Java Programming  
  
Report #1: Object-oriented programming  
Your project name

**Class : 19CLC-KTPM2**

|  |  |
| --- | --- |
| **Your group**: | **Full name 1 – Student ID 1**  **Full name 2 – Student ID 2**  **Full name 3 – Student ID 3** |

Table of content

[Revision History 3](#_Toc54770434)

[Introduction 4](#_Toc54770435)

[Analysis and design 5](#_Toc54770436)

[Implementation 6](#_Toc54770437)

[Result 7](#_Toc54770438)

[Plan 8](#_Toc54770439)

[References 9](#_Toc54770440)

# Revision History

[*Provide in this section a revision history table. A such sample table is given below*]

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

*[You present here:*

* *The context of the project*
* *Give the reason why you choose this topic*
* *Review existing applications / resources, which functionnalities did these systems provide*
* *Propose the requirements (including functionnal and non-functional requirements) of your proposed system, clearly explain the differences/improvements between yours and the existing application.*
* *Also define the scope of your project*
* *Also present the expected outcome of this project*

*]*

# Analysis and design

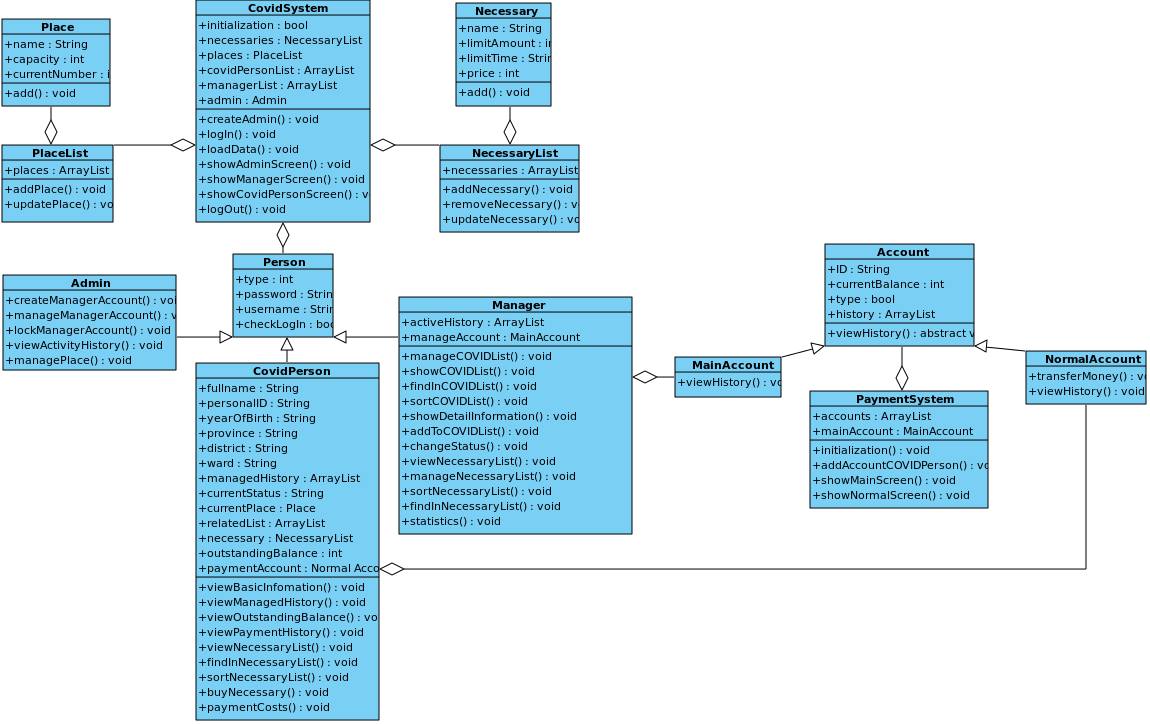
*[ Provide the class diagram to show the organization of your code to be implemented. (If possible, present a general diagram to better show the class hierarchy and then the detail of each class (with main attributes and operations). You are encouraged to draw UML class diagram with Visual Paradigm).*

*Give the package diagram to show the decomposition of your code into packages (if any). Also give a brief description for each package.*

*Give the explanation to describe each figure or each class and the reason for your program's structural design.*

*Present and give explanation for all the design patterns, algorithms you use in the project.*

*]*



* **Person**: This is an abstract class stores basic attributes of an account of COVID System such as type, username, password, checkLogin.
* **Admin**: This is a child class from class Person. This class represents the Admin module and performs some admin-specific operations.
* **Manager**: This is a child class from class Person which has some own attributes such as activeHisory for storing activity history and manageAccount for manage main account of Payment System.This class represents the Manager module and performs some manager-specific operations.
* **COVIDPerson**: This is a child class from class Person which has many own attributes such as fullName, personalID, yearOfBirth, province, dictrict, ward, managedHistory, relatedList, currentPlace and currentStatus for basic information of a COVID person, necessary for list of bought necessary, outstandingBalance for outstanding balance and paymentAccount for own payment account of each COVID peson.This class represents the COVIDPerson module and performs some COVIDPerson-specific operations.
* **Place**: This class represents treatment place, stores basic attributes of a treatment place such as name, capacity, currentNumber and performs some management operations like add,….
* **PlaceList**: This is a class used for storing list of treatment place and performing management operations on the list of treatment place such as add a place to the list, update a place in list.
* **Necessary**: This class represents necessary for Covid person, stores basic attributes of a necessary such as name, limitAmount, limitTime, price and performs some management operations like add,….
* **NecessaryList**: This is a class used for storing list of necessary and performing management operations on the list of necessary such as add a necessary to the list, update a necessary in list, remove a necessary in list.
* **CovidSystem**: This is a class which manages the whole Covid System. This class used for storing all the necessary data of COVID System , performing authentication operations, showing menu and performing specific operations for each type of account.
* **Account**: This is an abstract class stores basic attributes of an account of Payment System such as ID, currentBalance, type, history. This class also has an abstract operations viewHistory for its childs.
* **MainAccount**: This is a child class from class Account. This class represents the main account of Payment System to receive payment from normal account and performs some own operations.
* **NormalAccount**: This is a child class from class Account. This class represents the normal account of Payment System. This account is created when a Covid person added to Covid System. This class also has some own operations.
* **PaymentSystem**: This is a class which manages the whole Payment System. This class stores list of normal payment account and a main payment account. This class also show menu and performs specific operations for each type of payment account.

# Implementation

*[ You present here the way you transfer design models mentioned above into Java code]*

# Result

*[Explain what you have achieved until now, advantages, disadvantages and planned solutions (if possible)]*

# Plan

*[Give your project plan (in detail) until the end of the project: task decomposition, ressources allocation, duration of each task, etc.]*

1. **Task decomposition**:

Diagram

Description automatically generated

1. **Process Management**
   1. Project Estimates

The estimated time for the whole project is 8 weeks, following the detailed plan in part 2.2 below. This estimate is based on the fortnightly tasks for each phase of the project. As this is a voluntary program, the user satisfaction would be the compensation for our development team. Re-estimation would only occur in the event of feedback received once the program published.

* 1. Project Plan

Sprint 1:

* Duration: 14 days (01/11/2021 – 14/11/2021)
* Outcome: Class diagram, Package diagram, Report#1
* Task information:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | Member | Type of Task | Duration | Start date | End date |
| Report - Introduction | Tùng | Document | 6 | 01/11 | 07/11 |
| Report - Plan | Khuê | Document | 6 | 01/11 | 07/11 |
| Class diagram - Basic | Trung | Document | 6 | 01/11 | 07/11 |
| Database and Network research | All | Training | 6 | 01/11 | 07/11 |
| Class diagram | All | Document | 2 | 07/11 | 09/11 |
| Package diagram | All | Document | 2 | 09/11 | 11/11 |
| Report - Analysis and Design | Khuê, Trung | Document | 3 | 11/11 | 14/11 |
| Report - Result | Tùng | Document | 3 | 11/11 | 14/11 |
| Report redesignation | Tùng | Document | 1 | 13/11 | 14/11 |

* Schedule:

Chart, bar chart

Description automatically generated

Sprint 2:

* Duration: 14 days (15/11/2021 – 28/11/2021)
* Outcome: Patient System, Report#2
* Task information:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | Member | Type of Task | Duration | Start date | End date |
| Implementation - Patient System - Admin side | Trung | Coding | 13 | 15/11 | 28/11 |
| Implementation - Patient System - Manager side | Khuê, Trung | Coding | 13 | 15/11 | 28/11 |
| Implementation - Patient System - Patient side | Tùng | Coding | 13 | 15/11 | 28/11 |
| Report update and redesignation | Tùng | Document | 3 | 25/11 | 28/11 |

* Schedule:

Chart

Description automatically generated

Sprint 3:

* Duration: 14 days (29/11/2021 – 12/12/2021)
* Outcome: Payment System, Report#3
* Task information:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | Member | Type of Task | Duration | Start date | End date |
| Implementation - Payment System - Manager side | Trung | Coding | 13 | 29/11 | 12/12 |
| Implementation - Payment System - Patient side | Tùng | Coding | 13 | 29/11 | 12/12 |
| System Synchronization | Khuê | Coding | 13 | 29/11 | 12/12 |
| Report update and redesignation | Tùng | Document | 3 | 09/12 | 12/12 |

* Schedule:

Chart, bar chart

Description automatically generated

Sprint 4:

* Duration: 14 days (13/12/2021 – 26/12/2021)
* Outcome: Final System, Report#4
* Task information:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task | Member | Type of Task | Duration | Start date | End date |
| Implementation - Data Transfer via Network | All | Coding | 6 | 13/12 | 19/12 |
| Component Testing | All | Coding | 2 | 19/12 | 21/12 |
| System Testing | All | Coding | 3 | 21/12 | 24/12 |
| Report update and redesignation | Tùng | Document | 3 | 23/12 | 26/12 |
| Meeting preparation | All | Document | 2 | 24/12 | 26/12 |

* Schedule:

Chart

Description automatically generated

1. **Risk Management**

Risks will be identified in the Inception Phase using the steps identified in the RUP for Small Projects activity “Identify and Assess Risks”. Project risks are evaluated at least once per iteration and documented in this table. The risks of the greatest magnitude are listed first in the table.

|  |  |  |
| --- | --- | --- |
| Risk magnitude | Risk description | Mitigation strategy |
| High | To complete the stakeholder’s requirements, the team will deal with 3 aspects of programming: Java, Database and Networking. Applying all of those to the app might have trouble and easily lead to poor-quality products. | Each member have to do some study and research in all of those fields. |
| Medium | The team consists of 3 members instead of 4 as the maximum number. This will be a disavantage for our team to compete with other teams. Overdue and poor-quality products might be the result of this point. | The whole project must be carefully seperated into phases to ensure balanced workloads. Each member is required to read each phase’s plan and self-managed time budget to complete his/her tasks on time. |
| Low | There might be a little gap in skill level among team’s members. Misunderstandings may appear and slow down the team’s process | Each member needs to keep a cooperative and supportive attitude towards teammates. |
| Low | A team member suddenly quits the project may lead to an unexpected result to the whole project. | All members must understand the whole project to be albe to handle quited member’s work. |

# References

*[Provide all the resources to use in your project, including existing codes, algorithms used, books, reports, links to consult, etc. ]*