Project Document

Meeting Minutes:

First:

Minutes of the first meeting

Date: 11th March

Time:21:30

Venue: Zoom

Attending: Keith, Steven, Anvia

Recorder: Keith

Agenda:

1. Setting up the team repository
2. Team formation form
3. Task allocation
4. UML diagram and attributes tables

Meeting Content:

1.Setting up the team repository

- 1.1 The team repository was set by us together in the meeting as according of the specification.
- 1.2 Keith invited the TAs to join the repository.
- 1.3 Anvia and Steven created the branches.

2 Problem for accessing the branches by Anvia and Steven

- 2.1 Anvia and Steven could not access the branches after creating the branches.
- 2.2 The problem is solved after Keith change the setting of the repository.

3 Team formation form

- 3.1 Keith will become the team leader.
- 3.2 The allocated task:

Group member	The task allocated
Anvia	Function A
Steven	Function B
Keith	Function C

- 3.3 The team formation form was completed in the meeting and added in the readme.md file

4 Problem of adding the form into the readme.md file

- 4.1 The team formation form can not be added directly in form of excel file.
- 4.2 The problem is solved by writing the form by html code.

5 The UML diagram and attributes table

- 5.1 The diagram and attributes table will be done in the next meeting.
- 5.2 Everyone need to read the specification and think about it before the next meeting.

6 The meeting adjournment and the next meeting

- 6.1 The meeting was adjourned at 23:00 and the next meeting will be held at the 14th March after the end of the COMP3111 lecture.
- 6.2 Steven booked the room 353 in the library for the next meeting.

Second:

Minutes of the second meeting

Date: 18th March

Time: 13:30

Venue: Zoom

Attending: Keith, Steven, Anvia

Recorder: Keith

Agenda:

1. Attributes table
2. UML diagram

Meeting content:

1 Attributes table
   - 1.1 Everyone spent 30 minutes to extract the attributes of the object from the specification and make the attributes table by themselves.
   - 1.2 Every groupmates shared their own attributes tables and had a discussion on them.
   - 1.3 Most of attributes of objects in the attributes table were filled excepted the attributes of the shipment and the staff
2 The problem on the definition of the shipment and staff
   - 2.1 Everyone were confused about whether the delivery notes are referring to the shipment and whether the staff need to be further classified.
   - 2.2 The problem is solved by send the email to the TA.
   - 2.3 Since further clarification is needed, the discussion on the attributes table will be continued in the next meeting.
3 Allocated task:
   - 3.1 Due to the time was running short, The UML diagram will be discussed in the next meeting.
   - 3.2 Everyone need to prepare for the UML diagram for the next meeting.
4 The meeting adjournment and the next meeting
   - 4.1 The meeting was adjourned at 23:00 and the next meeting will be held at the 20th March at the 15:00.
   - 4.2 Keith booked the LC3 in the library learning common for the next meeting.

Third:

Minutes of the third meeting

Date: 20th March
Time: 15:00
Venue: LC3, Library Learning Common
Attending: Keith, Anvia, Steven
Recorder: Keith
Agenda:

1. Follow-up of the attributes table
2. UML design

Meeting Content:

1 The follow-up of the attributes table
   1.1 The TA replied the questions about the definition of the shipment and the staff.
   1.2 The TA said the type of the staff does not need to be defined.
   1.3 The TA said the delivery notes is referring to the shipment.
2 The attributes table
   2.1 The attributes of the staff and the shipment was completed in the meeting.
   2.2 Everyone spent 15 minutes to proofread the attributes table.
   2.3 The attributes table was finalized in the meeting.
3 UML diagram
   3.1 Everyone spent 20 minutes on drawing the UML diagram by themselves.
   3.2 Everyone shared their own work and had a discussion on the UML diagram.
   3.3 A conclusion had drawn and summarized into one diagram.
4 The allocation task:
   4.1 Keith will finalize the UML diagram and send to Steven and Anvia for proofread.
   4.2 The deadline of the task is 21st March.
5 The meeting adjournment and the next meeting
   5.1 The meeting was adjourned at 17:00 and the next meeting will be held after the activity 2 starts.

Fourth:

Minutes of the fourth meeting

Date: 1st April

Time: 22:35

Venue: Discord

Attending: Keith, Anvia, Steven

Recording: Keith

Agenda:

1. Discussion on the code
2. Discussion on the allocation of task

Meeting:

1 Discussion on the code
- 1.1 Everyone agreed to use the skeleton code to speed up the start of software development.
- 1.2 The skeleton code tried to be imported in the meeting.
- 1.3 Problem was found when importing the skeleton code.

2 Problem of importing the skeleton code
- 2.1 The skeleton code can not be imported as a gradle project.
- 2.2 To solve the problem, Anvia checked the FAQ document that it is allowed to import the skeleton code as Maven project.
- 2.3 The problem is solved after importing the Maven project.

3 The allocation of task
- 3.1 Keith will create the template of each scene to unify the GUI design of the program.
- 3.2 Anvia and Steven will design a class to solve the optimization of the calculation.
- 3.3 Steven will do the grant chart.
- 3.4 Anvia will do the burnt down chart.
- 3.5 Keith will do the integration of the program after the every function is finished by each member.

4 The meeting adjournment and next meeting
- 4.1 The meeting was adjourned at 23:59 and the next meeting will be held on 13th April.

Fifth:

Minutes of the fifth meeting

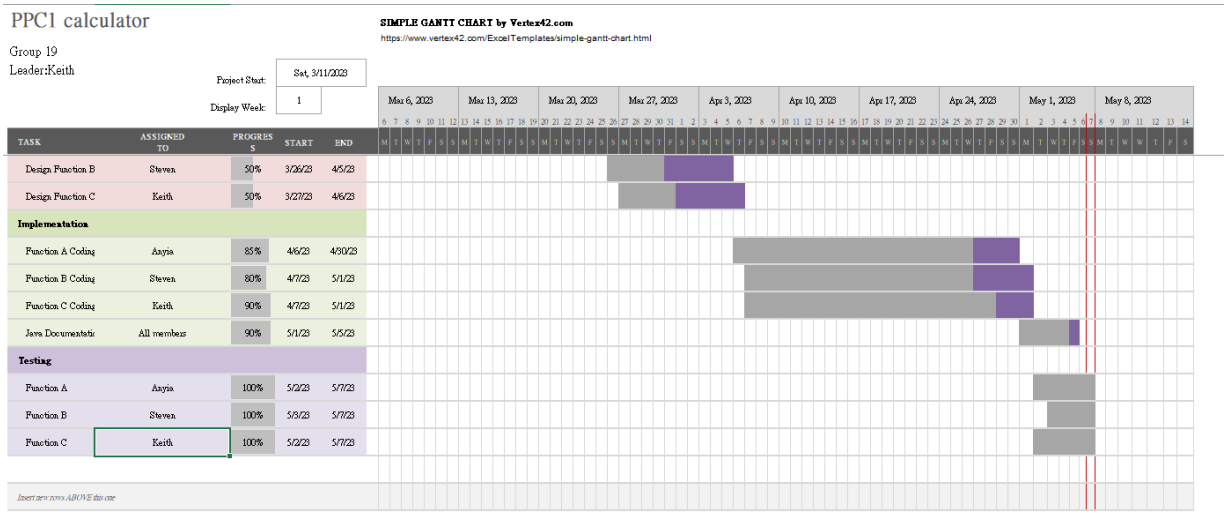
Date: 1st May
Time: 13:00
Venue: Discord
Attending: Keith, Anyia, Steven
Recording: Keith
Agenda:

1. Follow up on the progress
2. Timeslot of the demo
3. Review of the program
4. Unit testing and coverage
5. Java Documentation
6. Integration of the program

Meeting Content:

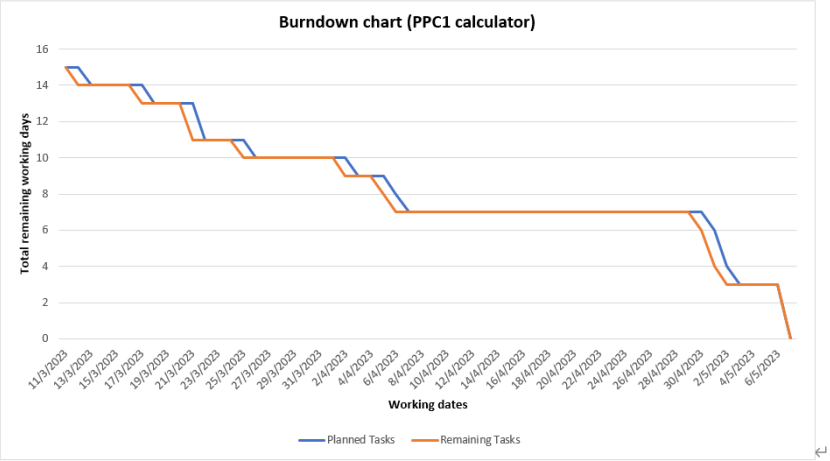
1 Follow up on the progress
  1.1 The burn down chart and the grant chart is done.
  1.2 Most of the group mates had finished 70% of the program
2 The timeslot of the demo
  2.1 The timeslot of the demo is decide and Keith will be the representative of group 19 in the demo.
3 Review of the program
  3.1 The master branch that created using the Eclipse is deleted in the meeting
4 Unit test
  4.1 The unit testing for each member will be done independently by each member since members are familiar with their own work
  4.2 Unit testing for each function should be more than 65%.
5 The Java Documentation
  5.1 The documentation of each function will be done by each member independently.
6 Integration of the program
  6.1 Keith will finalize the program.
  6.2 Unit testing report, coverage report, and the java documentation will be generated by Keith.
7 The meeting adjournment
  7.1 The meeting was adjourned at 13:55.

Gantt Chart:



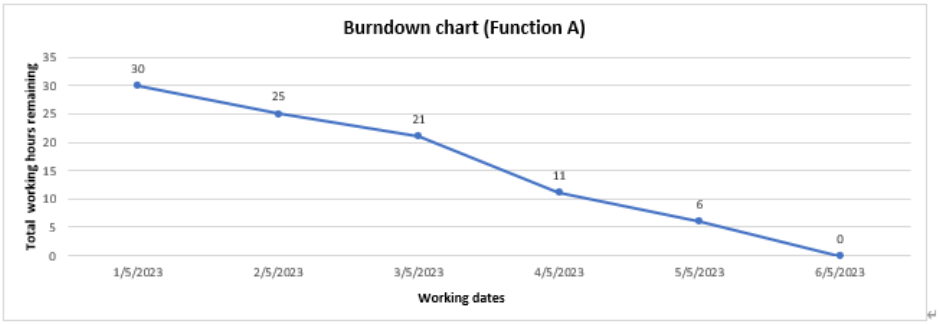
Burndown Chart:

Burndown chart for PPC1 calculator project



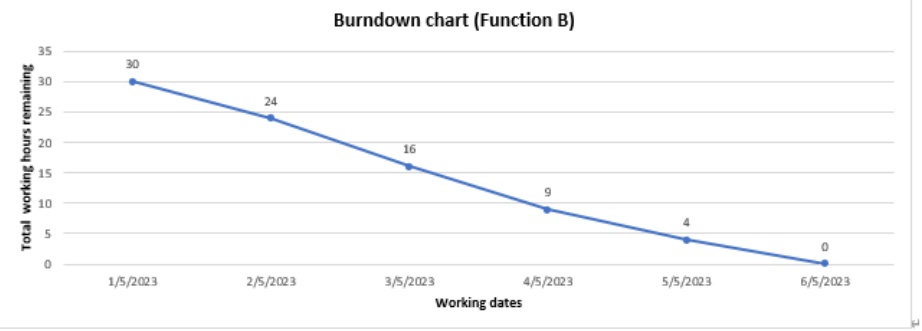
Task allocation & Burndown chart for Function A

Task allocation (Function A)						
Tasks	**1/5/2023**	**2/5/2023**	**3/5/2023**	**4/5/2023**	**5/5/2023**	**6/5/2023**
Design function A logic	6	3	2	0	0	0
Create and modify function A layout	6	4	3	1	1	0
Develop and modify function A	6	6	4	2	0	0
Write JX documentation	6	6	6	4	2	0
Write unit test case	6	6	6	4	3	0
Total working hours remaining	**30**	**25**	**21**	**11**	**6**	**0**



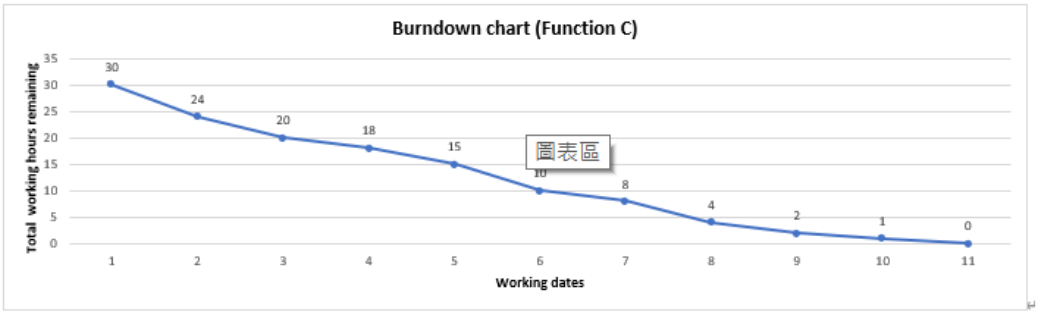
Task allocation & Burndown chart for Function B

Task allocation (Function B)						
Tasks	**1/5/2023**	**2/5/2023**	**3/5/2023**	**4/5/2023**	**5/5/2023**	**6/5/2023**
Design function B logic	6	3	0	0	0	0
Create and modify function B layout	6	4	2	0	0	0
Develop and modify function B	6	5	3	2	0	0
Write JX documentation	6	6	5	3	2	0
Write unit test case	6	6	6	4	2	0
Total working hours remaining	**30**	**24**	**16**	**9**	**4**	**0**

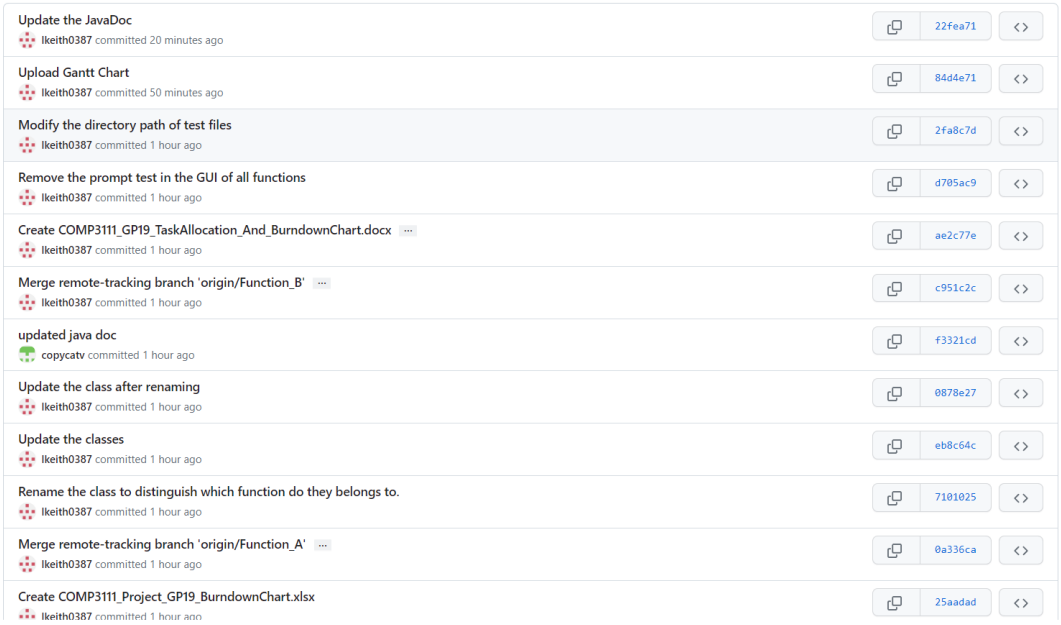


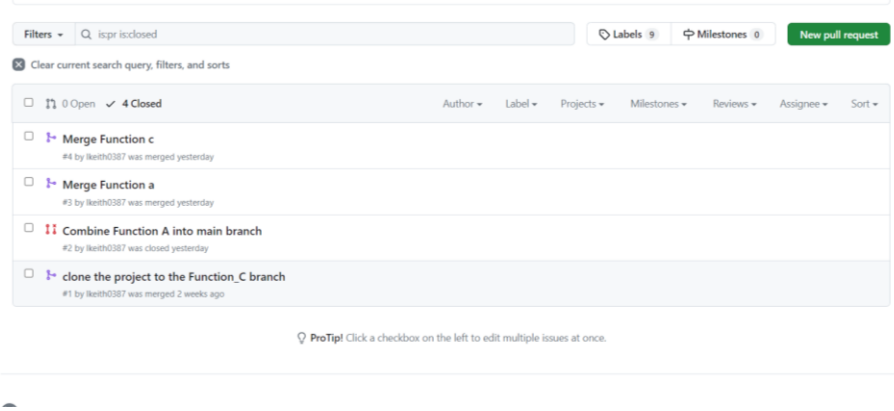
Task allocation & Burndown chart for Function C

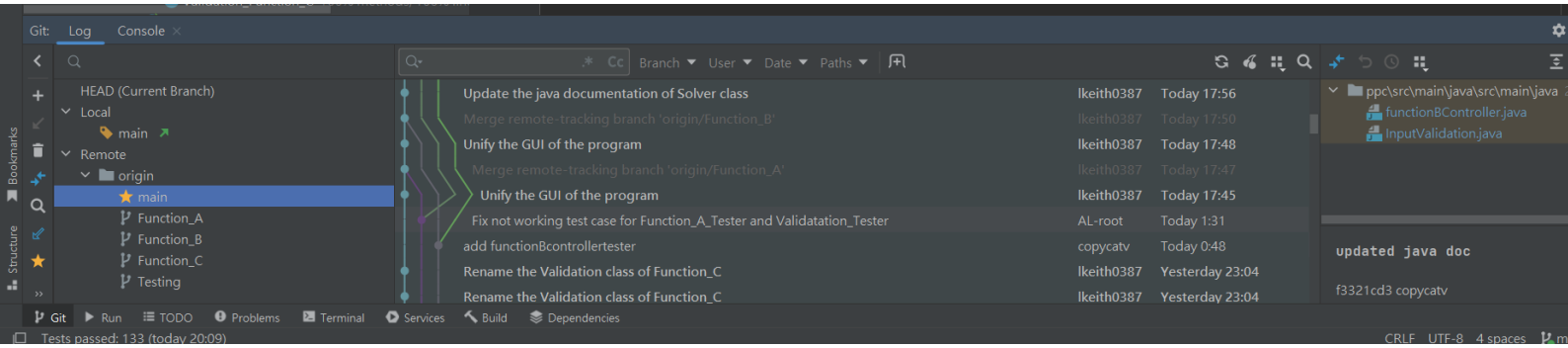
Task allocation (Function C)											
Tasks	**26/4/2023**	**27/4/2023**	**28/4/2023**	**29/4/2023**	**30/4/2023**	**1/5/2023**	**2/5/2023**	**3/5/2023**	**4/5/2023**	**5/5/2023**	**6/5/2023**
Design function C logic	6	2	0	0	0	0	0	0	0	0	0
Create and modify function C layout	6	4	3	2	1	0	0	0	0	0	0
Develop and modify function C	6	6	5	4	2	0	0	0	0	0	0
Write JX documentation	6	6	6	6	6	4	2	1	0	0	0
Write unit test case	6	6	6	6	6	6	6	3	2	1	0
Total working hours remaining	**30**	**24**	**20**	**18**	**15**	**10**	**8**	**4**	**2**	**1**	**0**



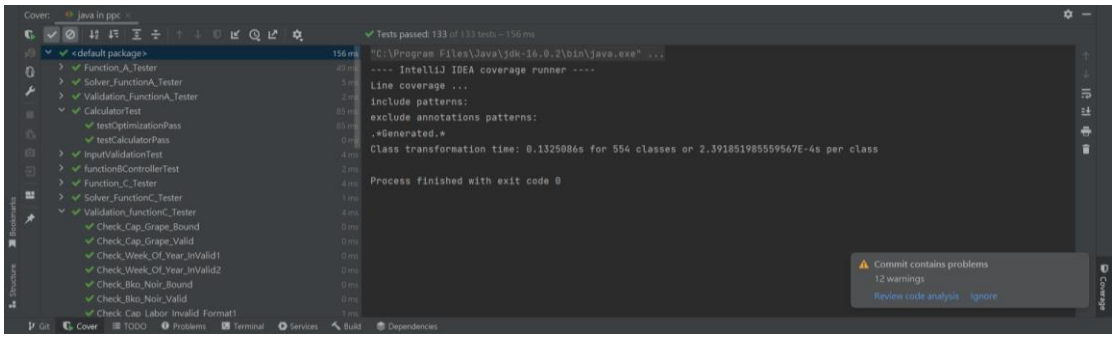
Git Commit Log:



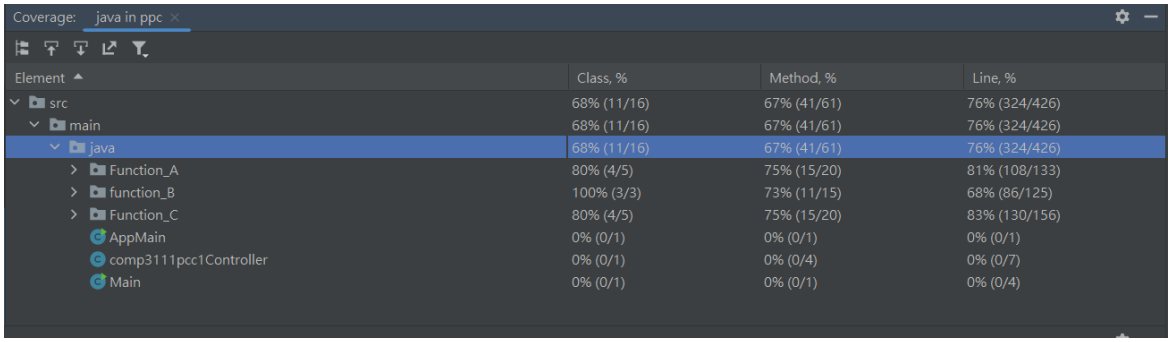




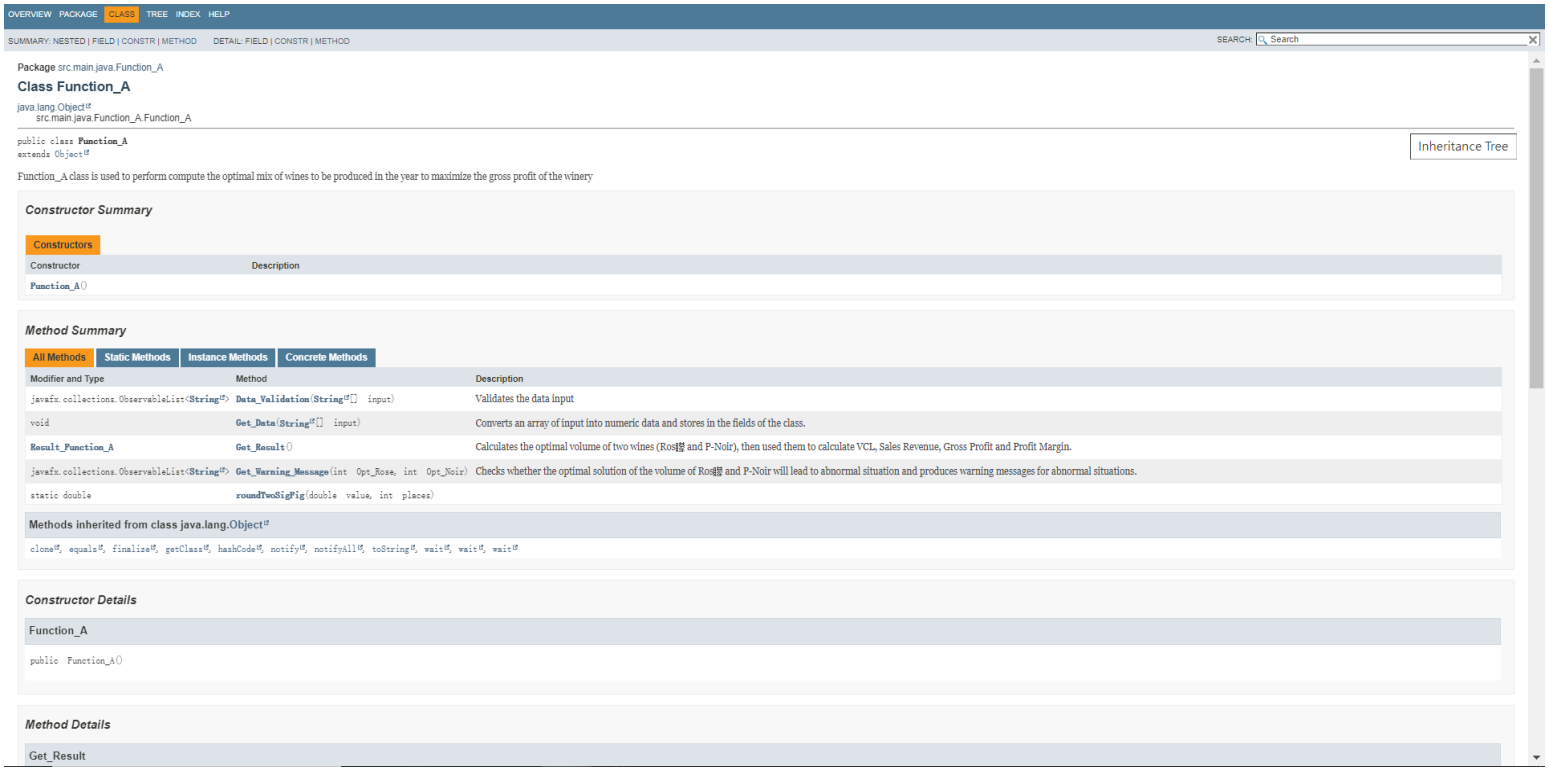
Unit Testing:



Coverage Report:



JavaDoc:



OVERVIEW PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package src.main.java.Function_A

Class Function_A

java.lang.Object
src.main.java.Function_A.Function_A

public class **Function_A**
extends Object

Inheritance Tree

Function_A class is used to perform compute the optimal mix of wines to be produced in the year to maximize the gross profit of the winery

Constructor Summary

Constructors

Constructor	Description
Function_A()	

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
javafx.collections.ObservableList<String>	**Data_Validation**(String[] input)	Validates the data input
void	**Get_Data**(String[] input)	Converts an array of input into numeric data and stores in the fields of the class.
Result_Function_A	**Get_Result**()	Calculates the optimal volume of two wines (Ros酵 and P-Noir), then used them to calculate VCL, Sales Revenue, Gross Profit and Profit Margin.
javafx.collections.ObservableList<String>	**Get_Warning_Message**(int Opt_Rose, int Opt_Noir)	Checks whether the optimal solution of the volume of Ros酵 and P-Noir will lead to abnormal situation and produces warning messages for abnormal situations.
static double	**roundTwoSigFig**(double value, int places)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

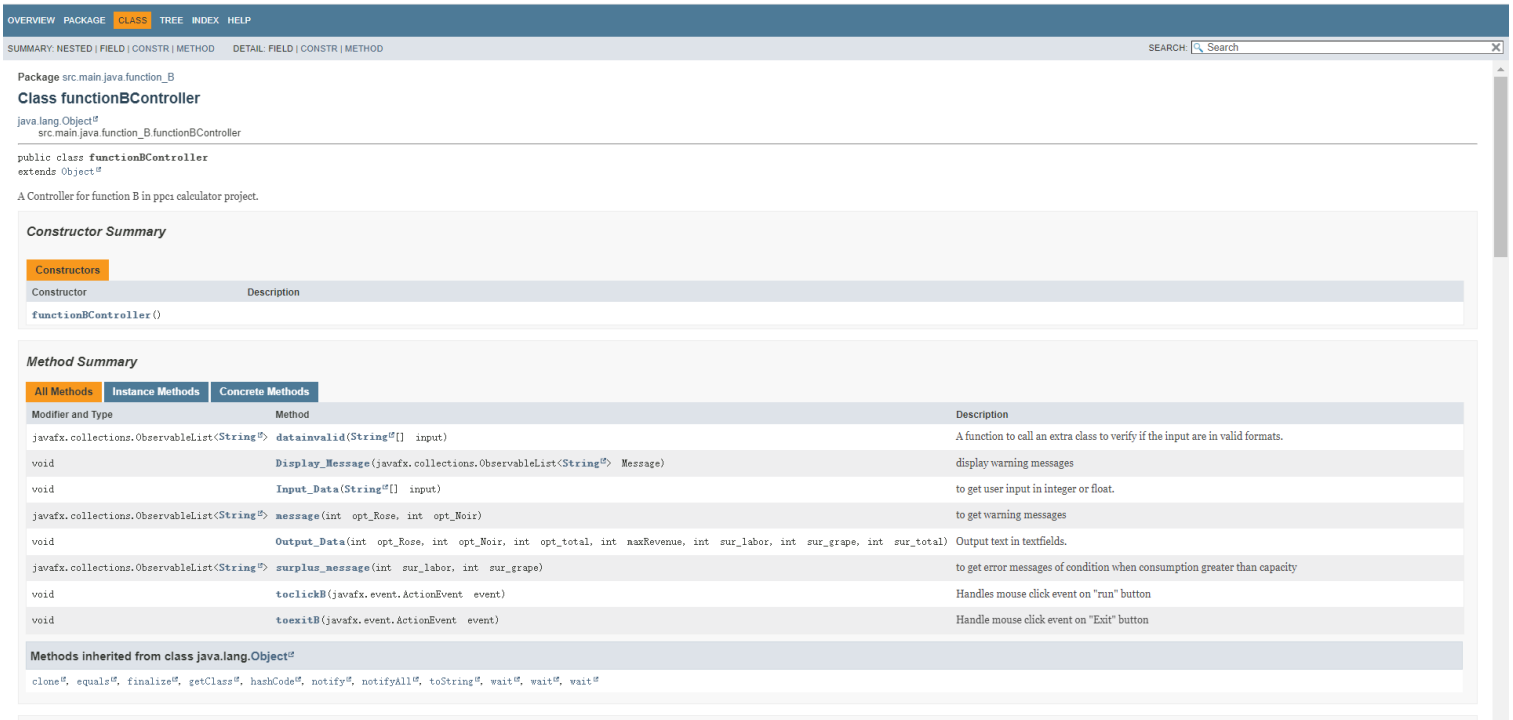
Constructor Details

Function_A

public Function_A()

Method Details

Get_Result



OVERVIEW PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package src.main.java.function_B

Class functionBController

java.lang.Object
src.main.java.function_B.functionBController

public class **functionBController**
extends Object

A Controller for function B in ppc1 calculator project.

Constructor Summary

Constructors

Constructor	Description
functionBController()	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
javafx.collections.ObservableList<String>	**datainvalid**(String[] input)	A function to call an extra class to verify if the input are in valid formats.
void	**Display_Message**(javafx.collections.ObservableList<String> Message)	display warning messages
void	**Input_Data**(String[] input)	to get user input in integer or float.
javafx.collections.ObservableList<String>	**message**(int opt_Rose, int opt_Noir)	to get warning messages
void	**Output_Data**(int opt_Rose, int opt_Noir, int opt_total, int maxRevenue, int sur_labor, int sur_grape, int sur_total)	Output text in textfields.
javafx.collections.ObservableList<String>	**surplus_message**(int sur_labor, int sur_grape)	to get error messages of condition when consumption greater than capacity
void	**toclickB**(javafx.event.ActionEvent event)	Handles mouse click event on "run" button
void	**toexitB**(javafx.event.ActionEvent event)	Handle mouse click event on "Exit" button

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Package src.main.java.Function_C

Class Function_C

java.lang.Object
src.main.java.Function_C.Function_C

public class **Function_C**
extends Object

The Function_C class contains the process performed in the function C.

Constructor Summary

Constructors

Constructor	Description
Function_C()	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
javafx.collections.ObservableList<String>	**Data_Validation**(String[] input)	Validates the data input
void	**Get_Data**(String[] input)	Converts an array of the texted input into numeric data and stores in the fields of the class.
Result_Function_C	**Get_Result**()	Calculates the optimal solution of the volume of Ros膠 and P-Noir with or without any backorder of Ros膠 and P-Noir to maximize the revenue
javafx.collections.ObservableList<String>	**Get_Warning_Message**(int Opt_Rose, int Opt_Noir)	Checks whether the optimal solution of the volume of Ros膠 and P-Noir will lead to abnormal situation and produces warning messages for abnormal situations.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

Function_C

```
public Function_C()
```


Package src.main.java.Function_C

Class Solver_Function_C

java.lang.Object
src.main.java.Function_C.Solver_Function_C

public class **Solver_Function_C**
extends Object

The Solver_Function_C class provides solutions for calculation of some outputs in functions, including optimal volume of Ros膠, optimal volume of P-Noir, and the revenue obtain with specified volume of Ros膠 and P-Noir.

Constructor Summary

Constructors

Constructor	Description
Solver_Function_C(int Cap_Labor, int Cap_Grape, float Prc_Rose, float Prc_Noir)	Initializes a new Solver_Function_C object that contains information on the capacity of Labor and Grape and the price of Ros膠 and P-Noir.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
float	**Calculation**(int Num_Rose, int Num_Noir)	Calculates the revenue obtained with specified volume of Ros膠 and P-Noir.
int[]	**Opt_Solution**(boolean celling, int celling_of_Rose, int celling_of_Noir)	Calculates the optimal solution of volume of Ros膠 and P-Noir to obtain the highest revenue and store the result into an int array.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

Solver_Function_C

```
public Solver_Function_C(int Cap_Labor,
                         int Cap_Grape,
                         float Prc_Rose,
                         float Prc_Noir)
```

Initializes a new Solver_Function_C object that contains information on the capacity of Labor and Grape and the price of Ros膠 and P-Noir.

Package src.main.java.function_B

Class Calculator

java.lang.Object
  src.main.java.function_B.Calculator

Inheritance Tree

```
public class Calculator
extends Object
```

A Calculator class used to do all the calculation on optimization

Constructor Summary

Constructors

Constructor	Description
`Calculator()`	

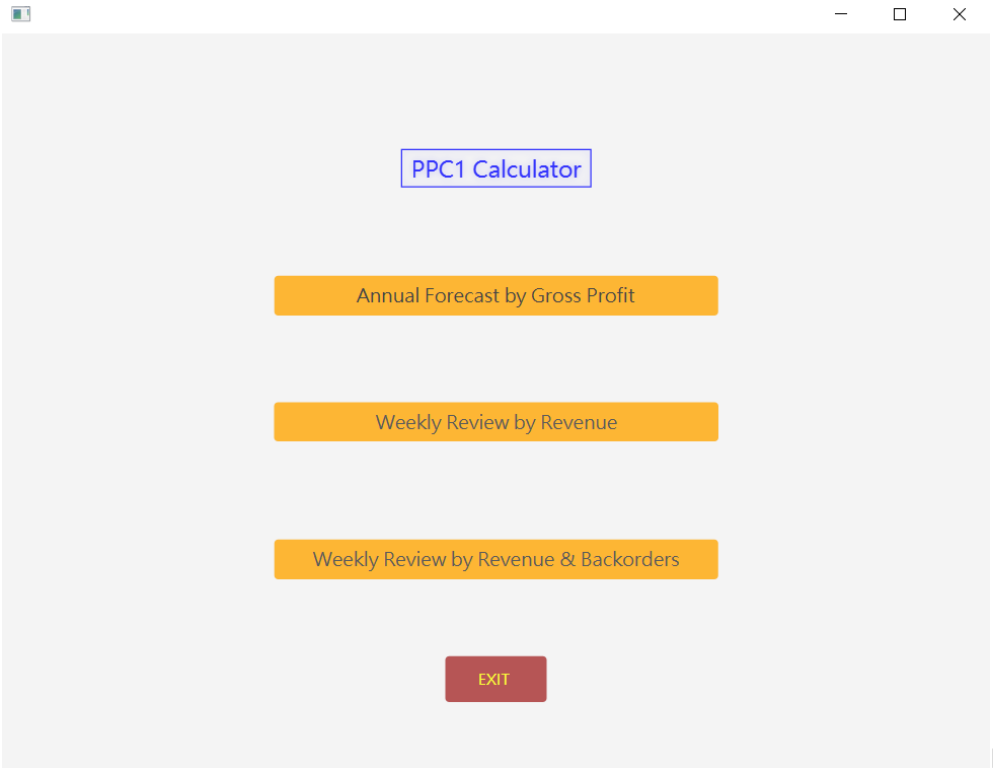
Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

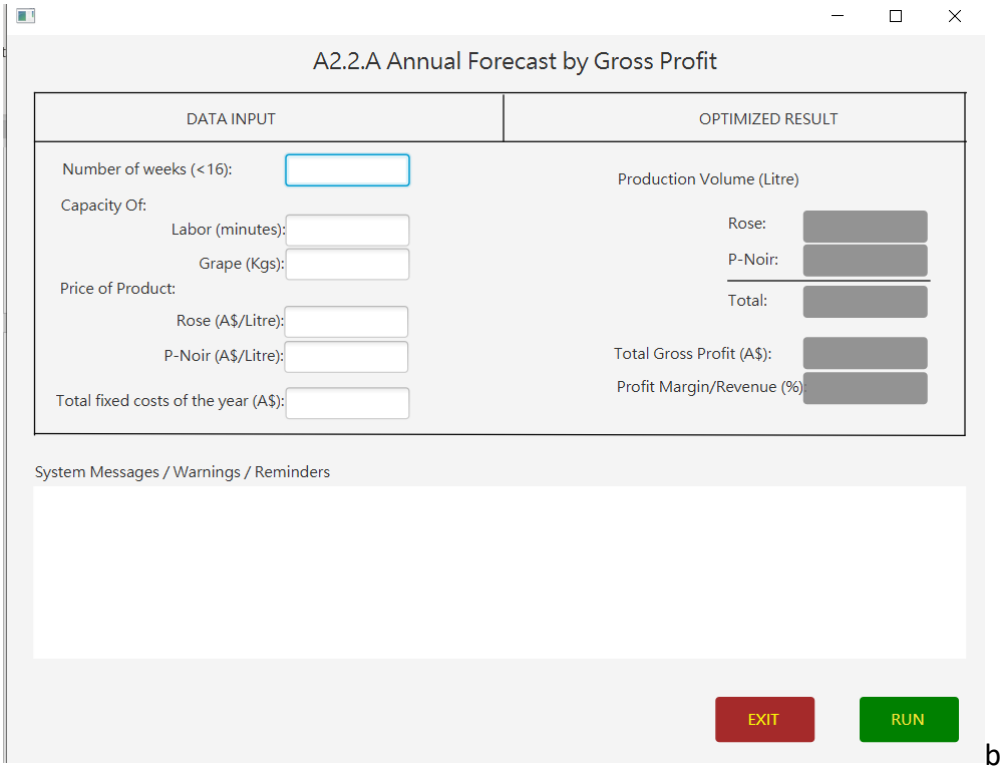
Modifier and Type	Method	Description
int	`Calculation(int Num_Rose, int Num_Noir, float prc_rose, float prc_noir)`	Calculation based multiplication and addition to calculate the revenue
int[]	`Optimization(int cap_labor, int cap_grape, float prc_rose, float prc_noir)`	Calculate the optimize solution of opt_Rose,opt_Noir

Methods inherited from class java.lang.Object

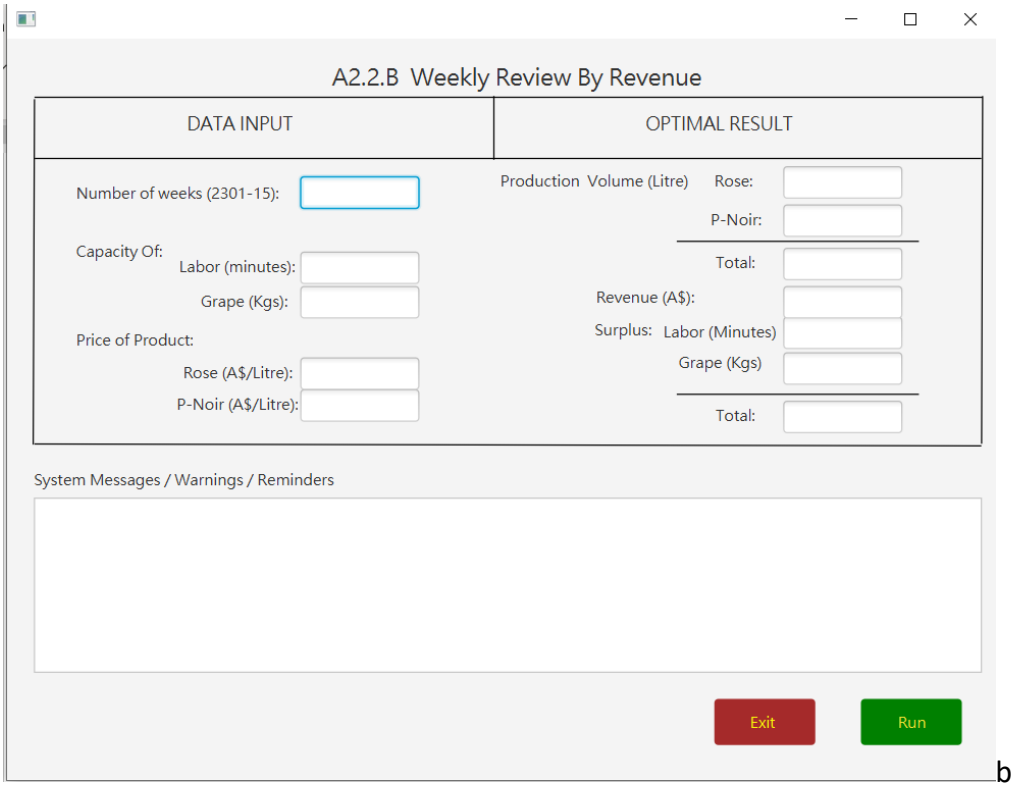
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait



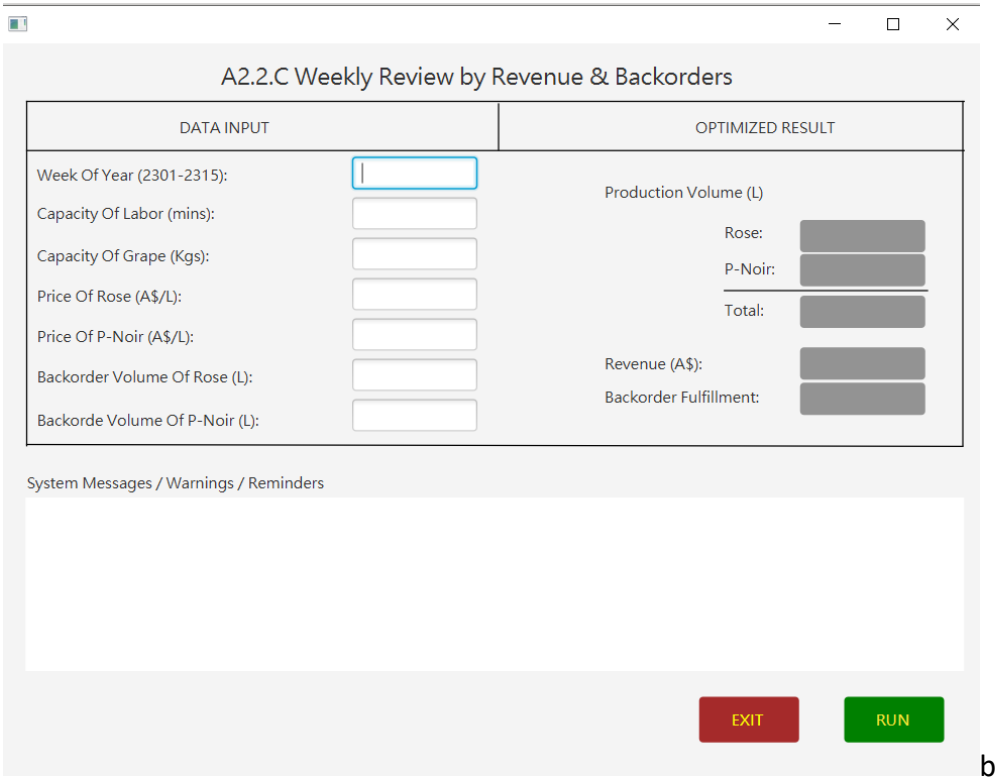
Menu



branch A



branch B



branch C