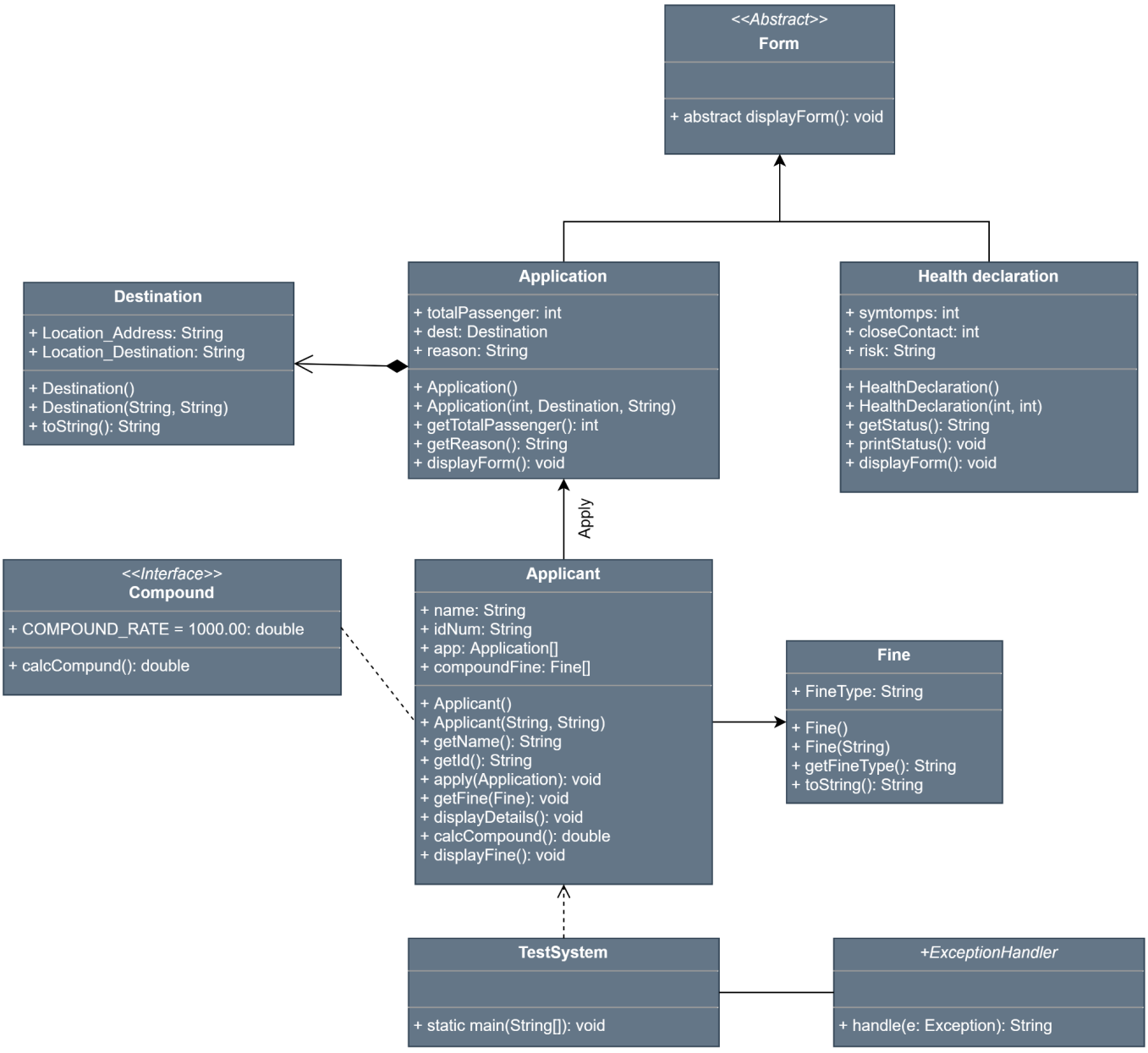
C. Scopes of the System

The scope of the system we want to develop is based on current global health issues, particularly this Corona outbreak. The system will contribute to more efficient data organization and also ensure the complete health of applicants nationwide to reduce the risk of virus spread.

2. Use Case Diagram



3. UML Class Diagram



4. Tasks Allocation

No	Group Members	Task
1	NUR KHUZAIMAH BINTI ISKANDAR	● Allocate task distribution ● Project Documentation ● Complete Use Case Diagram ● Slide Presentation
2	WAN NUR ATIQAH BINTI JUNAIDI	● Coding Implementation ● Brainstorm: Class Diagram & Use Case Diagram
3	ASWIND SARVANESH VARMAN A/L SARAVANAN	● Coding Implementation ● Brainstorm: Class Diagram & Use Case Diagram
4	SHAFIA CARINA SAPTOMO	● Project Documentation ● Complete Class Diagram ● Slide Presentation

5. Coding Implementation & Sample of the Output

a. Coding Implementation

```
// Applicant.java
import java.util.ArrayList;

class Applicant implements Compound{
        private String name;
        private String idNum;
        private ArrayList<Application> app;
        private ArrayList<Fine> compoundFine;

        public Applicant(){}

        public Applicant(String n, String id){
                name = n;
                idNum = id;
                app = new ArrayList<Application>();
                compoundFine = new ArrayList<Fine>();
        }
         public String getName(){
                return name;
        }

        public String getId(){
                return idNum;
        }
        public void apply(Application a){
                app.add(a);
        }
        public void getFine(Fine f){
                compoundFine.add(f);
        }
        public void displayDetails(){
                System.out.println("\nName: " + name);
                System.out.println("I/C: " + idNum);
        }
        public double calcCompound(){
                return(COMPOUND_RATE*((Application)app.get(0)).getTotalPassengers());
        }
        public void displayFine(){
          System.out.println(compoundFine.get(0).toString());
        }
}
```

```
// Application.java
class Application extends Form{
        private int totalPassengers;
        private Destination dest;
        private String reason;

        public Application(){}

        public Application(int totalPassengers, Destination
        dest, String reason)
        {
                this.totalPassengers = totalPassengers;
                this.dest = dest;
                this.reason = reason;
        }
        public int getTotalPassengers(){
                return totalPassengers;
        }
        public String getReason(){
                return reason;
        }
        public void displayForm(){
                System.out.println("\nNumber of passengers: " + totalPassengers);
                System.out.println("Destination details: " + dest.toString());
                System.out.println("Reason for travelling: " + reason);
        }
}
```

```
// Compound.java
public interface Compound
{
        public static final double COMPOUND_RATE = 1000.00;
        public abstract double calcCompound();
}
```

```
// Destination.java
public class Destination
{
        private String Location_Address;
        private String Location_Destination;

        public Destination(){}
        public Destination(String Location_Address, String Location_Destination)
        {
                this.Location_Address = Location_Address;
                this.Location_Destination = Location_Destination;
        }

        public String toString()
        {
                return ("\nHome Location: "+Location_Address+"\nName of Destination: "+
                  Location_Destination);
        }
}
```

```
// Fine.java
class Fine
{
        private String FineType;

        public Fine(){ FineType = "";}
        public Fine(String FineType)
        {
                this.FineType = FineType;
        }
        public String getFineType(){
                return FineType;
        }
        public String toString()
        {
                return "\nFine Reason: " + FineType;
        }
}
```

```
// Form.java
abstract class Form
{
        public abstract void displayForm();
}
```

```
// Healthdeclaration.java
public class HealthDeclaration extends Form
{
        private int symptoms;
        private int closeContact;
        private String risk;

        public HealthDeclaration(){}
        public HealthDeclaration(int symptoms, int closeContact)
        {
                this.symptoms = symptoms;
                this.closeContact = closeContact;
        }
        public String getStatus(){
                if(symptoms == 1 || closeContact == 1){
                        risk = "High Risk";
                }
                if(symptoms == 2 && closeContact == 2){
```

```
                        risk = "Low Risk";
                }
                return risk;
        }
        public void printStatus(int symptoms, int closeContact)
        {
                if(this.symptoms == 1 || this.closeContact == 1){
                        System.out.println("\nYour health status is: High Risk" );
                }
                if(this.symptoms == 2 && this.closeContact == 2){
                        System.out.println("\nYour health status is: Low Risk" );
                }

        }
        public void displayForm(){
                System.out.println("\nYour health status: " + getStatus());
        }
}
```

```
// TestSystem.java
import java.util.*;
import java.io.*;

public class TestSystem
{
        public static void displayMenu(){
                        System.out.println("-----------------------------------------");
                        System.out.println(" Covid-19 Highway Application System");
                        System.out.println("-----------------------------------------");
                        System.out.println("Which option do you want to choose?");
                        System.out.println("[1] Fill in health declaration form");
                        System.out.println("[2] Fill in application form");
                        System.out.println("[3] View application");
                        System.out.println("[4] View Fine");
                        System.out.println("[5] Exit to main menu");
                        System.out.print("Your choice: ");
        }

         public static void main(String args[]){

                Scanner input = new Scanner(System.in);
                int choice;
                ArrayList<Applicant> aList = new ArrayList<Applicant>();
                Applicant a = new Applicant();
                HealthDeclaration health = new HealthDeclaration();
                Destination d = new Destination();
                Application app = new Application();

                ArrayList<Fine> compound = new ArrayList<Fine>();
                Fine fine1 = new Fine("More than 4 passengers");

                System.out.println("\nPlease enter your name and identification number.");
                System.out.print("Name: ");
                String nameApplicant = input.next();

                System.out.print("Identification number: ");
                String idApplicant = input.next();

                a = new Applicant(nameApplicant, idApplicant);

                do{
                        displayMenu();
                        choice = input.nextInt();

                        switch(choice){
                                case 1:
                                        System.out.print("Do you have any symptoms? ( 1 -
                                        YES/ 2 - NO ): ");
                                        int symptoms = input.nextInt();

                                        System.out.print("Do you encountered any close
                                        contact with Covid-19 patients?  ( 1 - YES/ 2 -
                                        NO ): ");
                                        int close = input.nextInt();

                                        health = new HealthDeclaration(symptoms,close);
                                        health.printStatus(symptoms, close);
```

```
                                        break;

                                case 2 :
                                        if(health.getStatus() == "Low Risk"){
                                           System.out.print("\nNumber of passengers
                                            travelling: ");
                                           int numPassengers = input.nextInt();

                                           System.out.print("Your Home location: ");
                                           String home = input.next();

                                           System.out.print("Your destination: ");
                                           String destination = input.next();

                                           System.out.print("Reason for travelling: ");
                                           String reason = input.next();

                                           d = new Destination(home, destination);
                                           app = new Application(numPassengers,  d,
                                             reason);
                                           a.apply(app);
                                        }

                                        if(health.getStatus() == "High Risk"){
                                           System.out.println("\nCannot Apply Form!");
                                        }
                                        break;

                                case 3 :
                                        if(health.getStatus() == "Low Risk"){
                                        a.displayDetails();
                                          health.displayForm();
                                          app.displayForm();
                                        }
                                        if(health.getStatus() == "High Risk"){
                                           System.out.println("\nNo form can be viewed!");
                                        }
                                         break;

                                case 4:
                                         if(app.getTotalPassengers() > 4){
                                           a.getFine(fine1);
                         System.out.println("\nFine: " +
                                           a.calcCompound());
                                           a.displayFine();
                                         }
                                         else{
                                           System.out.println("\nNo fine charged!");
                                         }
                                         break;
                        }
                        System.out.println("\nReturning to main menu...");
                        try {
                          Thread.sleep(2500);
                        } catch (Exception e) {}
                }while(choice != 5);
                System.out.println("\nThank you for using this system :)");
        }
}
```

b. Sample of the Output

```
// Output 1
Please enter your name and identification number.
Name: dave
Identification number: 1234567
----------------------------------------
 Covid-19 Highway Application System
----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 1
Do you have any symptoms? ( 1 - YES/ 2 - NO ): 2
Do you encountered any close contact with Covid-19 patients?  ( 1 - YES/ 2 - NO ): 2

Your health status is: Low Risk

Returning to main menu...
----------------------------------------
 Covid-19 Highway Application System
----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 2

Number of passengers travelling: 1
Your Home location: jb
Your destination: kl
Reason for travelling: family

Returning to main menu...
----------------------------------------
 Covid-19 Highway Application System
----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 3

Name: dave
I/C: 1234567

Your health status: Low Risk

Number of passengers: 1
Destination details:
Home Location: jb
Name of Destination: kl
Reason for travelling: family

Returning to main menu...
----------------------------------------
 Covid-19 Highway Application System
----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
```

```
[4] View Fine
[5] Exit to main menu
Your choice: 4

No fine charged!

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 5
Returning to main menu...

Thank you for using this system :)
Press any key to continue . . .
```

```
// Output 2
Please enter your name and identification number.
Name: dave
Identification number: 1234567
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 1
Do you have any symptoms? ( 1 - YES/ 2 - NO ): 1
Do you encountered any close contact with Covid-19 patients?  ( 1 - YES/ 2 - NO ): 1

Your health status is: High Risk

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 2

Cannot Apply Form!

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 3

No form can be viewed!

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
```

```
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 4

No fine charged!

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 5
Returning to main menu...

Thank you for using this system :)
Press any key to continue . . .
```

```
// Output 3
Please enter your name and identification number.
Name: dave
Identification number: 1234567
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 1
Do you have any symptoms? ( 1 - YES/ 2 - NO ): 2
Do you encountered any close contact with Covid-19 patients?  ( 1 - YES/ 2 - NO ): 2

Your health status is: Low Risk

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 2

Number of passengers travelling: 5
Your Home location: jb
Your destination: kl
Reason for travelling: family

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 3

Name: dave
I/C: 1234567

Your health status: Low Risk
```

```
Number of passengers: 5
Destination details:
Home Location: jb
Name of Destination: kl
Reason for travelling: family

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 4

Fine: 5000.0

Fine Reason: More than 4 passengers

Returning to main menu...
-----------------------------------------
 Covid-19 Highway Application System
-----------------------------------------
Which option do you want to choose?
[1] Fill in health declaration form
[2] Fill in application form
[3] View application
[4] View Fine
[5] Exit to main menu
Your choice: 5
Returning to main menu...

Thank you for using this system :)
Press any key to continue . . .
```

6. Object-Oriented Concepts applied

Object-Oriented Programming (OOP) is a programming paradigm based on abstraction, encapsulation, inheritance, and polymorphism. It lets users design their own objects and ways to deal with them. The fundamental idea behind OOPs is to create objects, reuse them throughout the program, and modify them to achieve desired outcomes. This project we implement several java concepts, as follows:

a. Encapsulation

Encapsulation is a key concept in working with objects. Combining data and behavior in one package and hiding the implementation of the data from the user of the object.

- In this project all data is set to *private*, so that other classes cannot access it.
- The data will be executed on the main method in the class, we can not call the variable directly, but we can access the variable/data through those methods getter/setter.

```
class Fine
{
	// Variable Private
	private String FineType;

	//Default constructor
	public Fine(){}

	//Constructor with argument
	public Fine(String FineType)
	{
		this.FineType = FineType;
	}

	// Method Getter (Public)
	public String getFineType()
	{
		return FineType;
	}
	 public String toString()
	{
		return "\nFine Reason: " + FineType;
	}
}
```

b. Association

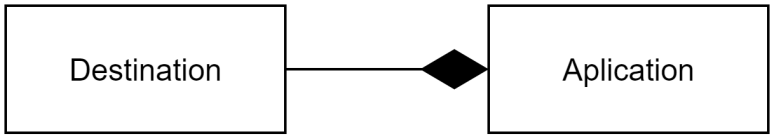
An association represents a general binary relationship that describes an activity between two classes with multiplicity. The relationship also allows objects to call methods in other objects.


```
class Applicant implements Compound{
      private String name;
      private String idNum;
      private ArrayList<Application> app;   //Association
      private ArrayList<Fine> compoundFine; //Association

      public Applicant(){}

      public Applicant(String n, String id){
            name = n;
            idNum = id;
            app = new ArrayList<Application>();
            compoundFine = new ArrayList<Fine>();
      }
       public String getName(){
            return name;
      }

      public String getId(){
            return idNum;
      }
      // Apply association
      public void apply(Application a){
            app.add(a);
      }
      // Apply association
      public void getFine(Fine f){
            compoundFine.add(f);
      }
      public void displayDetails(){
            System.out.println("\nName: " + name);
            System.out.println("I/C: " + idNum);
      }
      public double calcCompound(){
            return(COMPOUND_RATE*((Application)app.get(0))
              .getTotalPassengers());
      }
      public void displayFine(){
             System.out.println(compoundFine.get(0).
              toString());
      }
}
```

c. Aggregation/Composition

Aggregation is a special form of association, which represents an ownership relationship between two classes. Aggregation models the has-a relationship. If an object is exclusively owned by an aggregated object, the relationship between the object and its aggregated object is referred to as composition.

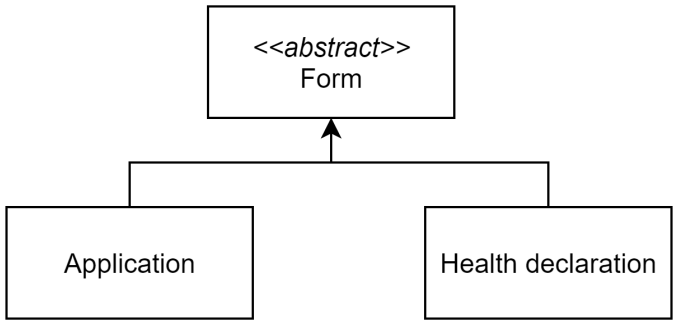
- In this project implement composition relationships.




```
class Application extends Form{
      private int totalPassengers;
      private Destination dest;       // Composition
      private String reason;

      public Application(){}

      public Application(int totalPassengers, Destination
          dest, String reason)
      {
            this.totalPassengers = totalPassengers;
            this.dest = dest;
            this.reason = reason;
      }
      public int getTotalPassengers(){
            return totalPassengers;
      }
      public String getReason(){
            return reason;
      }
      public void displayForm(){
            System.out.println("\nNumber of passengers: " +
              totalPassengers);
            System.out.println("Destination details: " +
              dest.toString());
            System.out.println("Reason for travelling: " +
              reason);
      }
}
```

d. Inheritance

Inheritance is the ability of one class to extend the capabilities of another. It allows code defined in one class to be reused in other classes. It's creating a parent-child relationship between two classes.



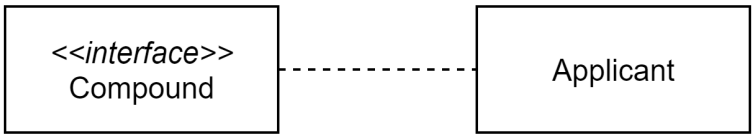

```
// Inheritance for HealthDeclaration class and
Application
abstract class Form{
      public abstract void displayForm();
}

public class HealthDeclaration extends Form
{
      private int symptoms;
      private int closeContact;
      private String risk;

      public HealthDeclaration(){}
      public HealthDeclaration(int symptoms, int
            closeContact)
      {
            this.symptoms = symptoms;
            this.closeContact = closeContact;
      }
      public String getStatus(){
            if(symptoms == 1 || closeContact == 1){
                  risk = "High Risk";
            }
            if(symptoms == 2 && closeContact == 2){
                  risk = "Low Risk";
            }
            return risk;
      }
      public void printStatus(int symptoms, int
            closeContact)
      {
            if(this.symptoms == 1 || this.closeContact ==
                1){
                  System.out.println("\nYour health status
```

```
                    is: High Risk" );
            }
            if(this.symptoms == 2 && this.closeContact ==
                2){
                 System.out.println("\nYour health status
                    is: Low Risk" );
            }
      }
      // Apply inheritance
      public void displayForm(){
            System.out.println("\nYour health status: " +
              getStatus());
      }
}

class Application extends Form{
      private int totalPassengers;
      private Destination dest;       // Composition
      private String reason;

      public Application(){}

      public Application(int totalPassengers, Destination
         dest, String reason)
      {
            this.totalPassengers = totalPassengers;
            this.dest = dest;
            this.reason = reason;
      }
      public int getTotalPassengers(){
            return totalPassengers;
      }
      public String getReason(){
            return reason;
      }
      // Apply inheritance
      public void displayForm(){
            System.out.println("\nNumber of passengers: " +
              totalPassengers);
            System.out.println("Destination details: " +
              dest.toString());
            System.out.println("Reason for travelling: " +
              reason);
      }
}
```

e. Interface & Implements

An interface is a classlike construct that contains only constant variables and abstract methods definition, cannot contain instance variables and concrete

methods, and specifies what methods will be implemented by classes that implement an interface.

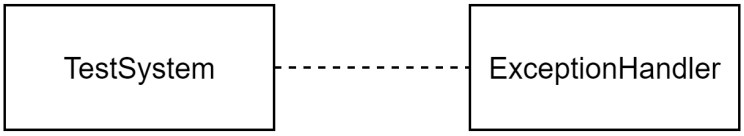



```
// Interface
public interface Compound

{
      public static final double COMPOUND_RATE = 1000.00;
      public abstract double calcCompound();
}

// Apply interface
class Applicant implements Compound{
      private String name;
      private String idNum;
      private ArrayList<Application> app;   // Association
      private ArrayList <Fine> compoundFine;     //
Association

      public Applicant(){}

      public Applicant(String n, String id)
      {
            name = n;
            idNum = id;
            app = new ArrayList<Application>();
            compoundFine = new ArrayList<Fine>();
      }
       public String getName(){
            return name;
      }

      public String getId(){
            return idNum;
      }
      public void apply(Application a){
            app.add(a);
      }
      public void getFine(Fine f){
            compoundFine.add(f);
      }
      public void displayDetails(){
            System.out.println("\nName: " + name);
            System.out.println("I/C: " + idNum);
```

```
      }
      // Apply interface
      public double calcCompound(){
            return
(COMPOUND_RATE*((Application)app.get(0)).
              getTotalPassengers());
      }
      public void displayFine(){
             System.out.println(compoundFine.get(0).
               toString());
      }
}
```

f. Exception handling

Exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. The program that does not provide code for catching and handling exceptions will terminate abnormally, and may cause serious problems. The code that handles the exception is called the exception handler. The exception handling implemented in this project such as:




```
import java.util.*;
import java.io.*;

public class TestSystem
{
        public static void displayMenu(){
                        System.out.println("----------------------------------------");
                        System.out.println(" Covid-19 Highway Application System");
                        System.out.println("----------------------------------------");
                        System.out.println("Which option do you want to choose?");
                        System.out.println("[1] Fill in health declaration form");
                        System.out.println("[2] Fill in application form");
                        System.out.println("[3] View application");
                        System.out.println("[4] View Fine");
                        System.out.println("[5] Exit to main menu");
                        System.out.print("Your choice: ");
        }

         public static void main(String args[]){

                Scanner input = new Scanner(System.in);
                int choice;
                ArrayList<Applicant> aList = new ArrayList<Applicant>();
                Applicant a = new Applicant();
                HealthDeclaration health = new HealthDeclaration();
                Destination d = new Destination();
                Application app = new Application();

                ArrayList<Fine> compound = new ArrayList<Fine>();
                Fine fine1 = new Fine("More than 4 passengers");

                System.out.println("\nPlease enter your name and identification number.");
                System.out.print("Name: ");
                String nameApplicant = input.next();

                System.out.print("Identification number: ");
                String idApplicant = input.next();
```

```
            a = new Applicant(nameApplicant, idApplicant);

            do{
                    displayMenu();
                    choice = input.nextInt();

                    switch(choice){
                            case 1:
                                    System.out.print("Do you have any symptoms? ( 1 -
                                    YES/ 2 - NO ): ");
                                    int symptoms = input.nextInt();

                                    System.out.print("Do you encountered any close
                                    contact with Covid-19 patients?  ( 1 - YES/ 2 -
                                    NO ): ");
                                    int close = input.nextInt();

                                    health = new HealthDeclaration(symptoms,close);

                                    health.printStatus(symptoms, close);
                                    break;

                            case 2 :
                                    if(health.getStatus() == "Low Risk"){
                                       System.out.print("\nNumber of passengers
                                        travelling: ");
                                       int numPassengers = input.nextInt();

                                       System.out.print("Your Home location: ");
                                       String home = input.next();

                                       System.out.print("Your destination: ");
                                       String destination = input.next();

                                       System.out.print("Reason for travelling: ");
                                       String reason = input.next();

                                       d = new Destination(home, destination);
                                       app = new Application(numPassengers,  d,
                                         reason);
                                       a.apply(app);
                                    }

                                    if(health.getStatus() == "High Risk"){
                                       System.out.println("\nCannot Apply Form!");
                                    }
                                    break;

                            case 3 :
                                   if(health.getStatus() == "Low Risk"){
                                    a.displayDetails();
                                      health.displayForm();
                                      app.displayForm();
                                    }
                                    if(health.getStatus() == "High Risk"){
                                       System.out.println("\nNo form can be viewed!");
                                    }
                                     break;

                            case 4:
                                     if(app.getTotalPassengers() > 4){
                                       a.getFine(fine1);
                                      System.out.println("\nFine: " +
                                      a.calcCompound());
                                       a.displayFine();
                                    }
                                    else{
                                      System.out.println("\nNo fine charged!");
                                    }
                                    break;
                    }
                    System.out.println("\nReturning to main menu...");
                   // Apply exception handling
                    try {
                      Thread.sleep(2500);
                    } catch (Exception e) {}
            }while(choice != 5);
            System.out.println("\nThank you for using this system :)");
    }
```

```
}
```

7. Conclusion

For this project, we successfully completed a system aimed at applicants who want to leave the city / cross-government travel. This system applies the Java concepts that we have learned, including encapsulation where we apply all data sets to the private, association, aggregation/composition, inheritance, interface & implementation, and last exception handling. The system has also fulfilled its objectives according to our objectives, this application can determine the applicant's health declaration as there will only be low risk and no symptoms, obtain the applicant's information before requesting cross-border travel, ensure that the application form is completed by the applicant, and give fine to applicants who have problem with their application.

8. Reference

Rungta, K. (2021, April 22). *What is, Basics with Examples*. OOPs Concepts in Java. Retrieved June 16, 2021, from https://www.guru99.com/java-oops-concept.html