Software Design Specifications

Version 1.0

Prepared by the Delta Group

Anh Pham

Angelito Sabino

Chandler Atchley

Krish Shah

Kazuki Kanke

Sammy Portillo

11/20/2019

Revision History 1

Date	Name	Section	Reason For Changes
11/12/2019	Sam Portillo	All	Initial template
11/12/2019	Sam Portillo	1	Software Specification Outline Descriptions.
11/12/2019	Sam Portillo	2	System View Descriptions
11/12/2019	Sam Portillo	3.100.1	os Package View Descriptions
11/12/2019	Sam Portillo	4.100.1	os Class View Descriptions
11/15/2019	Sam Portillo	3.100.2	server Package View
11/15/2019	Krish Shah Monteiro	3.600.1	Package View -Adding the buttons class
11/15/2019	Sam Portillo	3.100.3	Android client Package View
11/15/2019	Sam Portillo	4.100.2	clients, server Class Views
11/15/2019	Sam Portillo	Appendix	Copied SRS Appendix & styled format.
11/17/2019	Kazuki Kanke	3.300	Package of shopping cart
11/17/2019	Kazuki Kanke	4.300.1	Main Activity class Views and description
11/17/2019	Kazuki Kanke	4.300.2	ItemAdapter class views
11/17/2019	Krish Shah Monteiro	3.600	Finishing the buttons class
11/17/2019	Kazuki Kanke	4.300.3	ImageActivity class views
11/17/2019	Kazuki Kanke	4.300.4	DetailActivity class view and description
11/18/2019	Angelito Sabino	3.500	Package View of Tet-ri-poff
11/18/2019	Sam Portillo	RH2	Revision History 1 is full.

Revision History 2

Date	Name	Section	Reason For Changes
11/17/2019	Anh Pham	3.400	Package of Messenger App
11/17/2019	Anh Pham	4.400.1	Messenger - Client View class
11/17/2019	Anh Pham	4.400.2	Messenger - Authentication Class
11/17/2019	Anh Pham	4.400.3	Messenger - Server View/ Controller class
11/17/2019	Anh Pham	4.400.4	Messenger - Content Database class
11/17/2019	Anh Pham	4.400.5	Messenger - User Database class
11/18/2019	Krish Shah Monteiro	3.600	Package View - Adding the addfeat class
11/19/2019	Anh Pham	3.400,4.400	Refine the diagrams and correct some information.
11/19/2019	Angelito Sabino	3.500,4.500	Finished Package View, Added more Class view stuff
11/20/2019	Krish Shah Monteiro	3.600 & 4.600	Making changes pointed out by the group leader. Adding the class view for the calculator
11/20/2019	Anh Pham	all	Checking format, cleaning up unnecessary texts
11/20/2019	Sam Portillo	3.100.4	Added Android Client Package
11/20/2019	Sam Portillo	2.100.2	Added Sequence Diagram
11/20/2019	Sam Portillo	4.100.3	Client Class
11/20/2019	Sam Portillo	4.100.4	Android Client Class
11/20/2019	Sam Portillo	all	Change color to black, checking spelling, requesting changes, waiting…, done.

Software Design Specification

The Software Design Specification Outline

1. Introduction

1.1 Purpose of this document

- This Software Design Specification (SDS) completes the System Design phase of the Waterfall model.

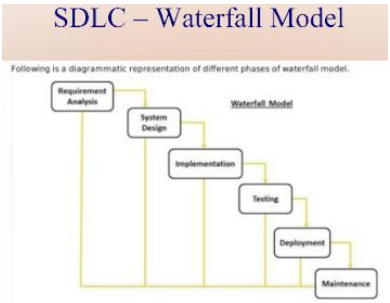


Figure 1.1.1

- Reference and satisfy all SRS requirements.
- Show how all SRS requirements will be satisfied.
- Provide an explicit comprehensive single source detailed blue-print document, that can be a developers only reference, to develop an OS & System Applications.
- Illustrating how a system is to be assembled (put together) by providing a high level view of the system that drills down to its atomic parts (data dictionary) in a systematic comprehensible approach.

1.2 Scope of the development project

This system includes an OS & System Applications - Text Editor, Messaging App, Shopping Cart, Calculator, Tetris game.

1.3 Package Developers

The following table denotes each member developer ID, developer name, and the package that the developer is solely responsible for developing.

ID	Delta Member	Package	System Name
100	Sam Portillo	os, server, client, android_client	Operating System
200	Chandler Atchley	editor	Text Editor
300	Kazuki Kanke	shop	Shopping Cart
400	Anh Pham	messenger	Messaging App
500	Angelito Sabino	tetris	Tetris
600	Krish Shah	calculator	Calculator

Figure 1.5
Package Developer Reference Table

1.4 Meta-Architecture

The following are requirements to this software project:

- Each developer write at least 500 statements.
- At least 80% is Java.
- Submit this SDS on or before November 20, 2019.

These requirements imply that each developer write their own class. Furthermore since there is a limitation on collaborating this implies that each developer write their own application. In addition, this implies that each developer write their own software design specifications for what they will be developing.

1.5 Overview of document

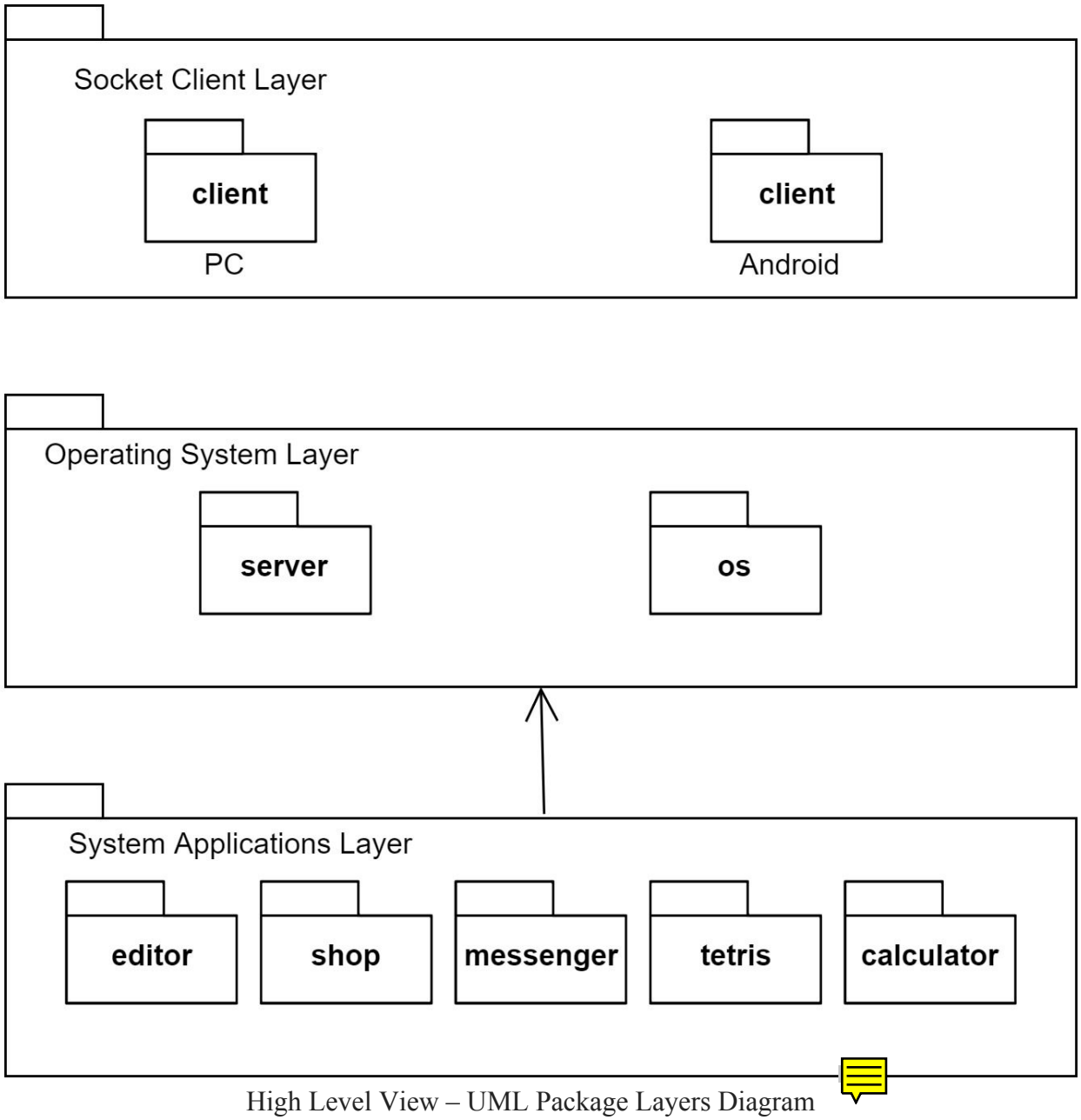
This SDS document is divided into 3 levels of detail:

- System View → High level layer description of the entire OS & System Applications drilling down to a package entity.
- Package View → Detailed description of each package drilling down to a class entity.
- Class View → Detailed description of each class entity.

2. System View

The following diagram illustrates a high level layer package view of the entire OS & System Applications.

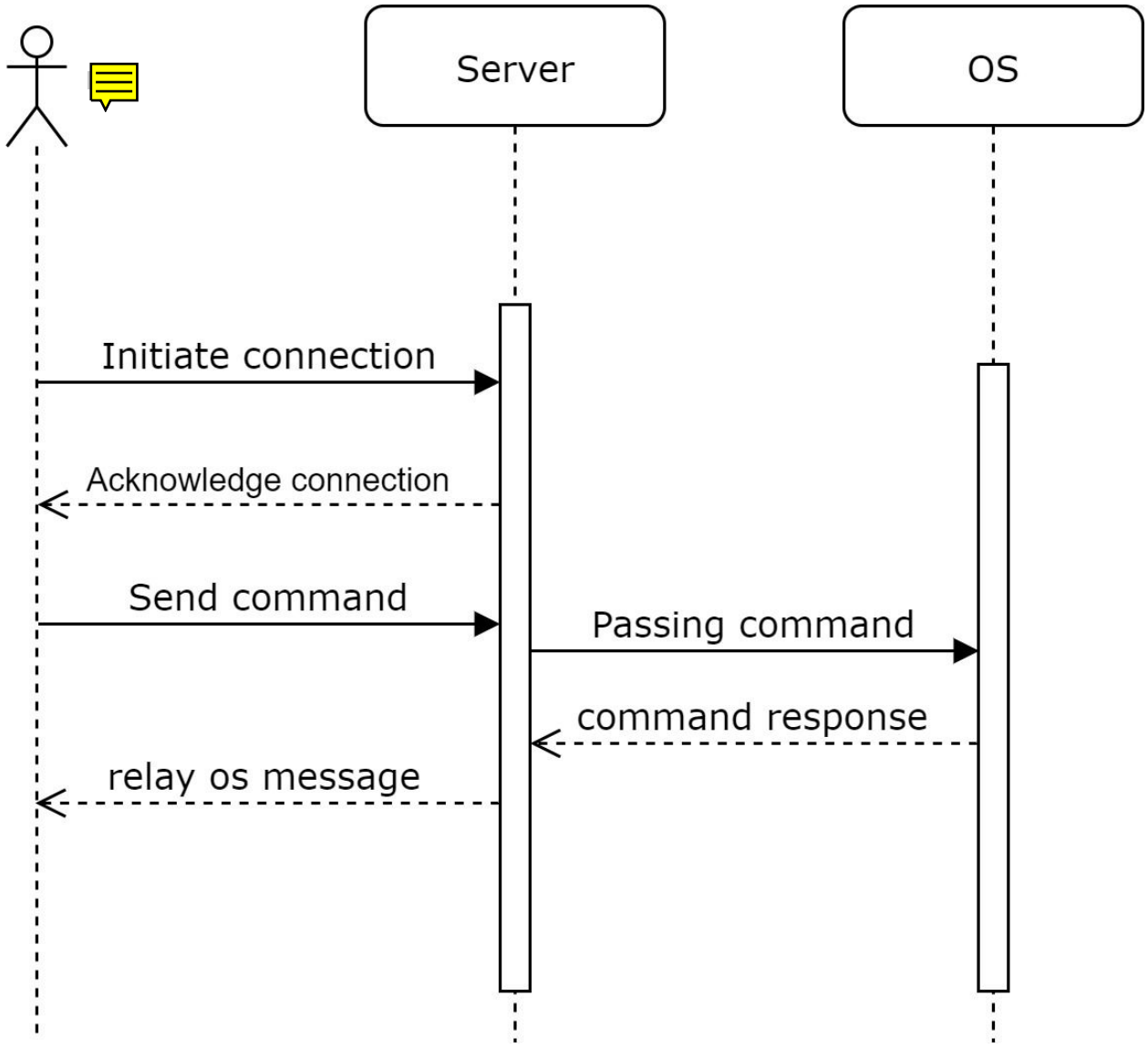
UML System Diagram:



High Level View – UML Package Layers Diagram
Figure 2.100.1

UML Sequence Diagram:

4



High Level View – UML Sequence Diagram
Figure 2.100.2

Identification: System
Type: System
Purpose: To support our architectural pattern.
Function: Executes a single user OS and shell that can execute System Applications.
Serve multi users via its multi threaded socket server over the LAN or WAN.
Provide client PC terminal and client Android terminal connectivity.
Subordinates: None

Dependencies: None
Interfaces: None
Resources: None
Processing: Not applicable
Data: None

3. Package View

This section provides a detailed description of each package.

3.100.1 os Package View

The following is an illustration of the os package, the OS makes up the Operating System layer.

UML Class Diagrams:

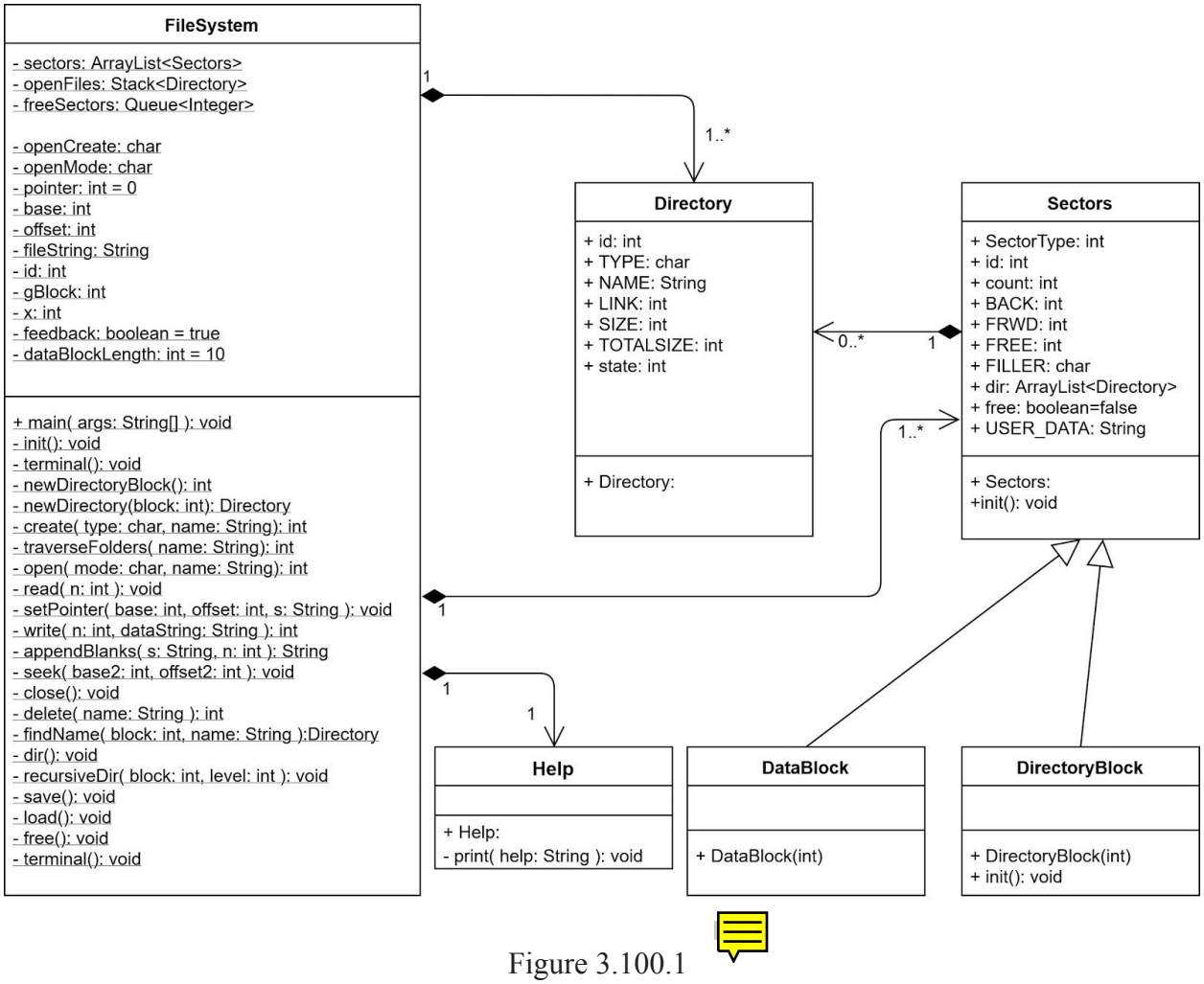


Figure 3.100.1
os UML Package View Diagram

Identification: os Package
Type: Package
Purpose: [illegible]e to the design paradigm of our architecture.
Function: Executes a single user OS and shell that can execute System Applications.
Subordinates: None
Dependencies: None
Interfaces: None

Resources:	None
Processing:	Not applicable
Data:	None

3.100.2 server Package View

The following is an illustration of the server package, which provides a multi threaded socket server connectivity to make up part of the Operating System layer.

UML Class Diagrams:

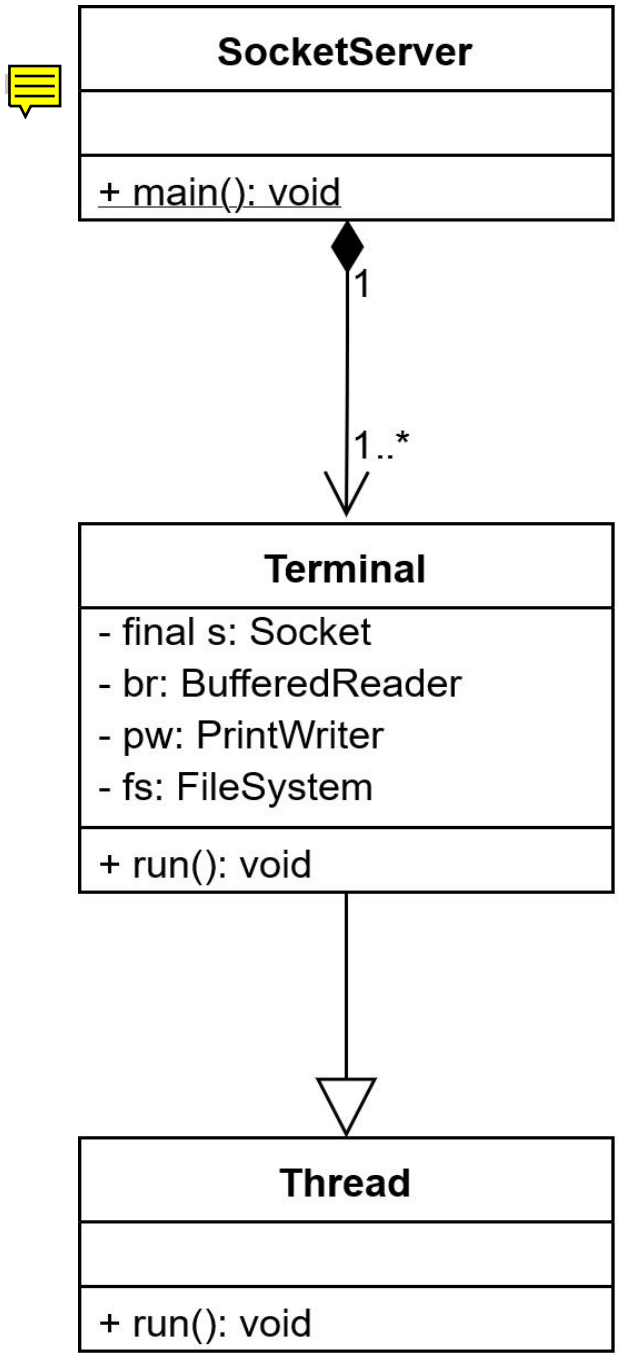


Figure 3.100.2
server UML Package View Diagram

Identification: server Package
Type: Package
Purpose: Due to the design paradigm of our architecture.
Function: Serve multi users via its multi threaded socket server over the LAN and WAN.
Subordinates: None
Dependencies: None
Interfaces: None
Resources: None
Processing: Not applicable
Data: None

3.100.3 client Package View

The following is an illustration of the client socket layer. This layer has one package with one class that provides remote socket connectivity over the LAN and WAN.

UML Class Diagram:

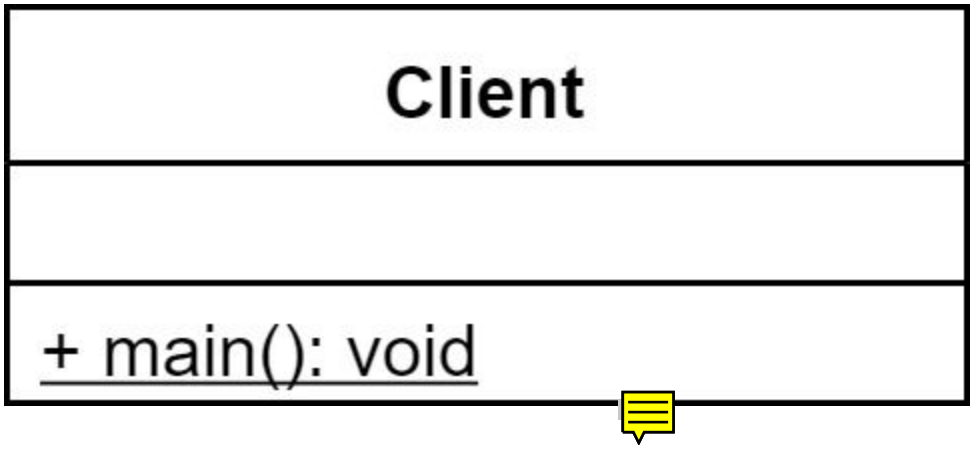


Figure 3.100.3
Client UML Package View Diagram

Identification:	client Package
Type:	Package
Purpose:	Due to the design paradigm of our architecture.
Function:	Creates a client terminal with remote socket connectivity to the OS.
Subordinates:	None
Dependencies:	None
Interfaces:	None
Resources:	None
Processing:	Not applicable
Data:	None

3.100.4 Android client Package View

The following is an illustration of the Android Socket client layer. This layer has one package that provides remote socket connectivity over the LAN and WAN.

UML Class Diagram:

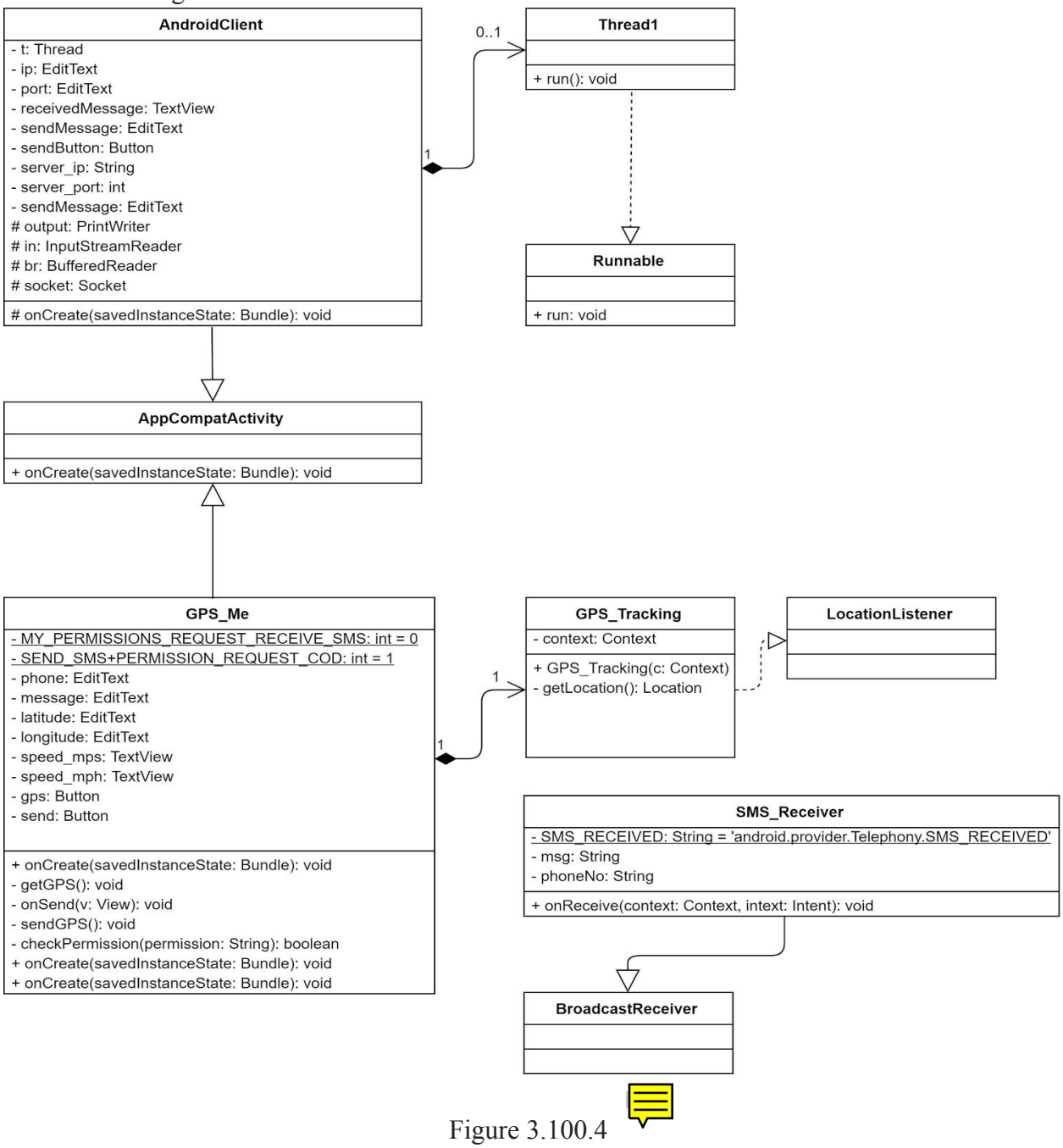
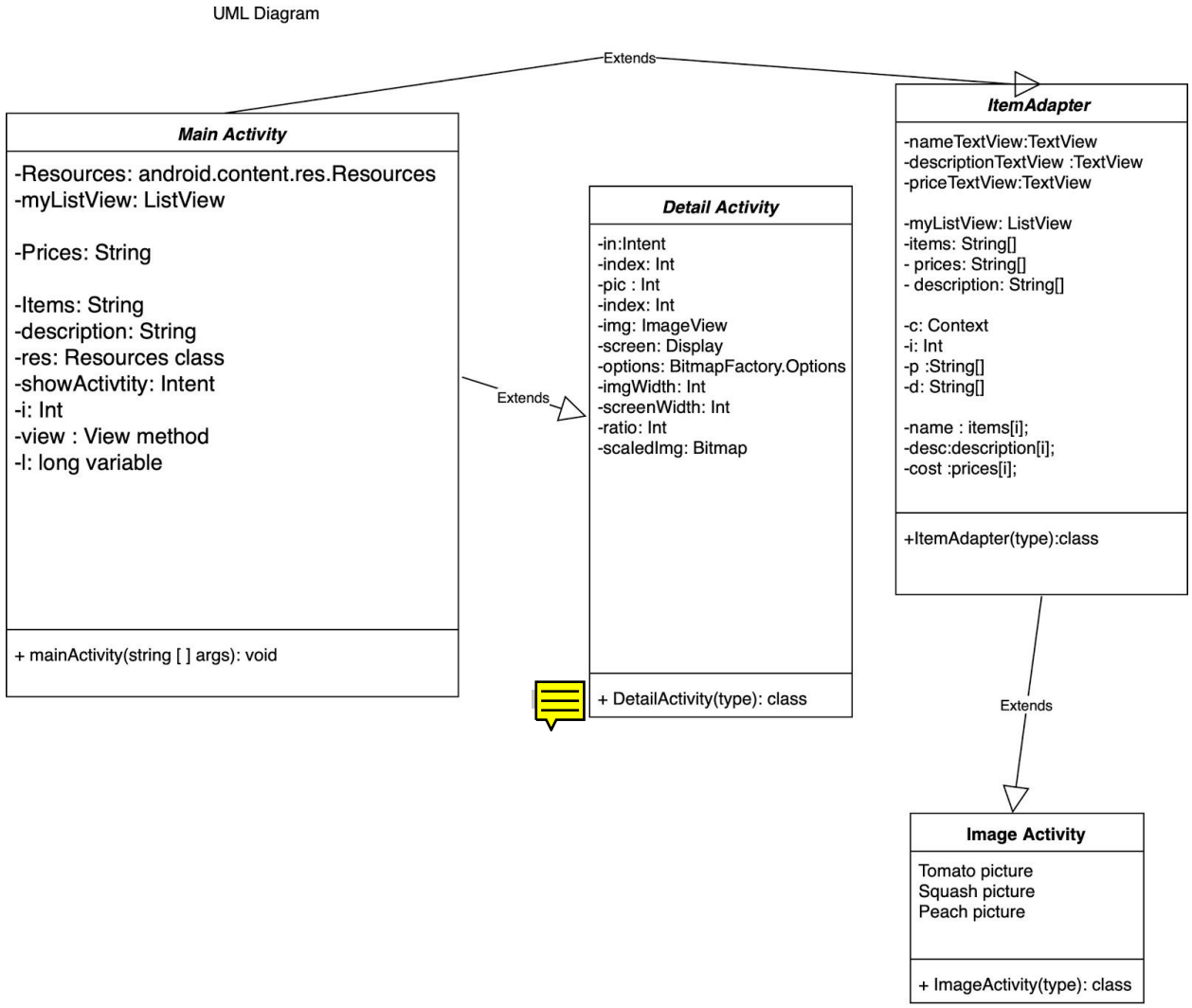


Figure 3.100.4
Android Client UML Package View Diagram

Identification:	Android client Package
Type:	Package
Purpose:	Due to the design paradigm of our architecture.
Function:	Creates an Android client with remote socket connectivity.
Subordinates:	None
Dependencies:	None
Interfaces:	None
Resources:	None
Processing:	Not applicable
Data:	None

3.300 Shopping Cart Packaging view

The following illustration represents the structure of shopping cart.



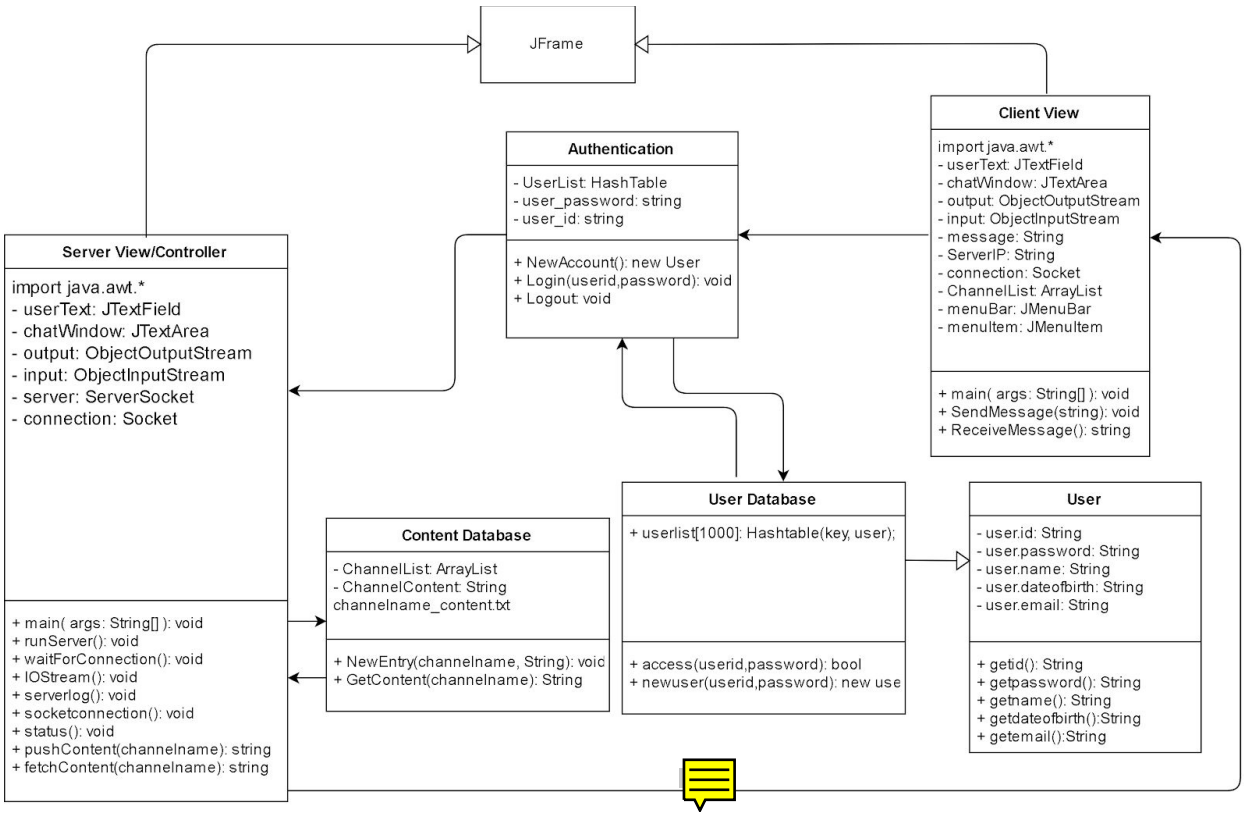
Identification: Shopping Cart package
Type: Package
Purpose: The purpose of shopping cart is that user can select product, and can view the price and product in app. User can check anywhere and anytime.
Function: This package has shopping cart program, and if we open android studio, it will open interface of shopping cart. User can choose products what they want, and they can select how manys do they want.
Subordinates: None
Dependencies: None
Interfaces: None
Resources: None

Processing: When user clicked button, this main activity will make an order for the other file, and it will show up the pictures.

Data: Picture of product data

3.400 The Messenger App Package view

The following is an illustration of the Messenger Application package. Demonstrating the relationship between each module.



Identification: Messenger App Package
Type: Package
Purpose: Due to the design paradigm of our architecture.
Function: Executes Messenger Application.
Subordinates: None
Dependencies: None
Interfaces: None
Resources: None
Processing: Not applicable
Data: None

3.500 Tet-ri-poff Package view

The Figure below is a UML Diagram representing the relationship between the classes and methods that will be used in the “Tet-ri-poff” Tetris Game.

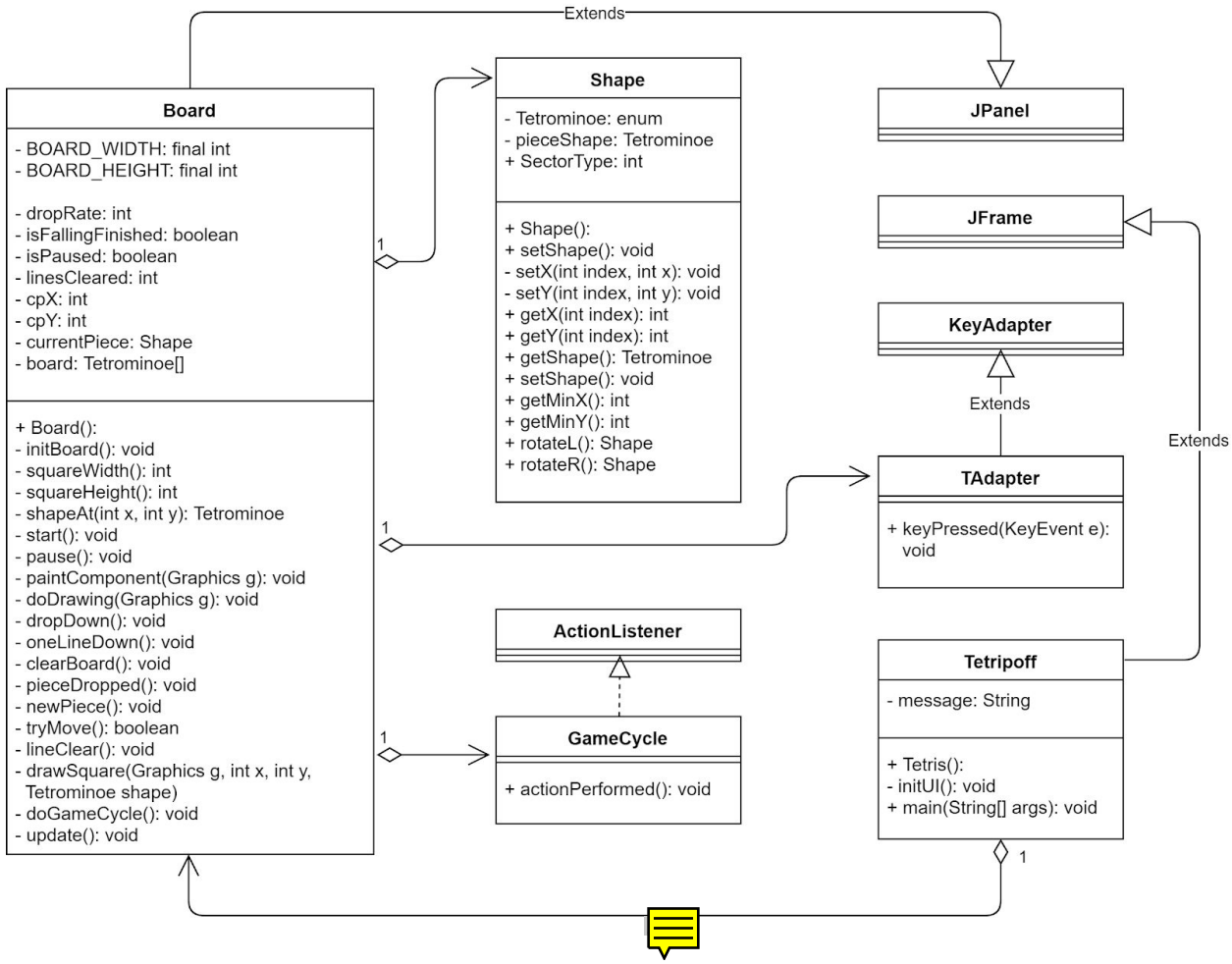
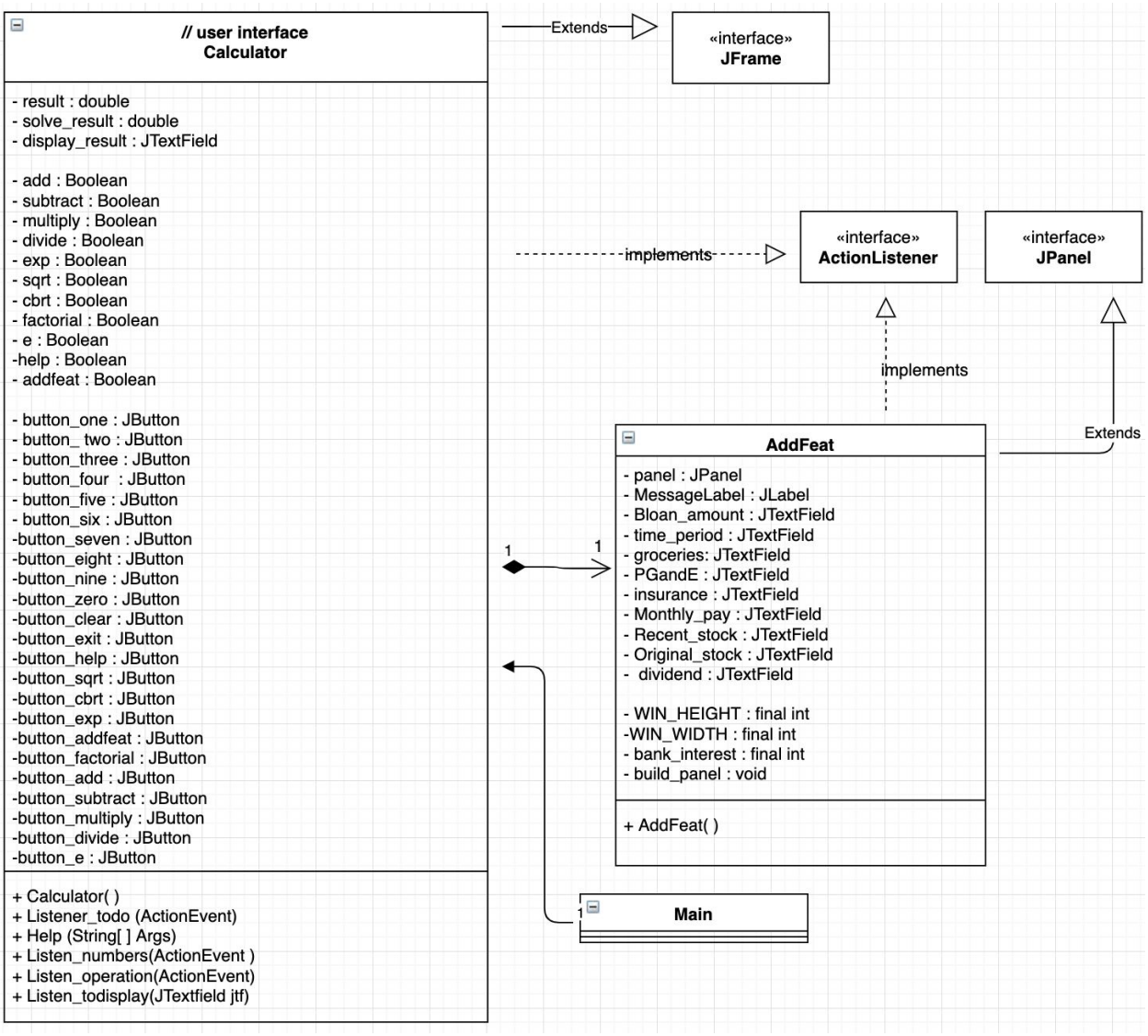


Figure 3.500

Identification: Tetripoff Package
Type: Package
Purpose: Due to the design paradigm of our architecture.
Function: A playable version of the original Tetris game.
Subordinates: None
Dependencies: None
Interfaces: None
Resources: None
Processing: Not applicable
Data: None

3.600 BigCalc (The calculator) Package View

3.600.1
The following is a UML diagram that covers the Package view of the calculator. It presents the different classes and relationships established amongst them.



Identification:	Bigcalc Package
Type:	Package
Purpose:	Due to the design paradigm of our architecture
Function:	To formulate results based on the user inputs
Subordinates:	None
Dependencies:	None
Interfaces:	None
Resources:	None
Processing:	Not applicable
Data:	None

4. Class View

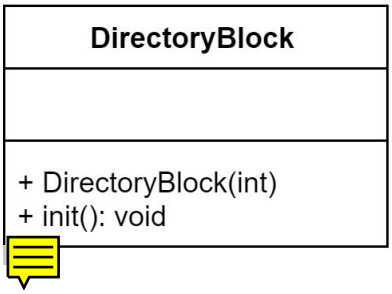
Class View provides a detailed description of each class.

4.100.1 OS classes.

4.100.1.1 DirectoryBlock

The following describes the DirectoryBlock class.

UML Class Diagram:



Identification: DirectoryBlock
Type: Class
Purpose: Satisfies SR 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.5, 4.100.9.7 4.100.9.8
Function: Allows a sector to be utilized as a directory block of Directories.
Subordinates: None
Dependencies: None
Interfaces:

1

Interface: DirectoryBlock(int freeSector)
Description: The DirectoryBlock constructor creates a Sector that is used for containing directory entries.
The DirectoryBlock class represents a directory that can contain up to 32 en . These entries can either be a directory or a file.
Parameters:

Parameters	Range	Description
int freeSector	[0, 100]	freeSector is an int that is used to id the DirectoryBlock in an array. The id is analogous to a pointer address in memory.

Return:Not Applicable

2

Interface: init( )

Description: The init() method initializes a directory block to contain 32 entries. These entries can be either a directory name or a file name.

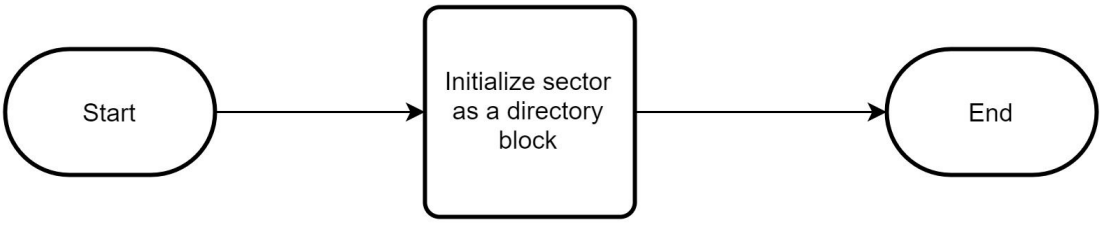
Parameters:

Parameters	Range	Description
None		

Return:void

Resources: None

Processing:

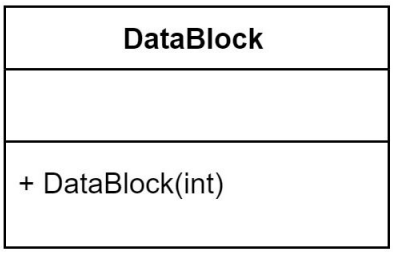


Data: None

4.100.1.2 DataBlock

The following describes the DataBlock class.

UML Class Diagram:



Identification: DataBlock
Type: Class
Purpose: Satisfies SRS 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7 4.100.9.8
Function: Allows a sector to be utilized as a data block.
Subordinates: None
Dependencies: None
Interfaces:

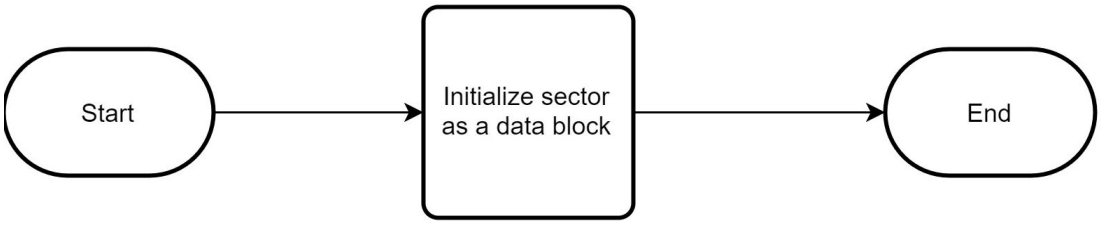
1

Interface: DataBlock(int freeSector)
Description: The DataBlock class is a Sector class that contains user data.
Parameters:

Parameters	Range	Description
int freeSector	[0, 100]	freeSector is an int that is used to id the DataBlock in an array. The id is analogous to a pointer address in memory.

Return:Not Applicable

Resources: None
Processing:

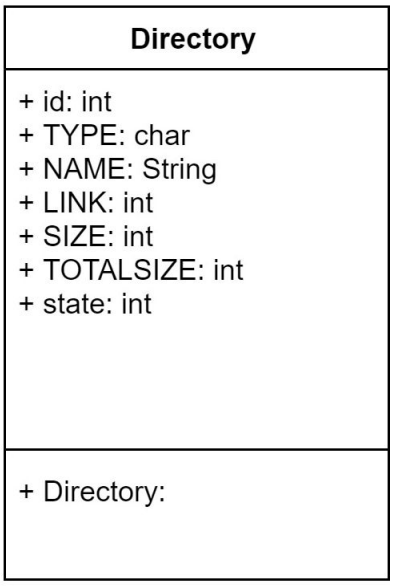


Data: None

4.100.1.3 Directory

The following describes the Directory class.

UML Class Diagram:



Identification: Directory
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.5, 4.100.9.7 4.100.9.8
Function: The Directory class implements Serializable in order to save its state. The Directory class represents one entry in a directory, that can be used to point to another directory (block) or a data block.
Subordinates: None
Dependencies: None
Interfaces:

1

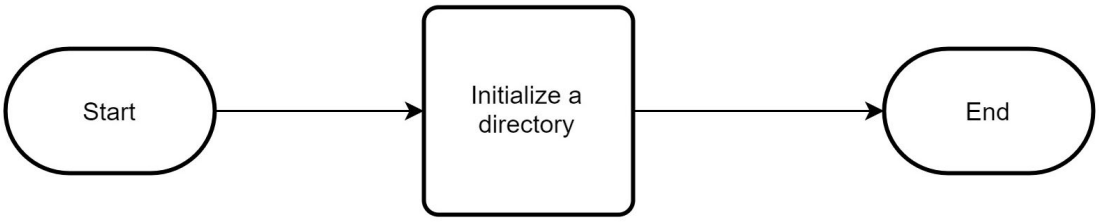
Interface: Directory( )
Description: The Directory constructor creates a directory entry that will be used to point to either another directory or a data block.

Parameters:

Parameters	Range	Description
None		

Return: Not Applicable

Resources: None
Processing:



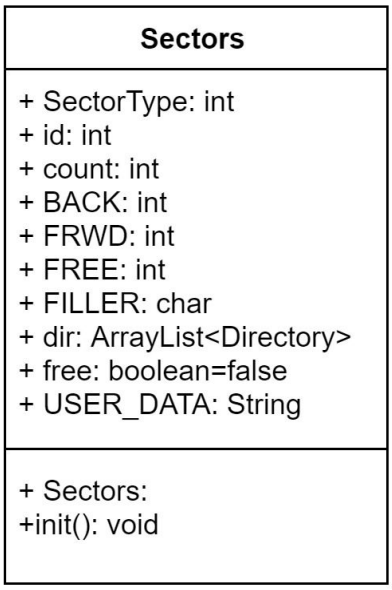
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
id	int		[0, 100]	Is the index of a directory with in a directory block.
TYPE	char	1	F, D, U	File type
NAME	String	100	Alpha-numeric	Name of a directory entry.
LINK	int		[0, 100]	Represents a sector pointer. → Point to either a directory block or data block.
SIZE	int		[0, 1024]	Command line argument string length
TOTALSIZE	int		[0,1024]	String length of data block.
state	int		{0, 1, 2}	Current state of data file.

4.100.1.4 Sectors

The following describes the Sectors class.

UML Class Diagram:



Identification: Sectors
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: The Sectors class implements Serializable in order to save its state.
The Sectors class serves as a parent class to DirectoryBlock & DataBlock.
Each Sector will either contain a directory block or a data block.
Subordinates: DirectoryBlock, DataBlock
Dependencies: Directory
Interfaces:

1

Interface: Sectors( )
Description: The Sectors constructor creates a sector that will contain either a directory block or data block a block.
Parameters:

Parameters	Range	Description
None		

Return: Not Applicable

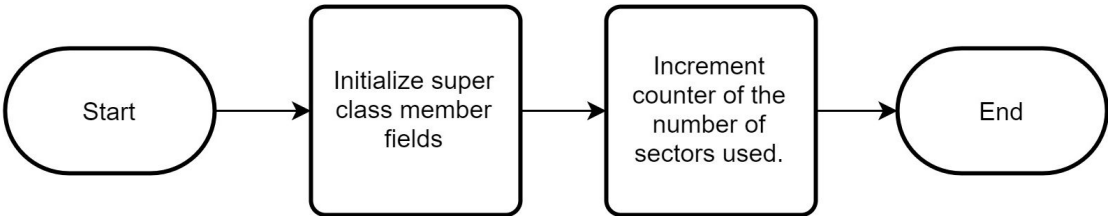
2

Interface: void init()
Description: The init method serves as a parent method.
Parameters:

Parameters	Range	Description
None		

Return: void

Resources: None
Processing:



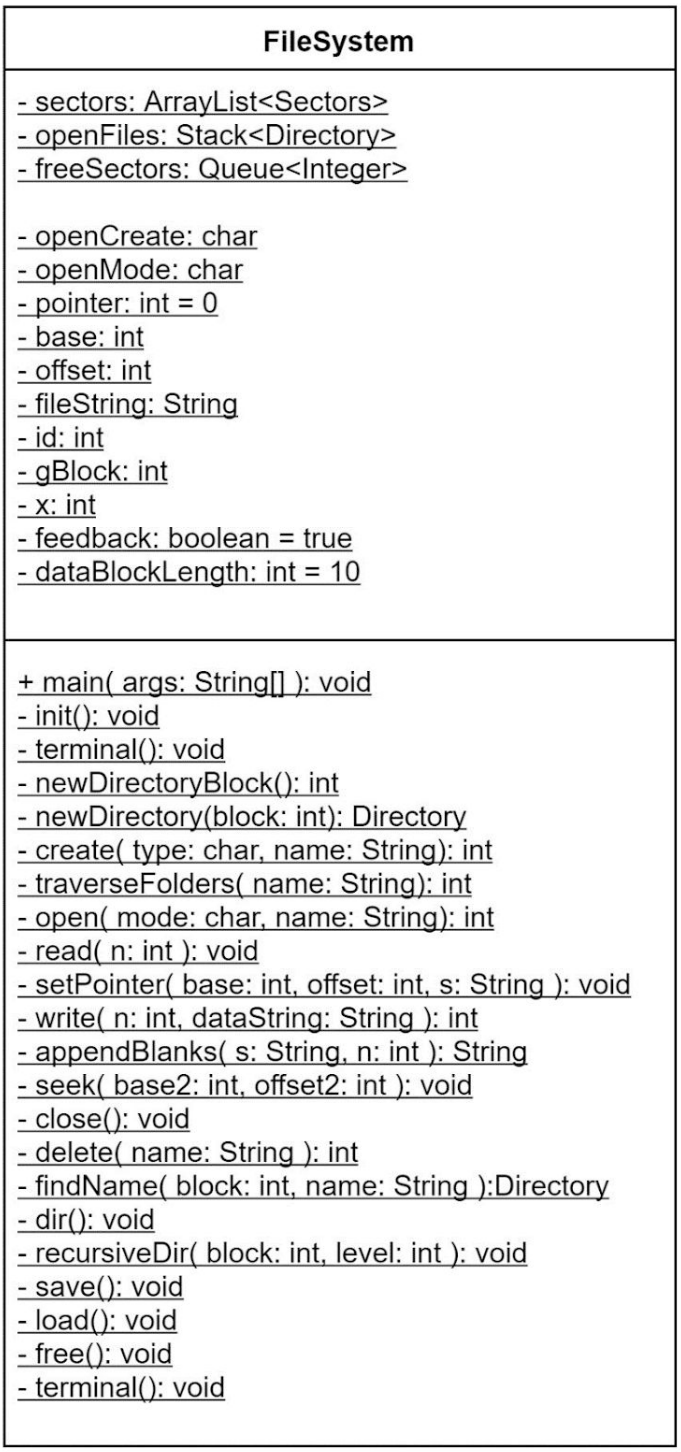
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
id	int		[0, 100]	Is the index of a sector within the sectors array list.
SectorType	int		{0, 1, 2}	0 = Free, 1 = Directory Block, 2 = Data Block
count	static int		[0, 100]	Number of Sectors in use.
BACK	int		[0, 100]	Back pointer for linked list of sectors.
FRWD	int		[0, 100]	Forward pointer for linked list of sectors.
FREE	int		[0,1024]	
FILLER	char	1	{.}	Used when writing n characters is greater than string to write.
dir	ArrayList <Directory >		Directory	As a DirectoryBlock, an array of Directories.
free	boolean		{True, False}	
USER_DATA	String	1024	Alphanumeric	As a DataBlock, i/o operations of user data.

4.100.1.5 FileSystem

The following describes the FileSystem class.

UML Class Diagram:



Identification: FileSystem
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.6, 4.100.9.7, 4.100.9.8
Function: The FileSystem class has a main where it initializes a file system then starts a user command line shell to simulate an Operating System.
The FileSystem class has a Sectors array to store memory.
The FileSystem class has a Directory Stack to keep track of open files.
Subordinates:
Dependencies: Sectors, Directory, Help
Interfaces:

1

Interface: public static void main(String parameters [])
Description: Where execution starts & calls two functions to commence the Operating System.
Parameters:

Parameters	Range	Description
parameters	Alphanumeric	Not used.

Return: void

2

Interface: private void init()
Description: The init() method initializes the file system.
Sets the FileSystem's first sector (block 0) as a directory block.
Parameters:

Parameters	Range	Description
None		

Return: void

3

Interface: private static int newDirectoryBlock()
Description: The newDirectoryBlock() method gets the next free sector id from the freeSectors array.
Creates a new sector as a Directory Block.
Parameters:

Parameters	Range	Description
None		

Return: int → Returns the id of this sector.

4

Interface: private static Directory newDirectory(int block)
Description: The newDirectory method searches in a Directory block for the first free entry.
Parameters:

Parameters	Range	Description

int block	[0, 100]	block is the selected Directory Block to start searching. If this block is full & has a forwarding block then searching will be re-referenced in this new block.

Return: Directory → The first free entry within a directory.

5

Interface: private static int create(char type, String name)

Description: The create method is invoked with the create command.
It is used to create either a directory file or a user data file.

Parameters:

Parameters	Range	Description
char type	{D, U}	Type is either D for Directory file or U for User data file.
String name	alphanumeric	name is the name of the file to be allocated.

Return: int → is the index of a Directory Block.

6

Interface: private static int traverseFolders(String name)

Description: The traverseFolders method is a non recursive method used to search for a given directory name.
Starts searching from root directory, drilling down to subfolders until folder not found (short circuit evaluation) or traversed to last subfolder.

Parameters:

Parameters	Range	Description
String name	alphanumeric	name is the value given for the absolute directory path to traverse.

Return:int → the id of the sector where a match is found or
if folder is not found then return -1.

7

Interface: private static int open(char mode, String name)

Description: The open method has three modes.
Input mode means that only READ and SEEK commands are permitted.
Output mode means only WRITE commands are permitted.
Update mode allows READ, WRITE, and SEEK commands.
Associated with each open file is a pointer to the next byte to be read or written.

Opening a file for input or update places the pointer at the first byte of the file, while opening a file for output places the pointer at the byte immediately after the last byte of the file.

Parameters:

Parameters	Range	Description
char mode	{I, O, U}	Represents Input, Output or Update.
String name	alphanumeric	is the name of the User data file to be opened.

Return:int → gives a status of the operation.
failure = -1, success = 1.

8

Interface: private static void read(int n)

Description: The read method is invoked when the read command is used.
This command may only be used between an OPEN (in input or update mode) and the corresponding CLOSE.
If possible, 'n' bytes of data should be read and displayed.
If fewer than 'n' bytes remain before the end of file, then those bytes should be read and displayed with a message indicating that the end of file was reached.

Parameters:

Parameters	Range	Description
int n	[0, 1024]	Represents the number of bytes to be read.

Return:void

9

Interface: private static void setPointer(int base, int offset,
String s)

Description: The setPointer method sets the file pointer to a new position in a file.

Parameters:

Parameters	Range	Description
int base	{-1, 0, 1}	Is the starting point from where an offset may be applied to a file pointer. -1 → Indicates the beginning of the file.

		0 → Indicates current position in the file. 1 → Indicates the end of the file.
int offset	[-1024, 1024]	Is a signed integer indicating the number of bytes from the 'base' that the file pointer should be moved.
String s	alphanumeric	

Return: void

10

Interface: private static int write(int n, String dataString)

Description: The write method is invoked by the write command.
The first 'n' data bytes of the 'dataString' is written to a file.
If the 'dataString' is less than 'n' bytes then a filler character is used, such as a period.

Parameters:

Parameters	Range	Description
int n	[0, 1024]	n is the number of bytes to write.
String dataString	alphanumeric	dataString is the user data to write

Return:int → Failure = -1 otherwise it will return the number of bytes written.

11

Interface: private static String appendBlanks(String s, int n)

Description: The appendBlanks method is called from the write method to append blanks at the end of the string in order to write n bytes.

Parameters:

Parameters	Range	Description
String s	alphanumeric	s is the String that will have blanks appended to.
int n	[0, 1024]	n is the length of s after appending blanks.

Return:String → String is the result of the given String after appending blanks.

12

Interface: private static void seek(int base2, int offset2)
Description: The seek method is invoked by the shell.
Performs input validation.
Calls the setPointer method to set the file pointer to a new position in a file.
Parameters:

Parameters	Range	Description
int base	alphanumeric	See setPointer
int offset	[0, 1024]	See setPointer

Return:void

13

Interface: private public static void close()
Description: The close method is invoke by the shell.
It pops the last open file off the stack.
Resets the state of the file to close.
Parameters:

Parameters	Range	Description
None		

Return:void

14

Interface: private static Directory findName(int block, String name)
Description: findName is a private method used to search for a name in a directory block.
Parameters:

Parameters	Range	Description
int block	[0, 100]	block the given directory block to search in.
String name	alphanumeric	name the given search term to match to a Directory entry.

Return:Directory → Directory is the found directory entry that matched the search criteria otherwise returns null.

15

Interface: private public static void dir()
Description: The dir method is invoked by the shell.
It lists the contents of the entire file system.
Parameters:

Parameters	Range	Description
None		

Return:void

16

Interface: private public static void save()
Description: The save method is invoked by the shell.
It saves the current state of the file system as a serialized file.
Parameters:

Parameters	Range	Description
None		

Return:void

17

Interface: private public static void save()
Description: The load() method is invoked by the shell.
It loads a serialized file to reset the file structure to its last saved state.
Parameters:

Parameters	Range	Description
None		

Return:void

18

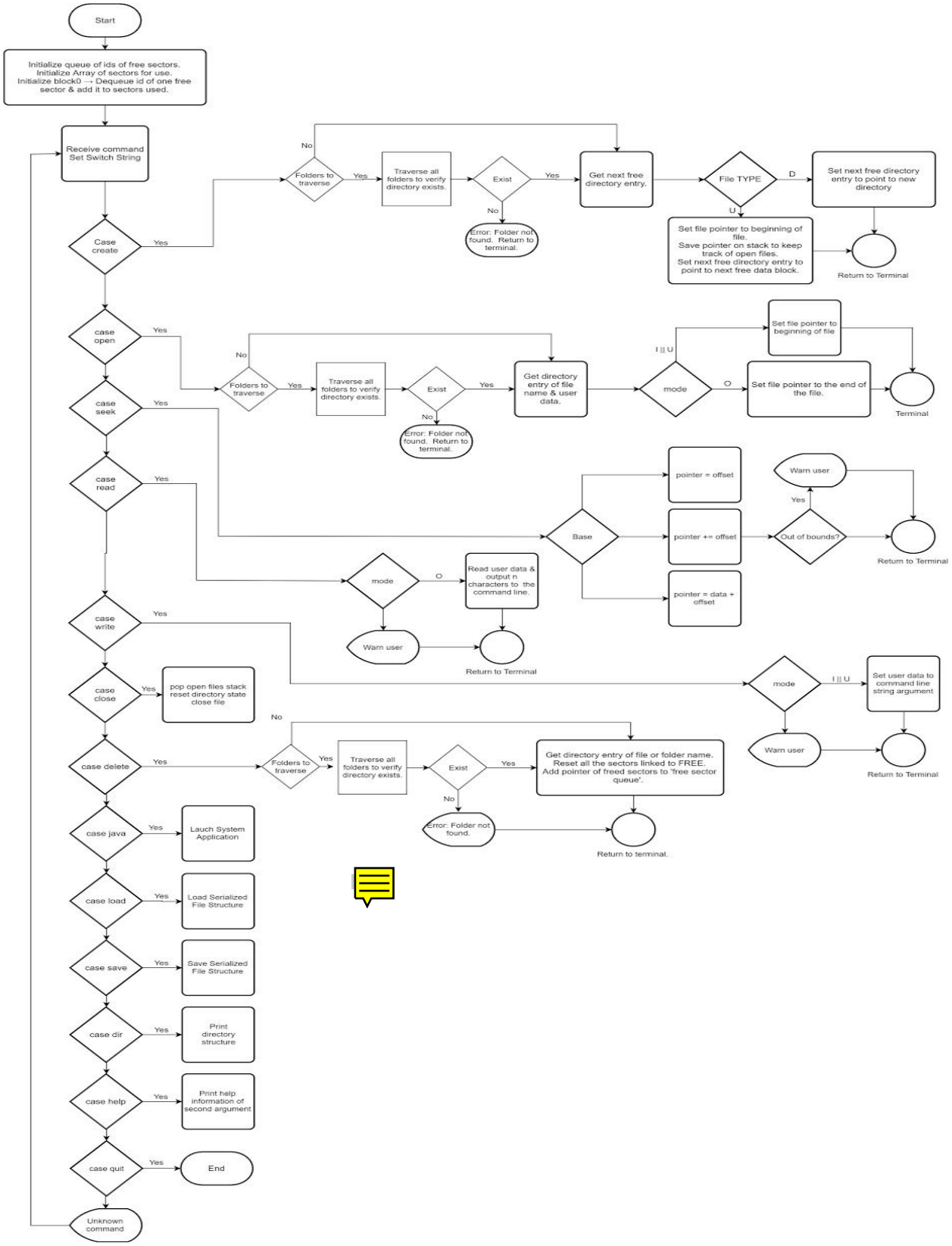
Interface: private static void shell()
Description: The terminal method is invoked by the main() method.
It simulates a shell.
Parameters:

Parameters	Range	Description
None		

Return:void

Resources: None

Processing:



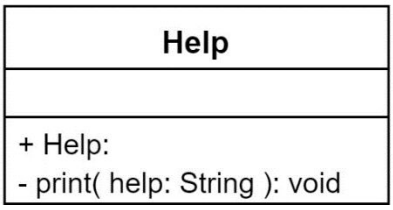
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
id	int		[0, 100]	Is the index of a sector within the sectors array list.
SectorType	int		{0, 1, 2}	0 = Free, 1 = Directory Block, 2 = Data Block
count	static int		[0, 100]	Number of Sectors in use.
BACK	int		[0, 100]	Back pointer for linked list of sectors.
FRWD	int		[0, 100]	Forward pointer for linked list of sectors.
FREE	int		[0,1024]	
FILLER	char	1	{.}	Used when writing n characters is greater than string to write.
dir	ArrayList <Directory >		Directory	As a DirectoryBlock, an array of Directories.
free	boolean		{True, False}	
USER_DATA	String	1024	Alphanumeric	As a DataBlock, i/o operations of user data.

4.100.1.6 Help

The following describes the Help class.

UML Class Diagram:



Identification: Help
Type: Class
Purpose: Satisfies SRS 2.6.100
Function: The Help class provides syntax & semantics for a given command.
Subordinates: None
Dependencies: None
Interfaces:

1

Interface: public Help(String help)
Description: The Help constructor creates an instance specified by the search term.
Parameters:

Parameters	Range	Description
String help	alphanumeric	help is a search term to filter results for a specific command.

Return:Not Applicable

2

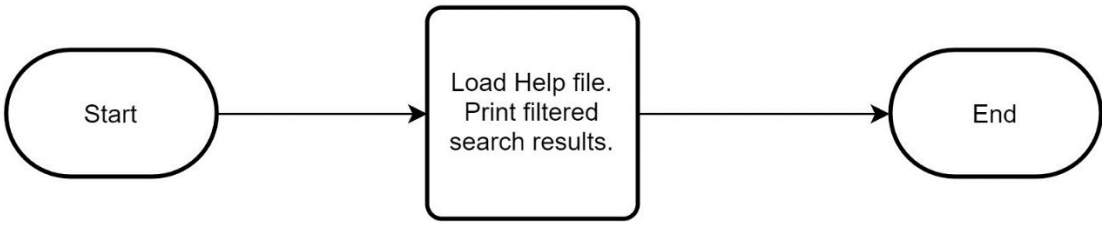
Interface: private static void print(String help)
Description: The print method is invoked by the constructor. It opens a help file & returns a filtered help result.
Parameters:

Parameters	Range	Description
String help	alphanumeric	help is a search term to filter results for a specific command.

Return:void

Resources: Help.txt → All data of commands, syntax, descriptions.

Processing:



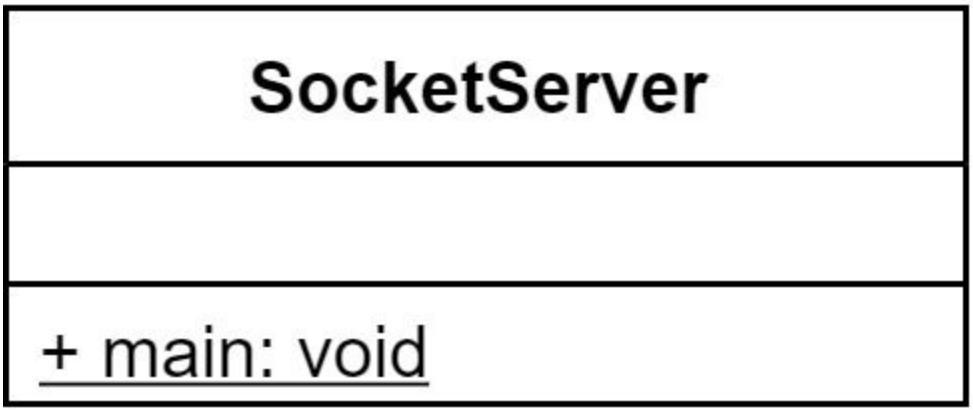
Data: None

4.100.2 Server Classes

4.100.2.1 SocketServer

The following describes the SocketServer class.

UML Class Diagram:



Identification: SocketServer
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: Serve multi users via its multi threaded socket server over the LAN or WAN. Waits for a request for a new socket connection, to create and start a terminal instance on a separate thread.
Subordinates: None
Dependencies: Terminal
Interfaces:

1

Interface: public static void main(String args[])
Description: This is the starting point of execution when the OS is in a multi user mode.
Parameters:

Parameters	Range	Description
String args[]	alphanumeric	Not used

Return:Not Applicable

Resources: Dell XPS 8930-7814BLK-PUS Tower Desktop i7-8700 32GB DDR4 RAM, 1TB Hard Drive + 16GB Intel Optane Memory, 6GB Nvidia GeForce GTX 1060, DVD Burner, Windows 10 Pro, Black
XAMPP, MySQL database

Processing:

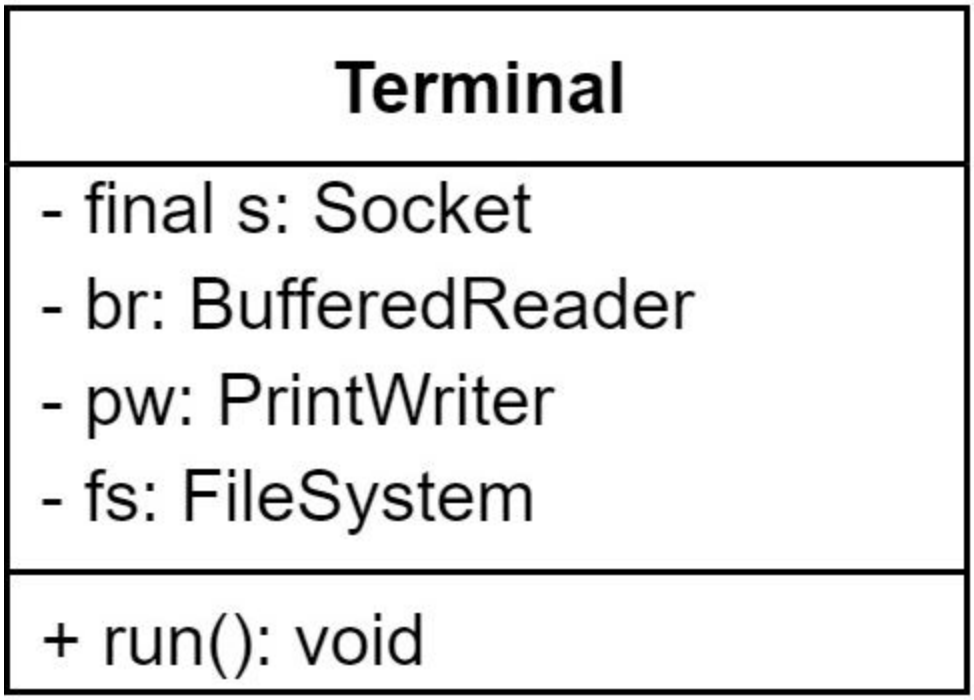
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
None				

4.100.2.2 Terminal

The following describes the Terminal class.

UML Class Diagram:



Identification:	Terminal
Type:	Class
Purpose:	Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function:	To receive the socket connection from the SocketServer class. Continue the socket connection. Receive commands from the client and relay those commands to an instance of the FileSystem class. Forward the response of the FileSystem class to the client via the socket connection.
Subordinates:	Thread
Dependencies:	None

Interfaces:

1

Interface: public Terminal(Socket s, BufferedReader br, PrintWriter pw)
Description:

Parameters:

Parameters	Range	Description
Socket s	Object	s is the object that references the socket server connection.

Return:Not Applicable

2

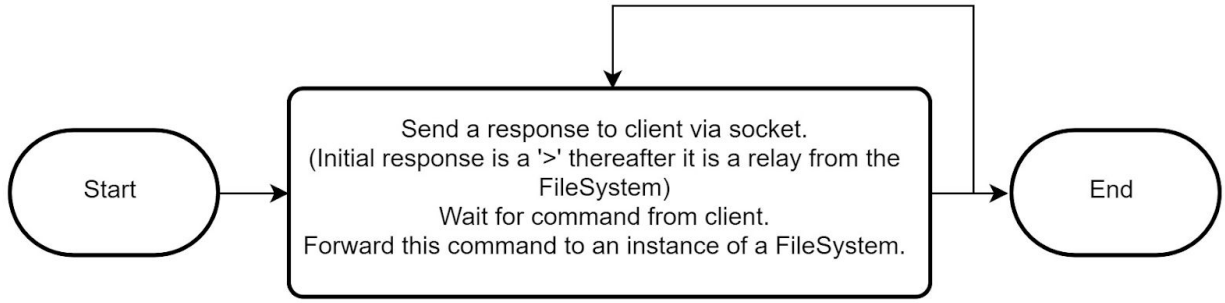
Interface: public void run( )
Description:

Parameters:

Parameters	Range	Description
None		

Return:Not Applicable

Resources: None
Processing:



Data: There are no member fields that accept a null value.

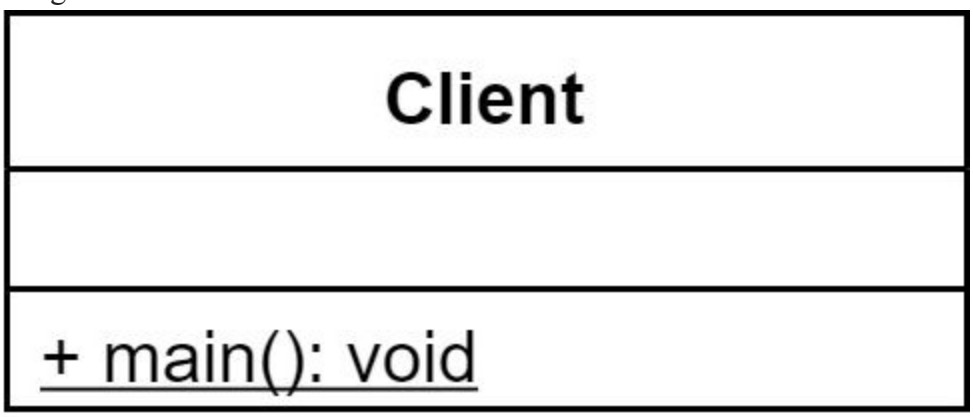
Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
s	Socket	n/a	n/a	s is the object that references the server socket connection.
br	BufferedReader	n/a	n/a	br is the object of the BufferedReader Class.
pw	PrintWriter	n/a	n/a	pw is the object of the PrintWriter class.
fs	FileSystem	n/a	n/a	

4.100.3 Client class

4.100.3.1 Client

The following describes the Client class.

UML Class Diagram:



Identification: Client
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: To request a socket connection from the SocketServer class.
Receive input from a user & send said input to the Server.
Receive responses from the server & print these messages for the user.
Subordinates: None
Dependencies: None
Interfaces:

1

Interface: public static void main( )
Description:

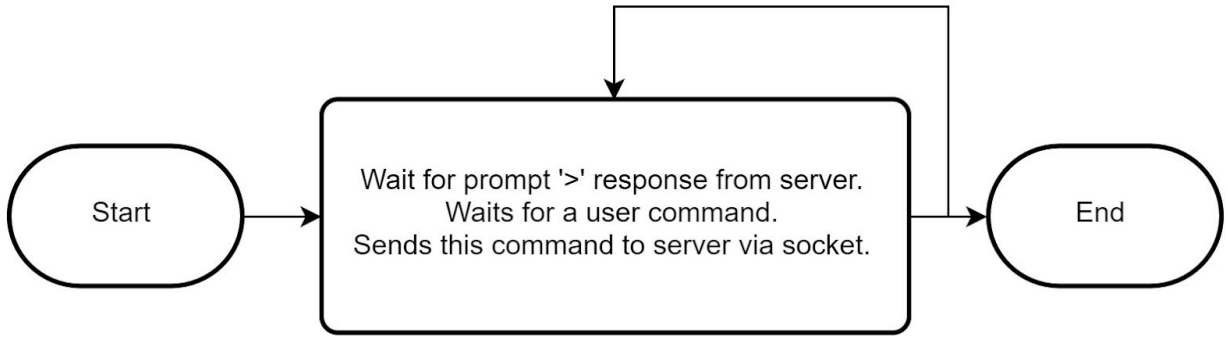
Parameters:

Parameters	Range	Description
None		

Return: Not Applicable

Resources: None

Processing:



Data: There are no member fields that accept a null value.

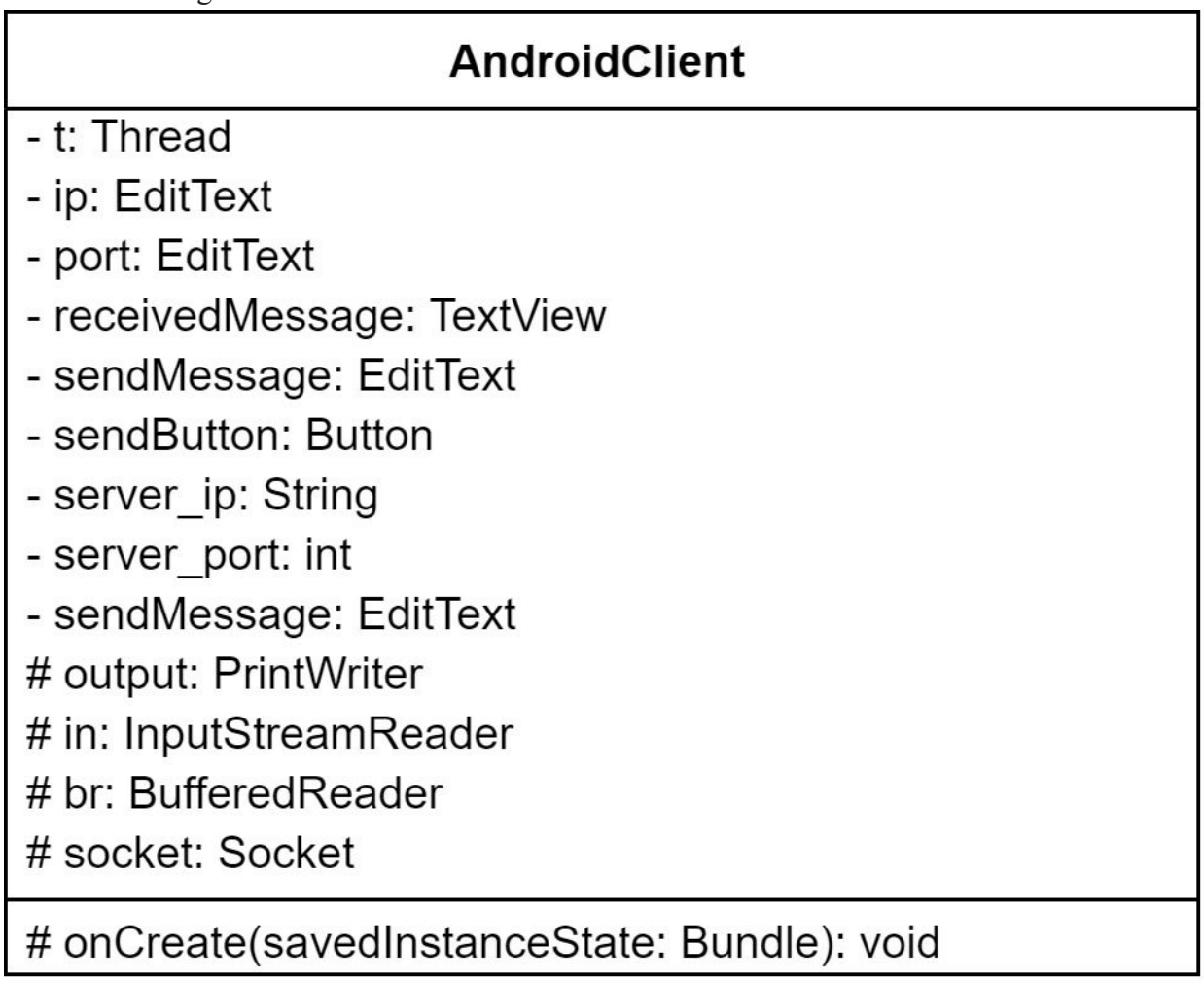
Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
s	Socket	n/a	alphanumeric	The object that references the server socket connection.
br	BufferedReader	n/a	n/a	Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.
pw	PrintWriter	n/a	n/a	Prints formatted representations of objects to a text-output stream.
fs	FileSystem	n/a	n/a	Creates a file structure for the OS.

4.100.4 Android Client Classes

4.100.4.1 AndroidClient

The following describes the Android Client class. These classes enable a user to log in from a remote location to the OS. With few modifications, the clients GPS coordinates can be automatically logged to the OS. This data can be accessed by others detailing a route, where their vehicle is parked, current location, when they will return to vehicle, destination.

UML Class Diagram:



Identification: AndroidClient
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: To request a socket connection from the SocketServer class.
Receive input from a user & send these messages to the Server.
Receive responses from the server & print these messages for the user.
Subordinates: None
Dependencies: Thread1

Interfaces:

1

Interface: public void onCreate(Bundle savedInstanceState)

Description: This method creates the user interface from the activity_main resource.

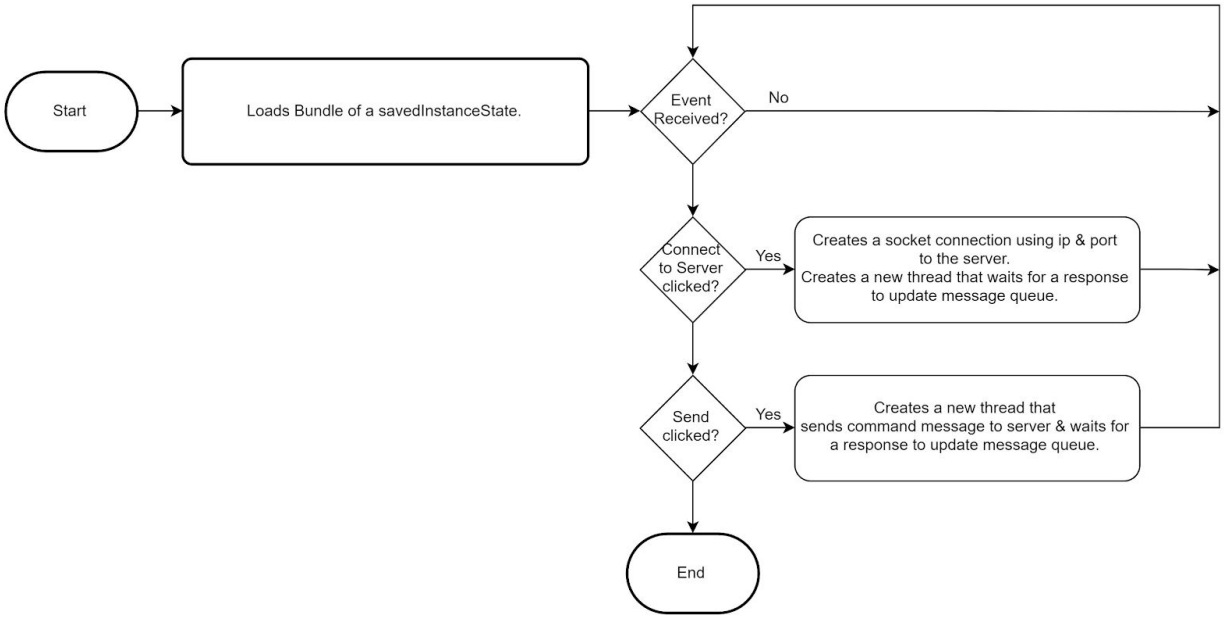
Parameters:

Parameters	Range	Description
Bundle savedInstanceState		Receives a reference to the Bundle object.

Return:void

Resources: None

Processing:



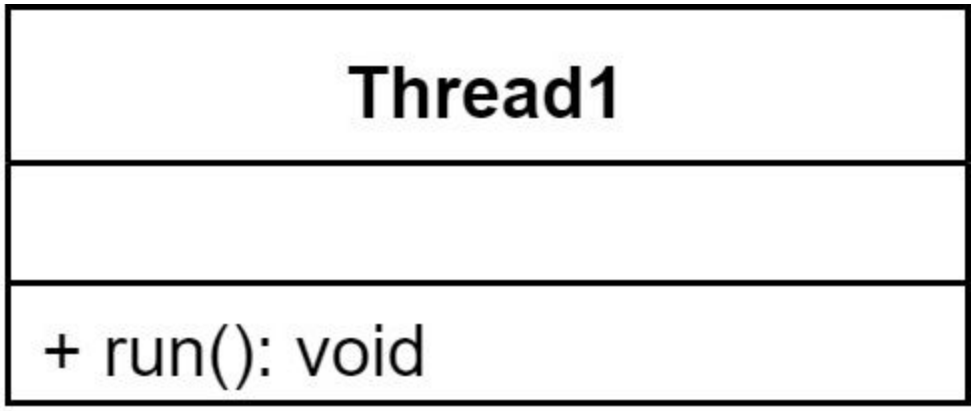
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Accepts null value?	Description
t	Thread		alphanumeric	No	The object that references the server socket connection.
ip	EditText			No	
port	EditText			No	
receivedMessage	TextView				
sendMessage	EditText				
sendButton	Button				
server_ip	String				
server_port	int				
sendMessage	EditText				
output	PrintWriter				
in	InputStreamReader				
br	BufferedReader				
socket	Socket				

4.100.4.2 Thread1

The following describes the Android Client Thread1 class.

UML Class Diagram:



Identification: Thread1
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: To request a socket connection from the SocketServer class.
Receive input from a user & send these messages to the Server.
Receive responses from the server & print these messages for the user.
This class provides these functions on a separate thread.
Subordinates: None
Dependencies: None
Interfaces:

1

Interface: public void run()
Description: This is an overridden method that implements Runnable.
Parameters:

Parameters	Range	Description
None		

Return:void

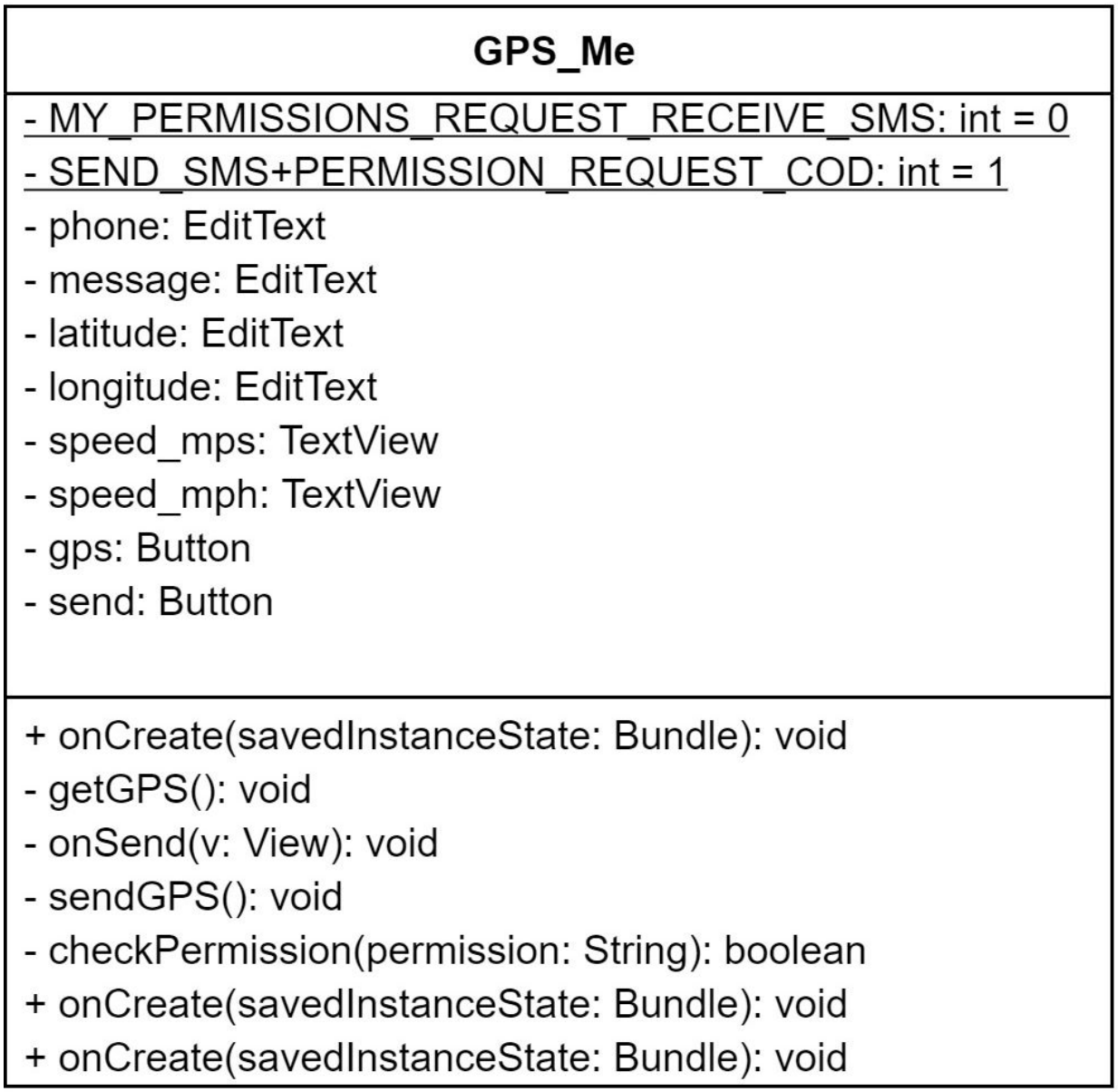
Resources: None
Processing: See SDS 4.100.11
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
None				

4.100.4.3 GPS_Me

The following describes the GPS_Me class this is the entry point class.

UML Class Diagram:



Identification: GPS_Me
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5,

4.100.9.7, 4.100.9.8

Function: When a user starts this application, this is the entry point of execution.
Shows a receiving text message.
Shows the receiving text number.
Gets GPS coordinates.
Sends GPS coordinates to a SMS text number.

Subordinates: None

Dependencies: GPS_Tracking

Interfaces:

1

Interface: protected onCreate(Bundle savedInstanceState)

Description: This method creates the user interface from the activity_main resource.

Parameters:

Parameters	Range	Description
Bundle savedInstanceState		Receives a reference to the Bundle object.

Return:void

2

Interface: private void getGPS()

Description: This method has an instance of the GPS_Tracking class, which returns the GPS coordinates & speed.

Parameters:

Parameters	Range	Description

Return:void

3

Interface: public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults)

Description: This interface method is the contract for receiving the results for permission requests.

Parameters:

Parameters	Range	Description
int requestCode		int: The request code for the permission requesting.
String permissions[]		The requested permissions.
int[] grantResults		The grant results for the corresponding permissions which is either PERMISSION_GRA

		NTED or PERMISSION_DENIED. Never null.

Return:void

4

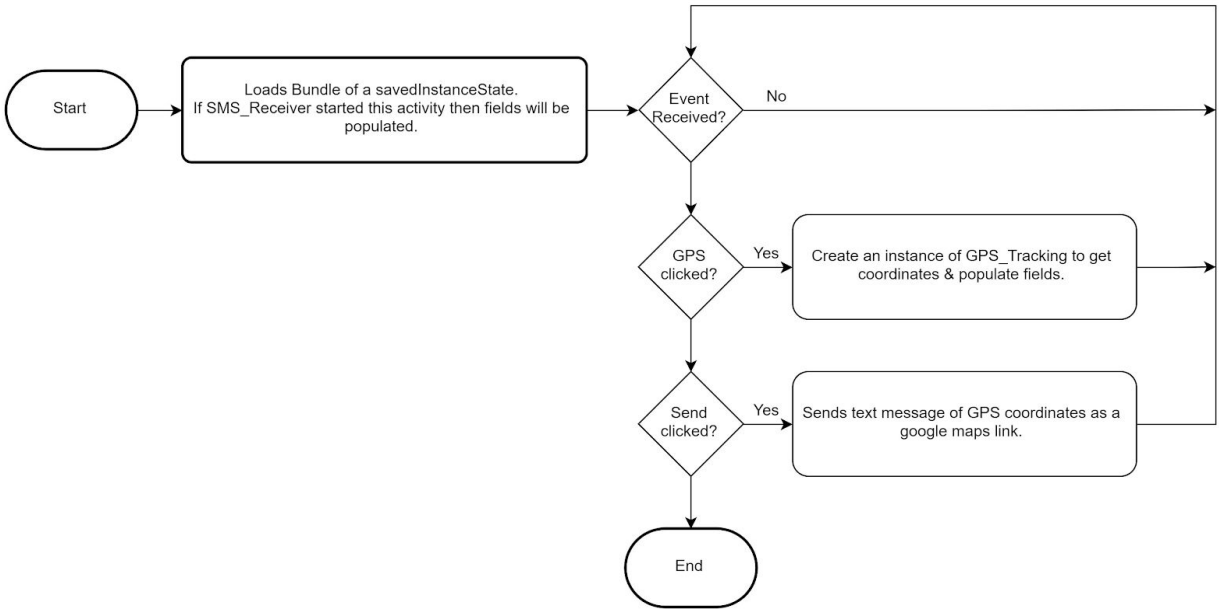
Interface: private void sendGPS()
Description: This method sends the currently listed GPS coordinates to the listed number. As a google maps link.

Parameters:

Parameters	Range	Description

Return:void

Resources: None
Processing:



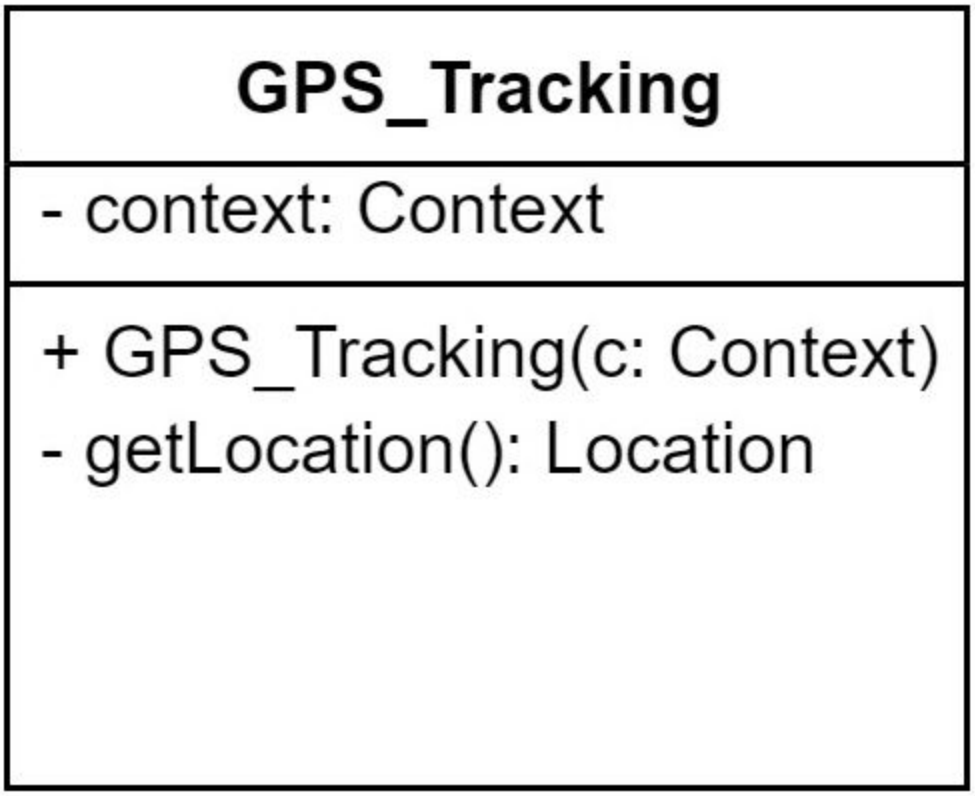
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
MY_PERMISSIONS_REQUEST_RECEIVE_SMS	int = 0		0	Request Permissions to receive SMS text messages.
SEND_SMS_PERMISSION_REQUEST_CODE	int = 1		1	Request Permissions to send SMS text messages.
phone	EditText			Enter phone number to send or is automatically filled when receiving SMS text message.
message	EditText			Initiating message received
latitude	EditText			GPS coordinate latitude of user.
longitude	EditText			GPS coordinate longitude of user.
speed_mps	TextView			GPS speed in meters per second
speed_mph	TextView			GPS speed in miles per hour.
gps	Button			Button to get current GPS coordinates
send	Button			Button to send GPS coordinates to the EditText phone.

4.100.4.4 GPS_Tracking

The following describes the GPS_Tracking class that requests location permissions & retrieves GPS coordinates.

UML Class Diagram:



Identification: GPS_Tracking
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: To get the current GPS coordinates, speed, distanceTo another location.
Subordinates: None
Dependencies: None
Interfaces:

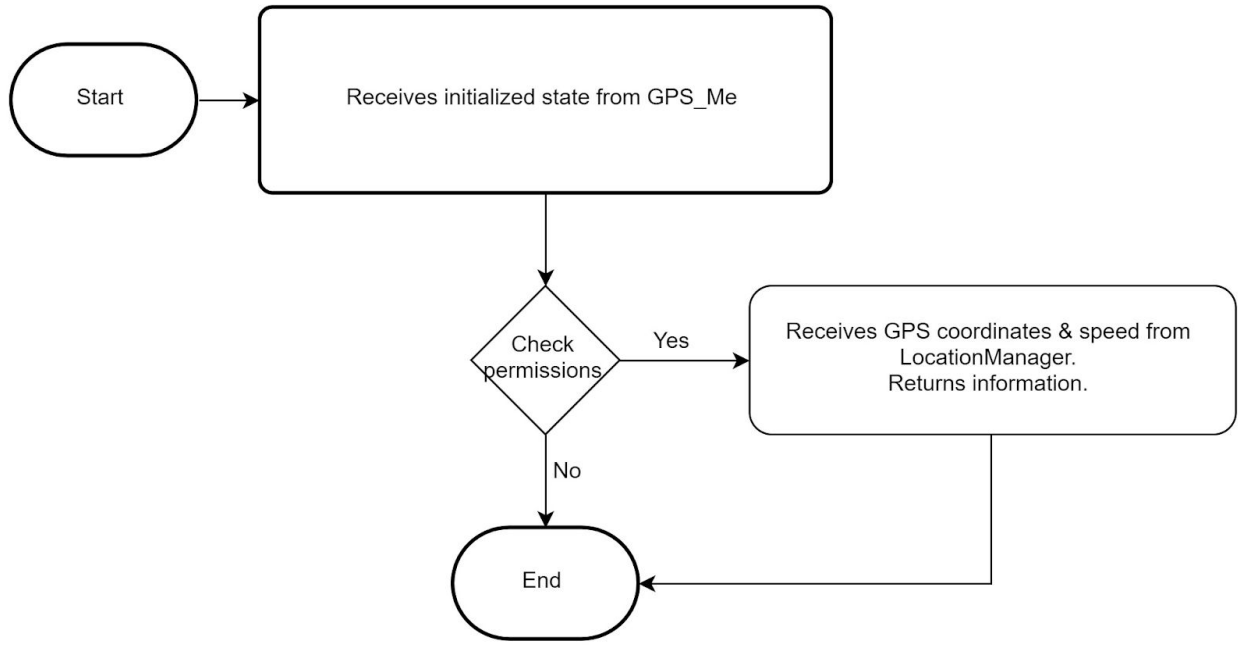
1

Interface: public void run()
Description: This method implements LocationListener.
Parameters:

Parameters	Range	Description
None		

Return:void

Resources: None
Processing:



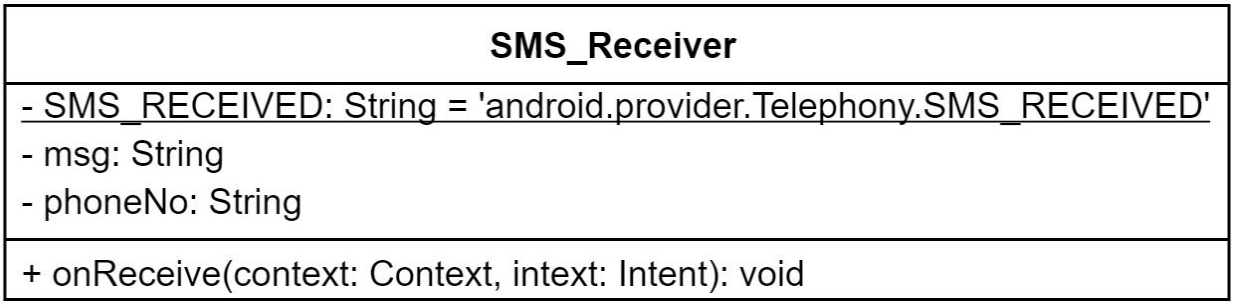
Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
context	Context			Interface to global information about an application environment.

4.100.4.5 SMS_Receiver

The following describes the SMS_Receiver class which requests Telephony permissions for receiving & sending text messages.

UML Class Diagram:



Identification: SMS_Receiver
Type: Class
Purpose: Satisfies SRS 4.100.9.1, 4.100.9.2, 4.100.9.3, 4.100.9.4, 4.100.9.5, 4.100.9.7, 4.100.9.8
Function: This class extends BroadcastReceiver which has a service that starts the onReceive method when ever there is an incoming text message.
This start service, starts execution even when the application Activity is shut down.
Subordinates: None
Dependencies: None
Interfaces:

1

Interface: public void onReceive(Context context, Intent intent)
Description: When this application is in a Activity Shutdown state, BroadcastReceiver enables this method to be the executing entry point.
It retrieves all incoming text messages.
Creates an intent that starts GPS_Me.
Parameters:

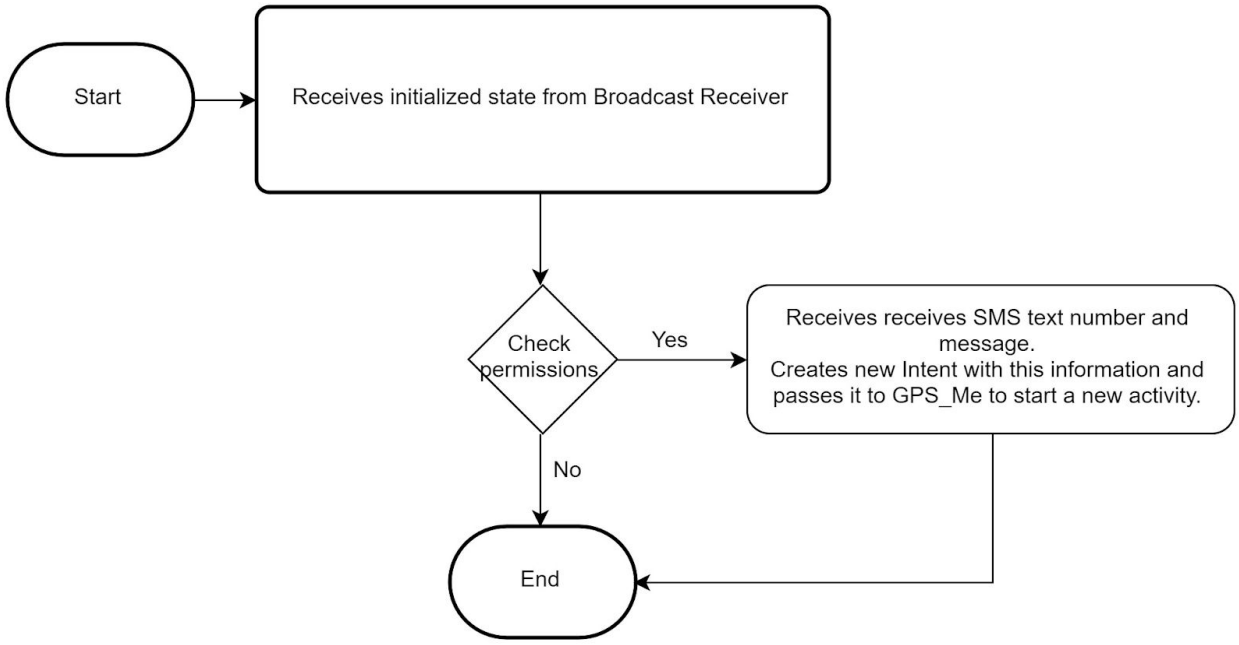
Parameters	Range	Description
Context context		Interface to global information about an application environment.
Intent intext		is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers and services. Intents are also

		used to transfer data between activities.

Return:void

Resources: None

Processing:

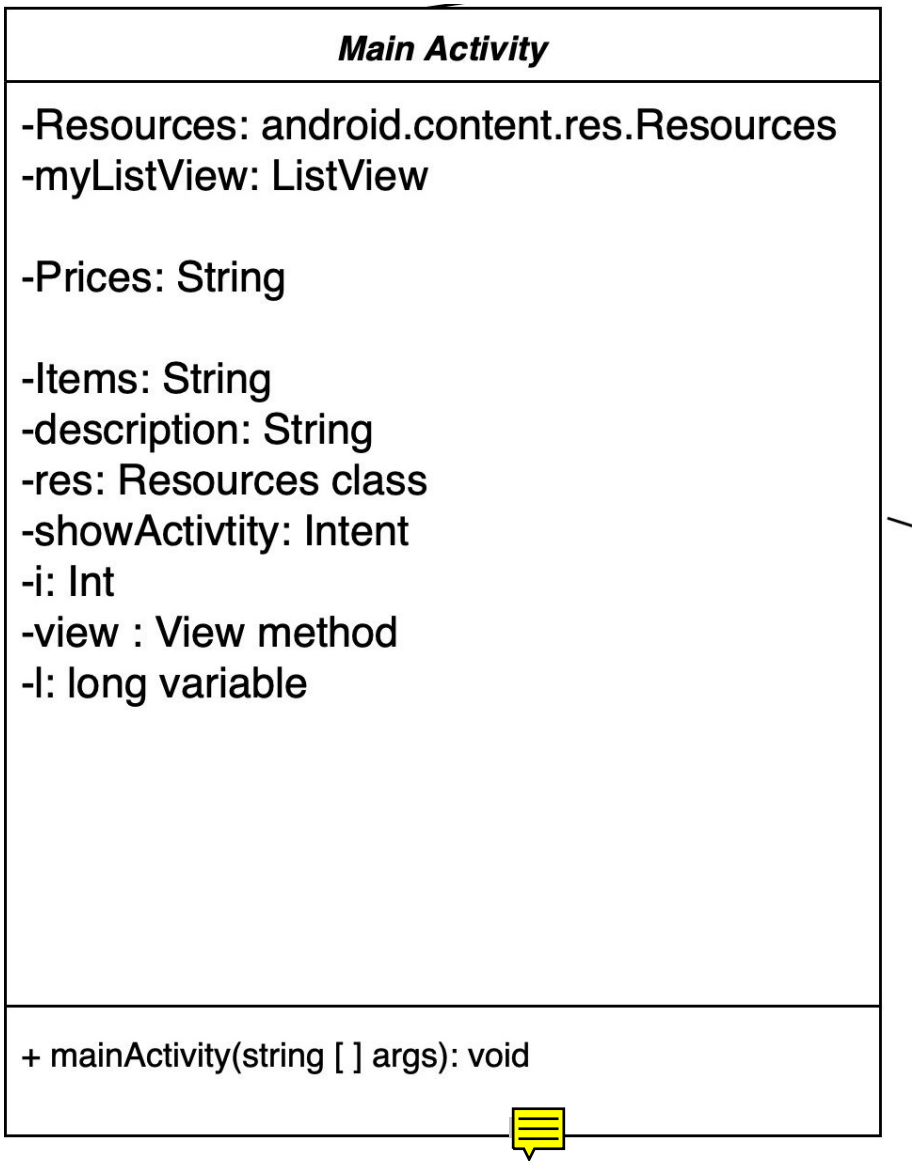


Data: There are no member fields that accept a null value.

Name	Data Type	Character Length (if applicable)	Acceptable Values	Description
phone	String		Integers	Phone number of receiving SMS text message or number that GPS will be sent to.
message	String		alphanumeric	The incoming message.

4.300.1 Main Activity

4.300.1UML Class Diagram:



Identification: Main Activity

Type: Main Class

Purpose: 4.300

Function: Implement the shopping cart application.

Subordinates: DetailActivity, ImageActivity, ItemAdapter

Dependencies: None

Interface: Main Activity from Android application

Description: Main Activity manage ItemAdapter and store the picture data into ImagaActivty file. The Main Activity class represents shopping cart apps interface. You can look the shopping interface.

Parameters	Range	Description
Intent method		Intent method is that if user click button, this intent function will implement the other file system. And this method will go to another class.

Return: Not Applicable

Parameters:

Parameters	Range	Description
None		

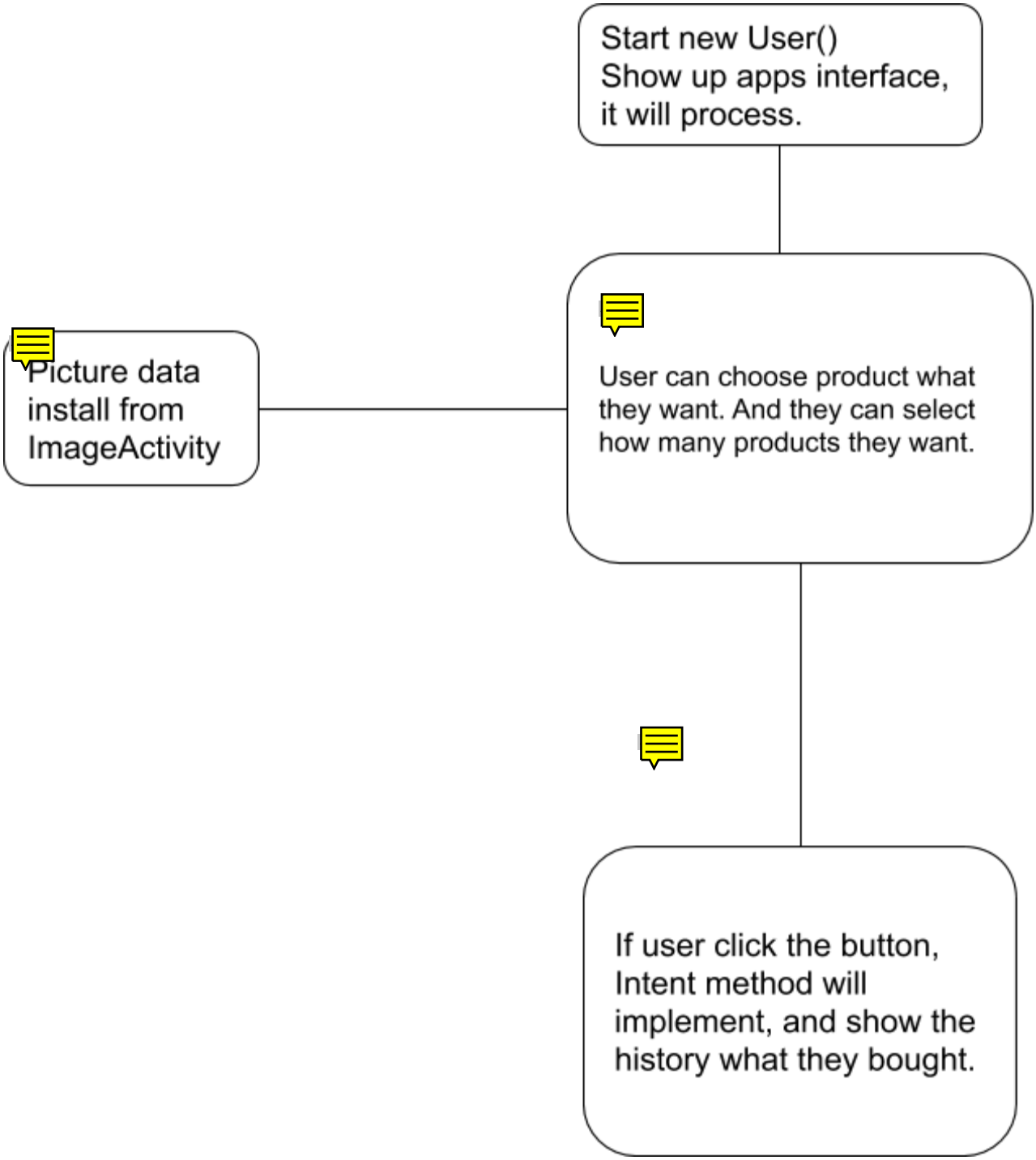
Return: void

Resources:

ListView, String variable :items :prices :description ,+Resources, +myListView ,+ItemAdapter and +Intent

Processing:

When user clicked button, this main activity will make an order for the other file, and it will show up the pictures.

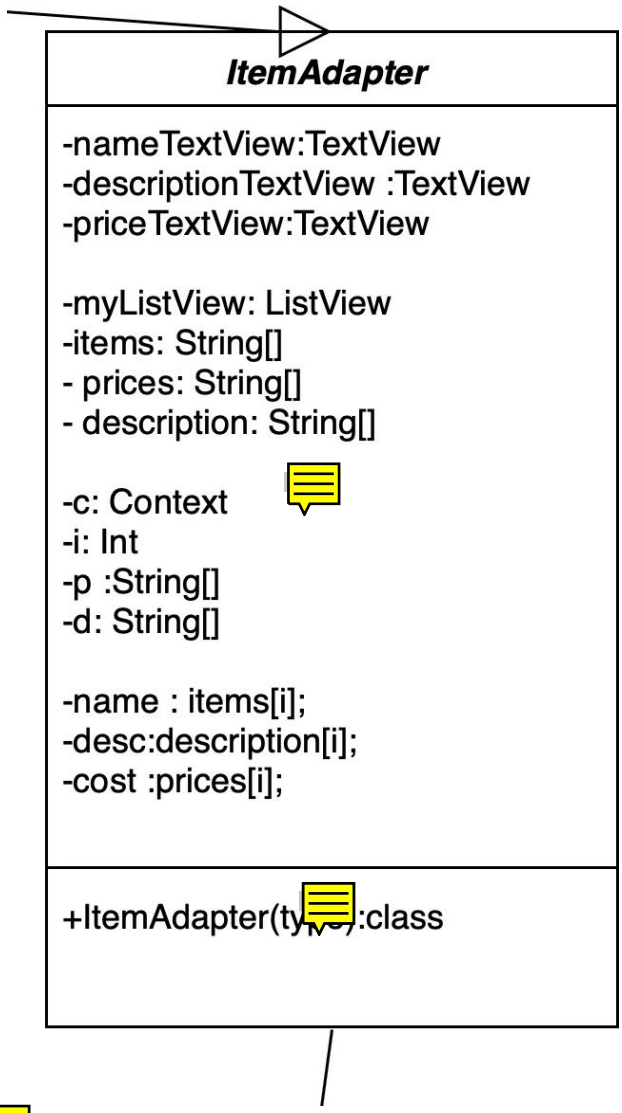


Data: When user clicked button, this main activity will make an order for the other file, and it will show up the pictures.

Name	Data Type	Character Length	Acceptable Values	Description
Main Activity	Class	Unknown		

4.300.2 ItemAdapter

UML Class Diagram:



Identification: ItemAdapter
Type: Class
Purpose: When User click on, this class will start preparing showing the product pictures.
Function: ItemAdapter class implements to save the picture data and arrange the content of apps.

Subordinates: ImageActivity
Dependencies: Main Activity
Interfaces:

1

Interface: ItemAdapter( )

Description:

Parameters:

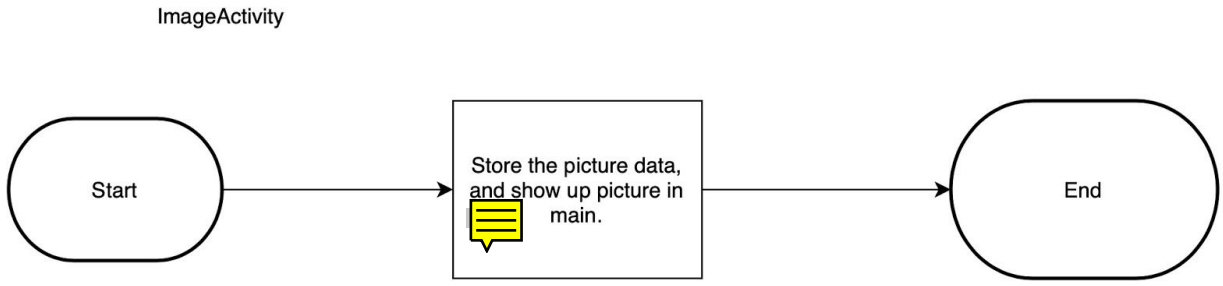
Parameters	Range	Description
None		

Return: Not Applicable

Resources: N

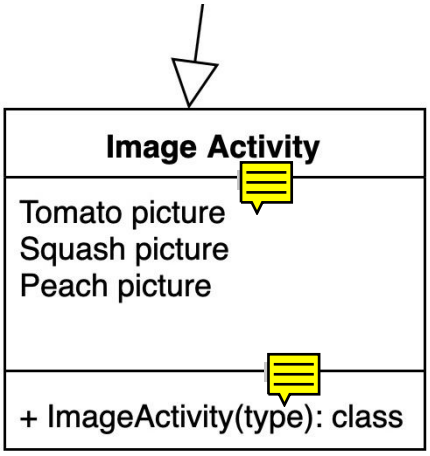
Data: None

Processing:



4.300.3 Image Activity

UML Class Diagram:



Identification: ImageActivity
Type: Class
Purpose: Store the picture data, and if ItemAdapter call , Image Activity will provide picture data.
Function: ImageActivity will store the picture data.This file function is very simple.Hold picture data. And if user click button, this imageActivity will respond to show the pictures.

Subordinates: None
Dependencies: Main Activity, ItemAdapter
Interfaces:

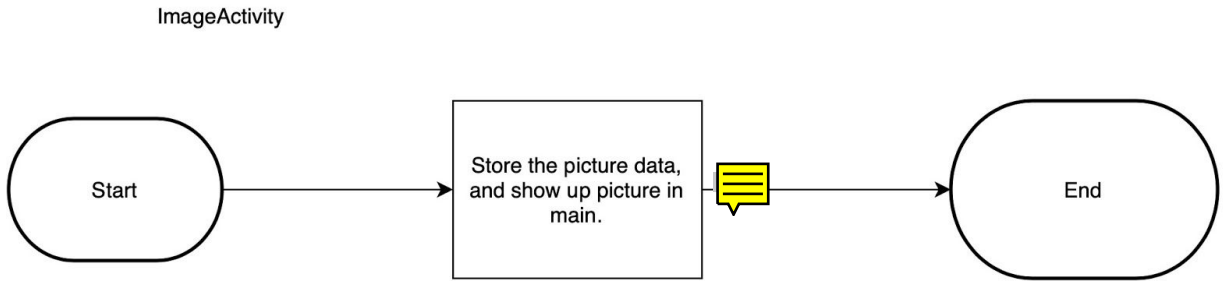
Interface: ImageActivity()
Description:

Parameters:

Parameters	Range	Description
None		

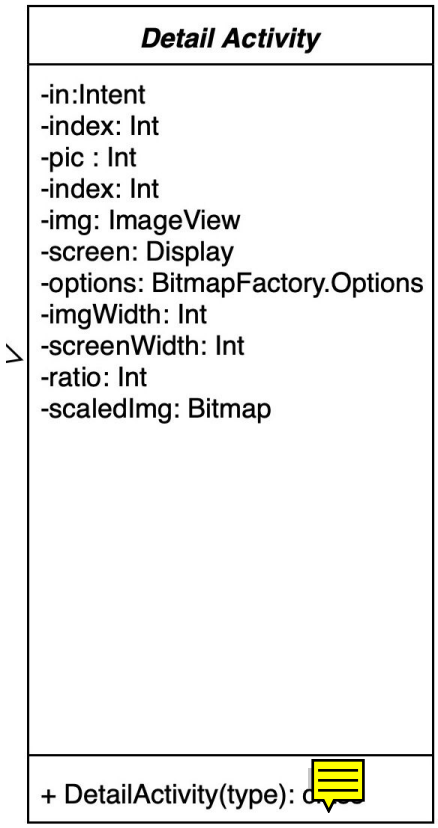
Return: Not Applicable

Resources: None
Data: Peach,squash and tomato picture data
Processing:



4.300.4 Detail Activity

UML Diagram



Identification: Detail Activity

Type: Class

Purpose: This Detail Activity will judge what user choose the product

Function: This Detail activity will judge the product. It means that if user selected any product, this detail activity will find the product, and it transfer the ItemAdapter file.

Subordinates: ImageActivity

Dependencies: None

Interface: Detail Activity

Description: The reason why I created Detail activity is that if user clicked button, program needs to show up product picture and price. Therefore, I use a switch statement, and select what picture and price I need to show up.

Return: Not Applicable

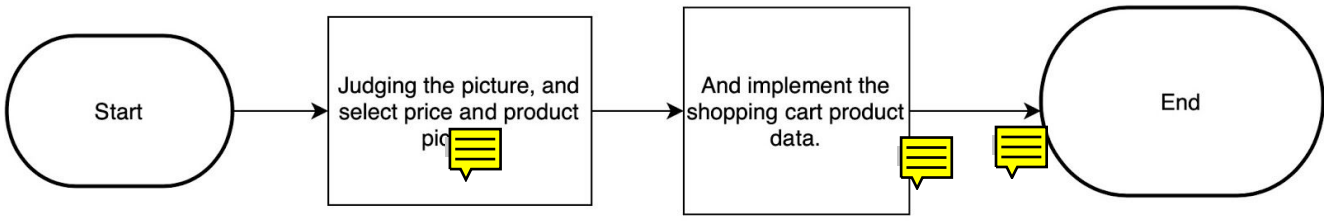
Parameters:

Return: void

Resources: None

Processing:

Detail Activity



Data: When user clicked button, this main activity will make an order for the other file, and it will show up the pictures. This detail Activity are judging the product and price using if statement and switch statement.

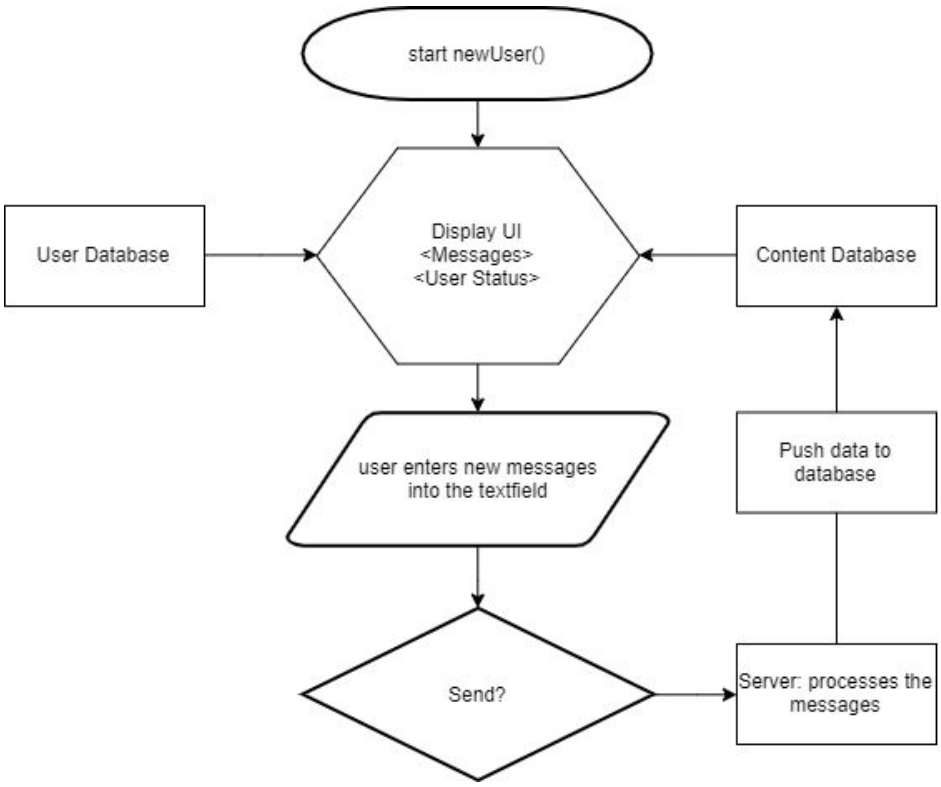
4.400 The Messenger Application classes

4.400.1 Messenger - Client View class

UML class diagram:

Identification:	Client View
Type:	A Java function
Purpose:	Giving the user a Graphical User Interface to interact with the application
Function:	Send/Receive Messages to/from the server.
Subordinates:	None.
Dependencies:	Authentication module, Server Controller,User Database, Content Database.
Interfaces:	None.
Resources:	The GUI in this module will generate a window that accounts for the minimum of 500x500 pixels on the screen, can minimize the Messenger App UI or expands the UI to full screen.

Processing:

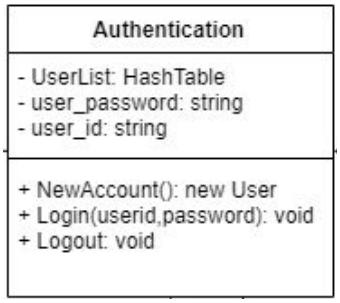


Data: Continuously fetching data from the server's database every 2 seconds.
Including data from <ChannelName.ChannelData>
Ref to Package View 3.400

Name	Data Type	Range	Description
message	String	Alpha-numeric	Store user message
serverIP	String	Numeric	Store server IP address
output	ObjectOutputStream	Alpha-numeric	Responsible for user outgoing data to the server
input	ObjectInputStream	Alpha-numeric	Responsible for server ingoing data to the client view

4.400.2 Messenger - Authentication Class

UML class diagram:



Identification: Authentication

Type: A Java function

Purpose: Giving the application the ability to manage the user account.

Function: Manage the user account, allow or deny access to the main Server.

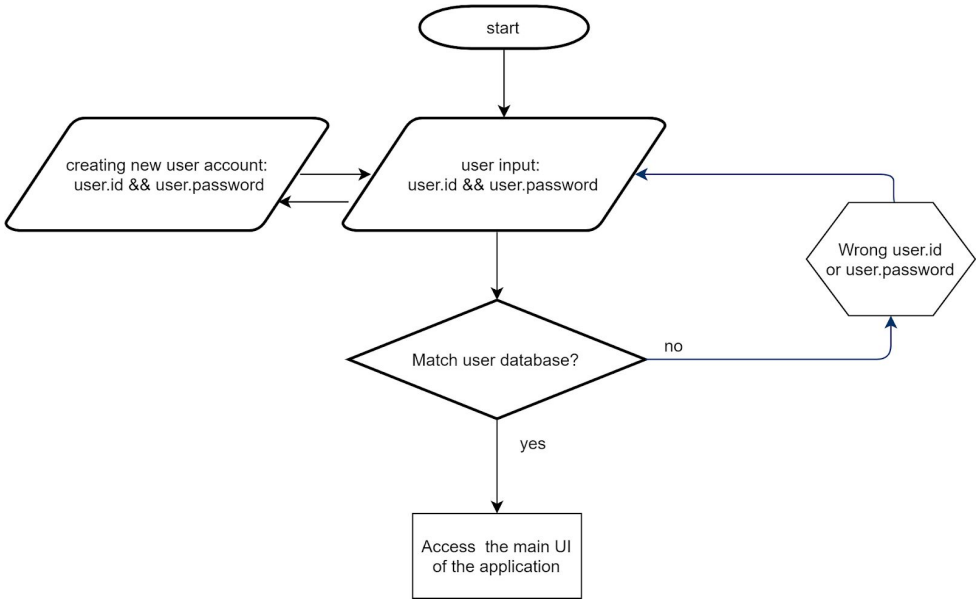
Subordinates: None.

Dependencies: User Database.

Interfaces: None.

Resources: None.

Processing: This function should be called to check with the user database to see if the user enters a valid account ID and password. This function is repeated until the user successfully enters a valid account ID and password.

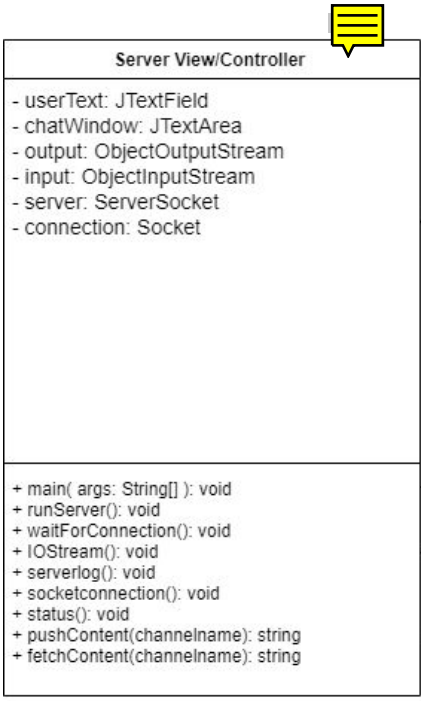


Data: User database.
+ UserList: Hashtable(key, user)
+ userid: String
+ userid_password: String

Name	Data Type	Range	Description
userlist	Hashtable	[1..1000] Users	Fetching user list f user database
user_password	String	Alpha-numeric	User input password
user_id	String	Alpha-numeric	User input ID

4.400.3 Messenger - Server View/ Controller class:

UML class diagram:



Identification: Server View/Controller

Type: Java function

Purpose: A central controller which manipulate and process user input and data.

Function: Provide an administrative server managing tool.
Fetch data from the Content Database to the Client view or Push the user input data to the Content Database.

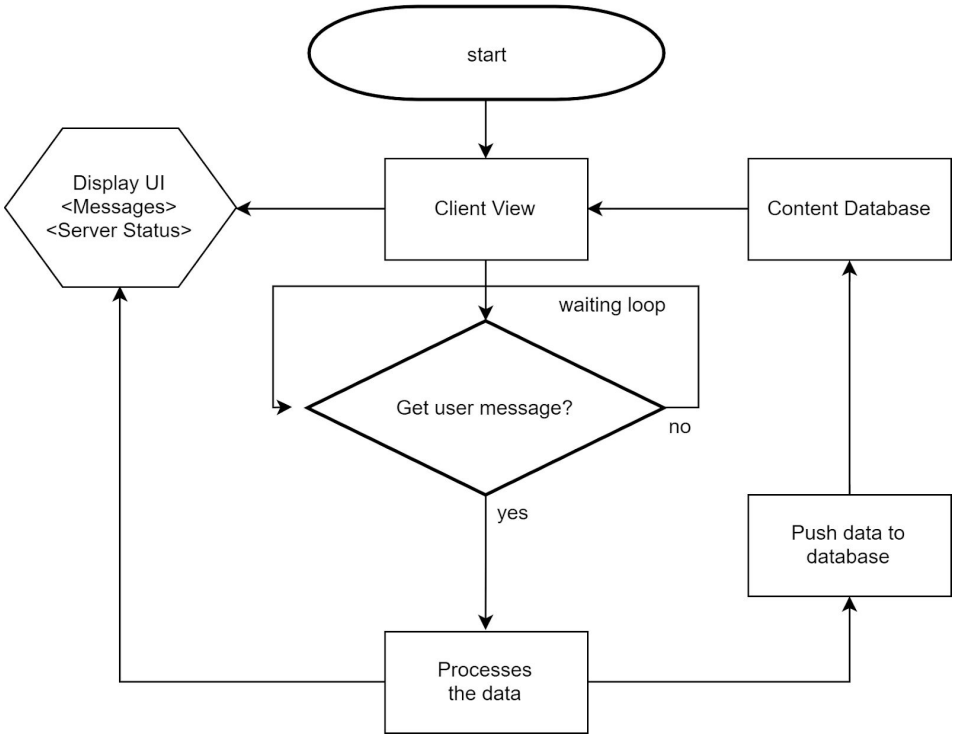
Subordinates: None.

Dependencies: Content Database, Client View.

Interfaces: None.

Resources: MongoDB.
The following dedicated server will be used to run this server view/controller package:
ABS Logic R730 / CPU: AMD FX8300@3.3GHz / RAM: 8GB / Hard Drive Disk: 1TB / Graphic Card: GeForce GT 730 / Internet: Download/ Upload Speed: 250/15 Mbps

Processing: This function should be called and run simultaneously with Client View. This function should be running until the administrator exits the main program.

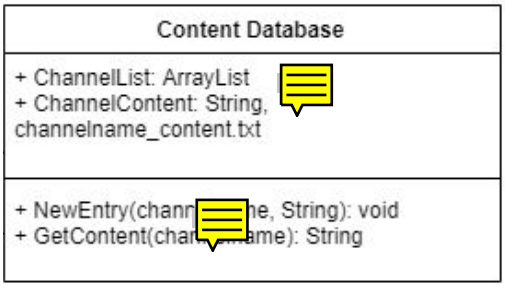


Data: User Input Data (Including data such as User Messages, User Action and Timestamps) and Text Data from the Content Database.
Including data from <ChannelName.ChannelData> , <userview.input> <userview.sendmessages>.
Ref to Package View 3.400

Name	Data Type	Range	Description
output	ObjectOutputStream	Alpha-numeric	Responsible for outgoing data to the Client View.
input	ObjectInputStream	Alpha-numeric	Responsible for ingoing data from the Client view.
server	ServerSocket	Number[1..99999]	Stores server's socket
connection	Socket	Number[1..99999]	Stores client's socket

4.400.4 Messenger - Content Database class

UML class diagram:



Identification: Content Database

Type: A Java class

Purpose: Store text data from user input (Messages, Html Links, Timestamps…)

Function: Transfer data in and out the module to server view/controller module per request.

Subordinates: None.

Dependencies: Server view/controller.

Interfaces: None.

Resources: The following dedicated server will be used to store this content database
ckage:
ABS Logic R730 / CPU: AMD FX8300@3.3GHz / RAM: 8GB / Hard Drive Disk: 1TB / Graphic Card: GeForce GT 730 / Internet: (Download/ Upload Speed: 250/15 Mbps)

Processing: Accepting text data from user input (Messages, Html Links, Timestamps…) from the Server View/Controller Module then append said data into corresponding text channel.
Fetching text data to display on Server View or Client View from Channelname.Content.

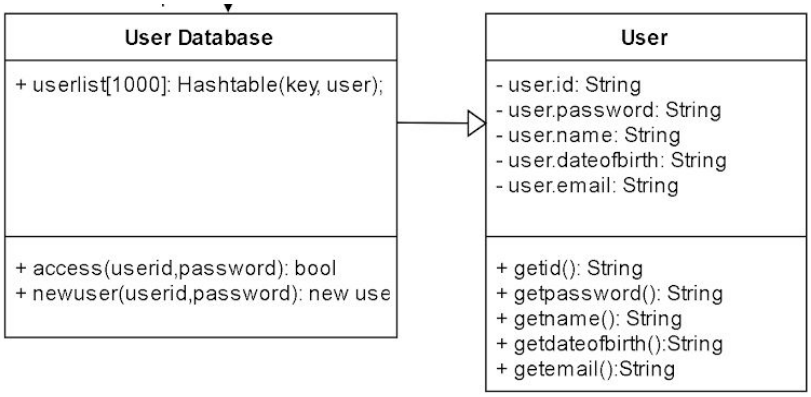
Data: User Input Data (Such as, but not limited to, User Messages, User Action and Timestamps) and Text Data.
Including data:
<ChannelName.ChannelData>
Ref to Package View 3.400

Name	Data Type	Range	Description

ChannelList	ArrayList<String>	Alpha-numeric	Store the list of channels
ChannelContent	String	Alpha-numeric	Store channel text content.

4.400.5 Messenger - User Database class

UML class diagram:



Identification: User Database

Type: A Java class

Purpose: Store users' data and information, provided by the user, such as theirs accounts, passwords, names, phone numbers, date-of-births, profile pictures (.jpg) and email addresses. As suggested in the SRS document, section 3.4[illegible]5

Function: Provide the authentication module information to authorize or deny access to the application or some part of the application.
Store and provide users' data for user-to-user communication purposes within the application.

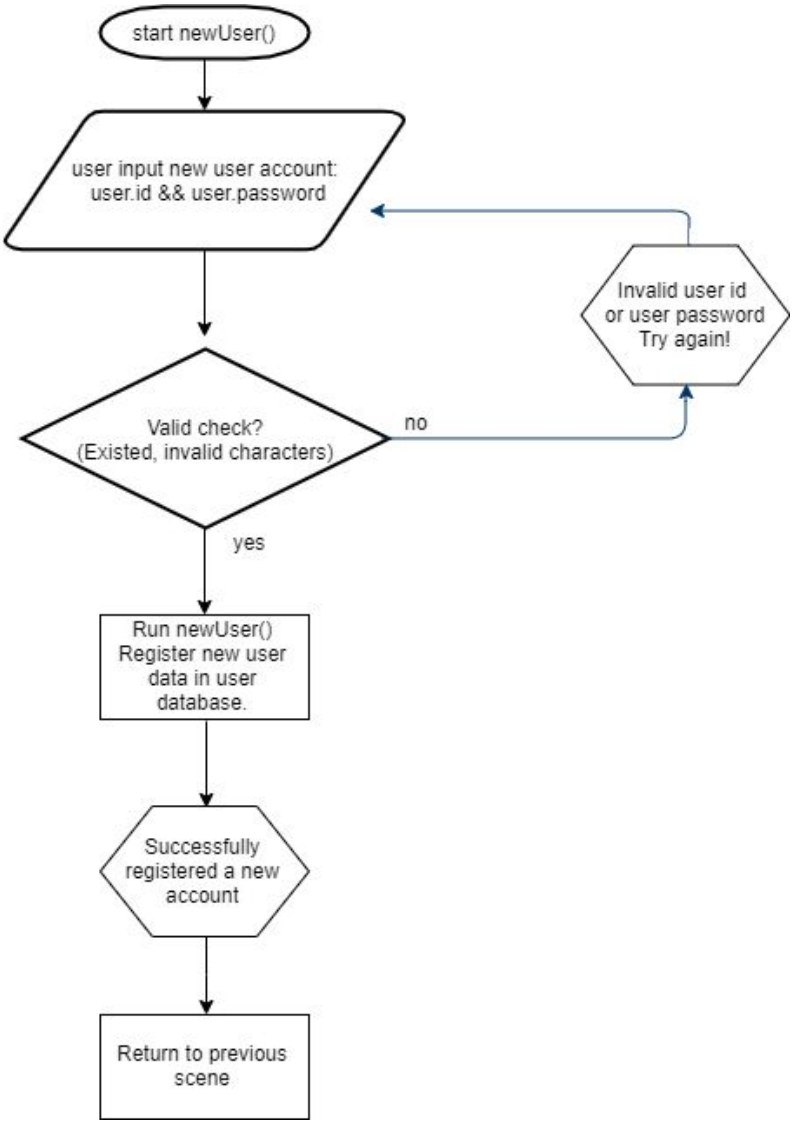
Subordinates: None.

Dependencies: Authentication Module and User View.

Interfaces: None.

Resources: The following dedicated server will be used to store this user database package:
- ABS Logic R730 / CPU: AMD FX8300@3.3GHz / RAM: 8GB / Hard Drive Disk: 1TB / Graphic Card: GeForce GT 730 / Internet: Download/ Upload Speed: 250/15 Mbps

Processing:



Data: <user.id>, <user.password>, <user.name>, <user.phonenumber>, <user.dateofbirth>, <user.emailaddress>.

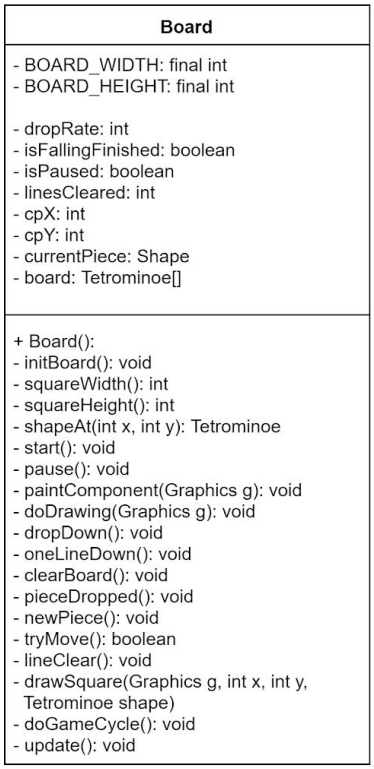
Name	Data Type	Range	Description

user.id	String	Alpha-numeric	Store user's ID
user.password	String	Alpha-numeric	Store user's password
user.name	String	Alpha-numeric	Store user's name
user.phonenumber	String	numeric	Store user's phonenumber
user.dateofbirth	Date	Alpha-numeric	Store user's date of birth
user.emailaddress	String	Alpha-numeric	Store user's email address

4.500 Tet-ri-poff Class View

4.500.1 Tet-ri-poff - Board Class

UML class diagram:



Identification: Board

Type: Java Class

Purpose: This class is used to create the gameboard and the functions to operate the game.

Function: Initializes gameBoard and allows the user to interact with the game using keystrokes that are assigned to certain functions that are created in this class.

Subordinates: JPanel

Dependencies: GameCycle, Shape, Tetripoff

Interfaces:

Resources: No external resources.

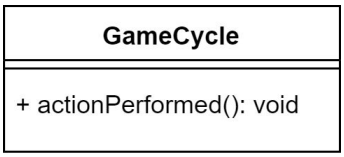
Processing: Start -> Creates GameBoard -> draws pieces-> calls GameCycle-> ActionListener wai[illegible] user input-> checks for lo[illegible] finishedFalling-> updates -> End

Data: <BOARD_WIDTH>, <BOARD_HEIGHT>, <dropRate>, <isFallingFinished>, <isPaused>, <linesCleared>, <cpX>, <cpY>, <currentPiece>, <Tetrominoe>.

Name	Data Type	Range	Description
BOARD_WIDTH	final int	numeric	Constant declaring the width of the board
BOARD_HEIGHT	final int	numeric	Constant declaring the height of the board
dropRate	int	numeric	Value assigned by difficult function to determine speed that pieces drop
isFallingFinished	boolean	true/false	Determines if piece has touched the bottom.
isPaused	boolean	true/false	Used to pause game and allow certain functions to work or not
linesCleared	int	numeric	Calculates when a line is cleared and is used to determine final score.
cpX	int	numeric	Positional value of the current piece's x coordinate
cpY	int	numeric	Positional value of the current piece's y coordinate
currentPiece	Shape	No range	A Shape Objec that describes the current piece being used
board	Tetrominoe Array	No range	An array that keeps track of the pieces and repaints the board at the rate of the game cycle

4.500.2 Tet-ri-poff - GameCycle Class

UML class diagram:



Identification: GameCycle

Type: Java Class

Purpose: This class is used to override the ActionListener and update the game when certain assigned keys are pressed.

Function: The ActionListener waits for an ActionEvent from a key being pressed, then updates and repaints the board following the action.

Subordinates: ActionListener

Dependencies: Board, Shape

Interfaces: None.

Resources: None

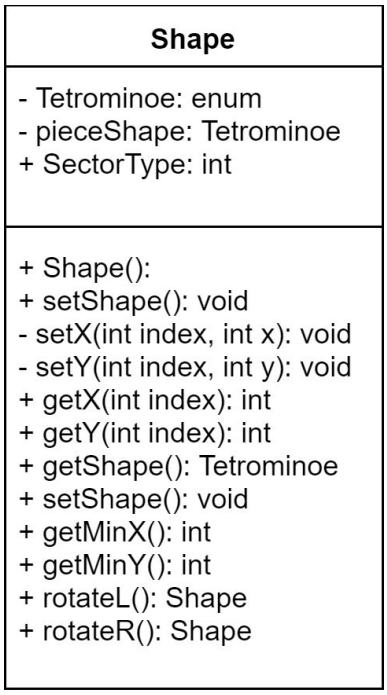
Processing:

Data: <e>

Name	Data Type	Range	Description
e	ActionEvent	Keyboard Keys	Received as an event which updates and repaints the board

4.500.3 Tet-ri-poff - Shape Class

UML class diagram:



Identification: Shape

Type: Java Class

Purpose: This class is used to create Tetrominoes, the pieces that are displayed on the game board.

Function: Initializes Board and allows the user to interact with the game using keystrokes that are assigned to certain functions that are created in this class.

Subordinates: Tetripoff

Dependencies: Tetripof

Interfaces: None

Resources: None

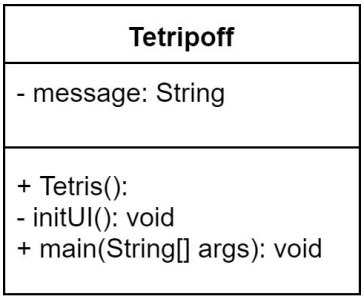
Processing:

Data: <Tetrominoe>, <pieceShape>, <user.name 

Name	Data Type	Range	Description
Tetrominoe	enum	numeric	An enumeration value that is used to randomly select a piece to place on the board.
pieceShape	Tetrominoe		A variable used to describe which Tetrominoe enumeration was selected and describe it's shape
sectorType	int	alphabetnumberic	Store user's name

4.500.4 Tet-ri-poff - Tetripoff Class

UML class diagram:



Identification: Tetripoff

Type: Java Class

Purpose: This class is used as a Driver to initiate the actual game.

Function:

Subordinates: None

Dependencies: Board, Shape, GameCycle

Interfaces: Swing UI

Resources: None

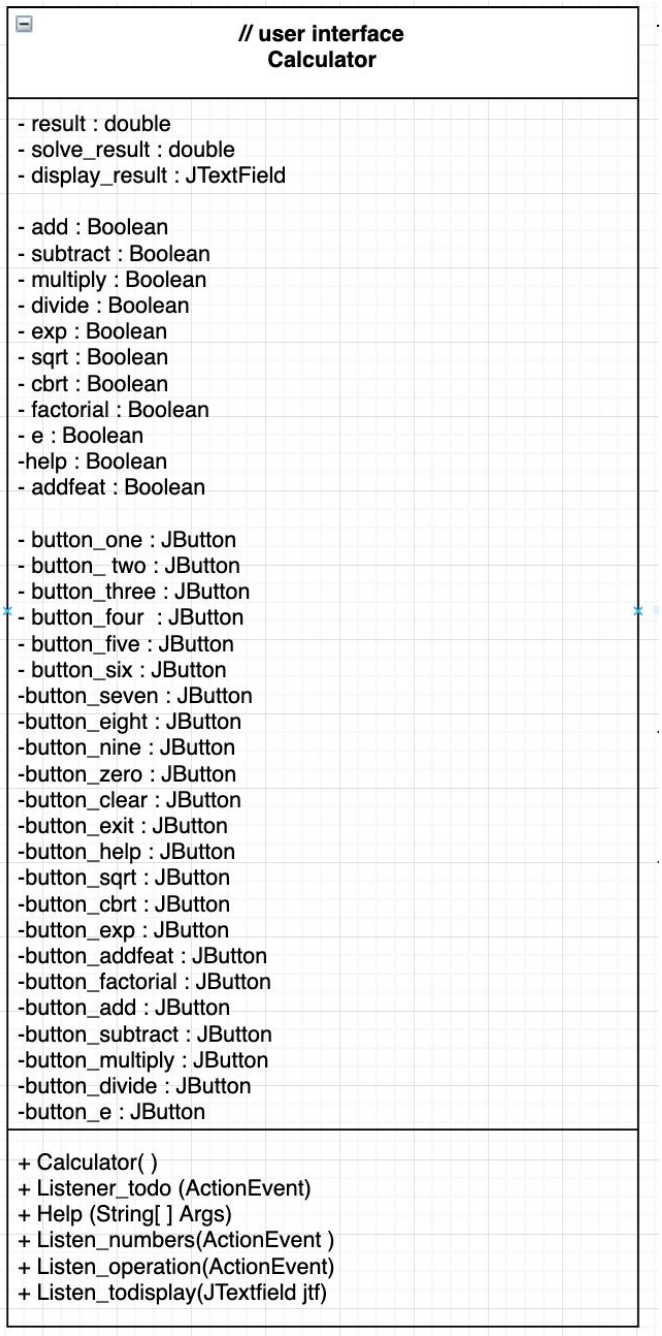
Processing:

Data:

4.600 BigCalc Class View

4.600.1 BigCalc - Calculator class

UML class diagram:



Identification: Calculator

Type: Java Class

Purpose: This class is used to create the physical calculator as the primary user interface.

Function: The class creates buttons to make the interface functional using the mouse. The class provides the user with a list of mathematical functions along with additional functionalities to clear, exit and help.

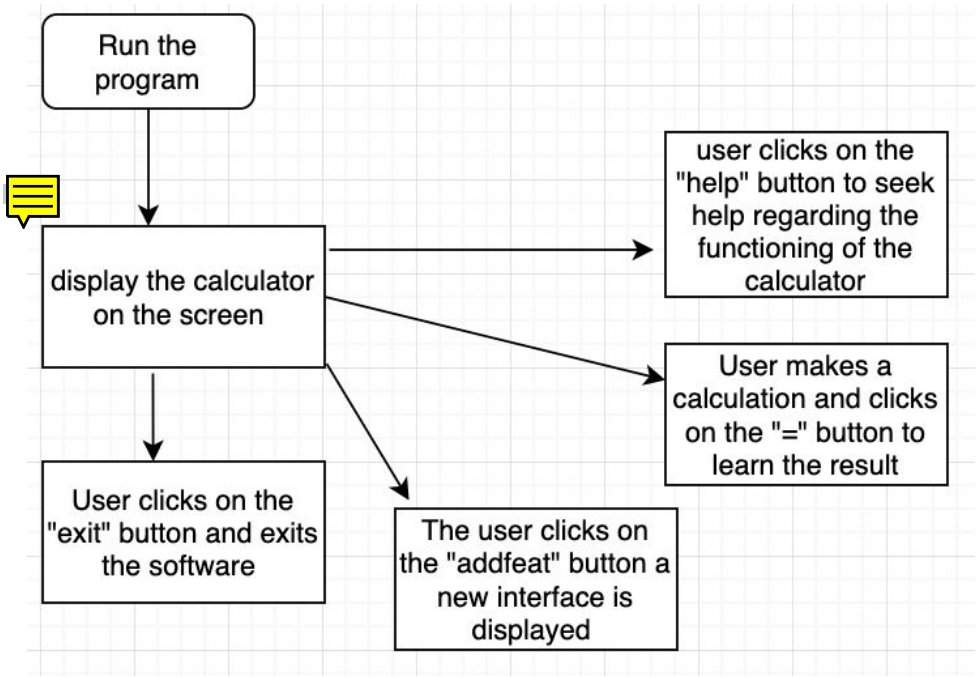
Subordinates: AddFeat

Dependencies: None

Interfaces: None.

Resources: The GUI in this module will generate a window that accounts for the minimum of 600 x 720 pixels on the screen, the user could reduce and increase the size accordingly by clicking on the buttons placed on the task bar.

Processing:



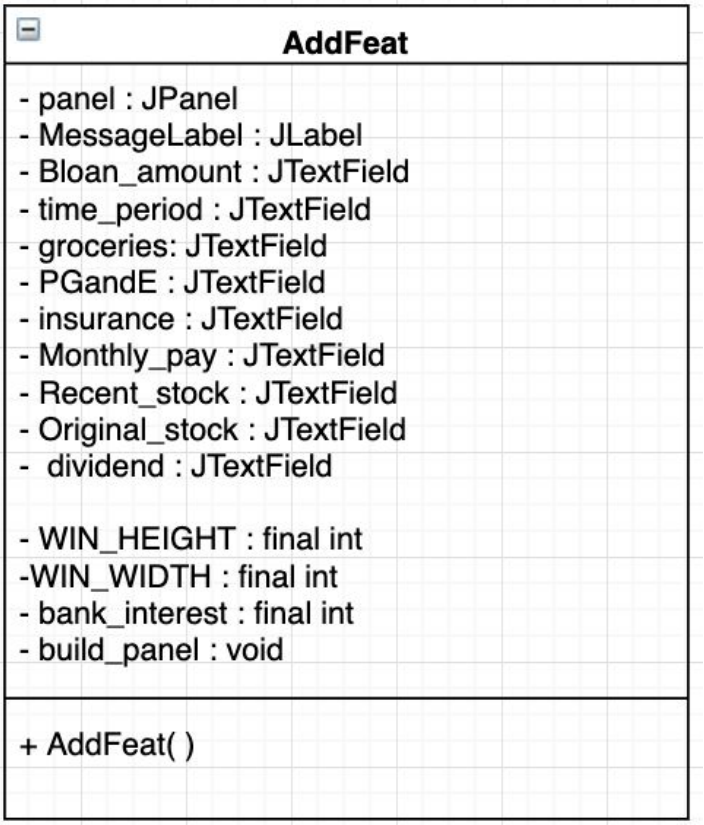
Data: The user will be able to make calculations based off his own data. The calculator in the most simpler since will use operators to perform operations on the operands presented by the user in the form of data.

Name	Data Type	Range	Description
1.Result	**double**	**numeric**	**Is used to store the final value of the whole operation being performed**
2.solve_result	**double**	**numeric**	**To store the value from the expression**
3.display_result	**JTextField**	**AlphaNumeric**	**For the list of values entered in the textfield**
4.add	**Boolean**	**True / False**	**Determines based on the T/F value if the - operation needs to be executed**
5.subtract	**Boolean**	**True / False**	**Determines based on the T/F value if the - operation needs to be executed**
6.multiply	**Boolean**	**True / False**	**Determines based on the T/F value if the - operation needs to be executed**
7.divide	**Boolean**	**True / False**	**Determines based**

			on the T/F value if the - operation needs to be executed
8.exp	**Boolean**	**True / False**	**Determines based on the T/F value if the exponential (e) operation needs to be executed**
9.facto	**Boolean**	**True / False**	**Determines based on the T/F value if the factorial (!) operation needs to be executed**
10.cbrt	**Boolean**	**True / False**	**Determines based on the T/F value if the cube root operation needs to be executed**
11.sqrt	**Boolean**	**True / False**	**Determines based on the T/F value if the square root operation needs to be executed**
12.help	**Boolean**	**True / False**	**Alerts the switch case if the help action listener needs to be executed**
13.addfeat	**Boolean**	**True ? False**	**Instantiates an instance of the addfeat class**
14.button_x // x refers to all the buttons used in this class from button_one to button_e	**JButton**	**(x,y)**	**Private member variable that invoke the corresponding ActionListeners**

4.600.2. BigCalc - The Addfeat class

UML class diagram:



Identification: AddFeat

Type: Java Class

Purpose: This class is used to create the additional features provided by BigCalc to make specific calculations easy to perform

Function: The class uses the JTextField interface to ask the user for all the data that is required, in the following chart fields. The class then uses prewritten algorithms to give the user an answer for their calculations on hitting the result button.

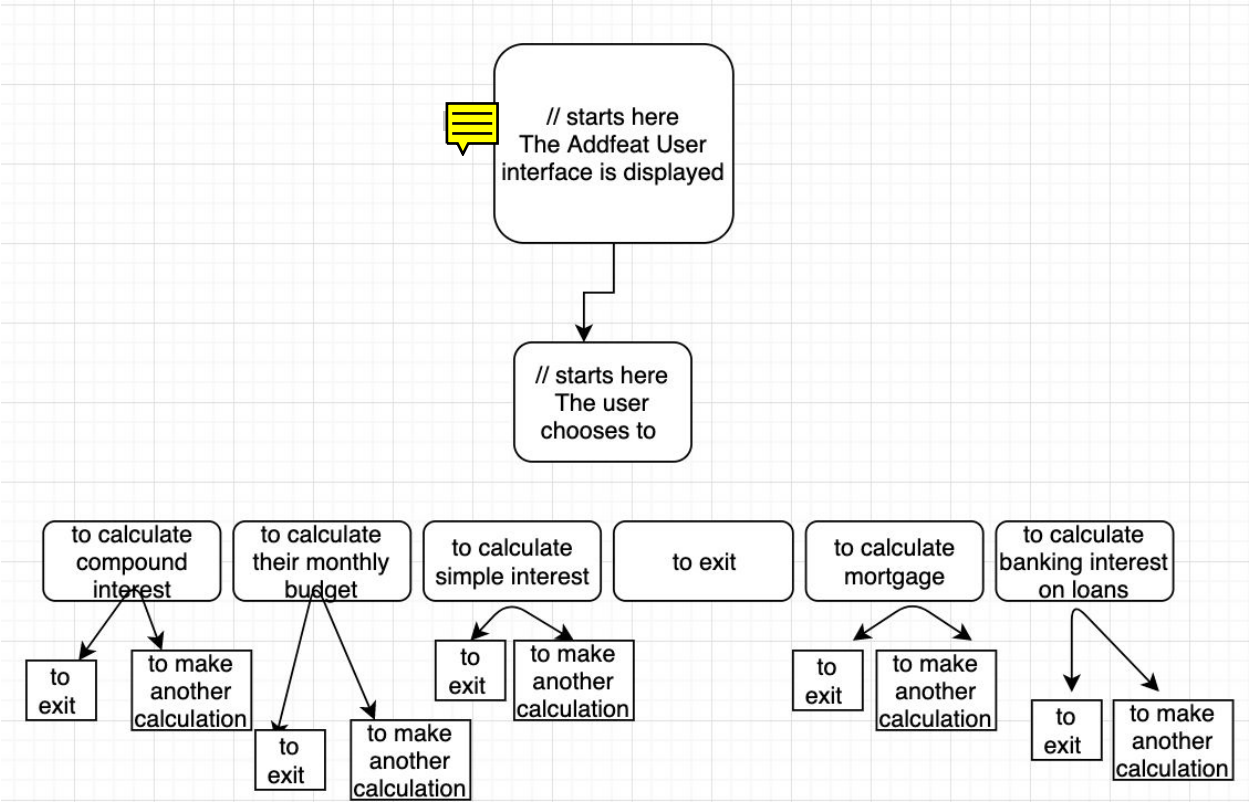
Subordinates: None

Dependencies: Calculator class

Interfaces: None.

Resources: None

Processing:



Data: The data for this class will be presented by the user. The “addfeat” calculator uses several chartfields where the user needs to input values in order to formulate his calculations. The addfeat class being a child class inherits a few features from the calculator class.

Name	Data Type	Range	Description
panel	Jpanel	(x,y)	Is used to create the actual panel that will hold the addfeat calculator
MessageLabel	JLabel	4 pixels	Are brief instruction labels that assist the users experience
Bloan_Amount	JTextField	Numeric	To ask the user for the sum amount of bank loan
time_period	JTextField	Numeric	Variable asking the user for a time period
groceries	JTextField	Numeric	Stores the amount of groceries for the budgeter option
PGandE	JTextField	Numeric	Stores the amount of PGandE for the budgeter option
insurance	JTextField	Numeric	Stores the amount of insurance for the budgeter option
monthly_pay	JTextField	Numeric	Stores the amount of monthly_pay for the budgeter option
dividend	JTextField	Numeric	Variable to store the rate of dividend for the stco_return option
recent_stock	JTextField	Numeric	
original_stock	JTextField	Numeric	
WIN_HEIGHT	JTextField	1000	To set the height of the panel
WIN_WIDTH	JTextField	1000	

build_panel	JTextField	-	To build the panel for the physical addfeat calculator interface
bank_interest	JTextField	double	Is a constant variable used to store the rate of interest charged on the bank loan option, stock return option, simple interest option and compound interest option.

6.0 Appendix: Glossary

Application: An application is any program, or group of programs, that is designed for the end user.

ASCII: The original, and now widely adopted, standard for encoding the letters and punctuation of the English alphabet, as well as a number of control sequences. It is a subset of the UTF-8 standard.

CLI: Command Line Interface

Database: A data structure storing data

Design Entity: An element (component) of a design that is separately named and referenced and is structurally and functionally distinct.

Design View: A subset of design entity attribute information that is specifically suited to the needs of a software project activity.

Entity attribute: A named characteristic or property of a design entity. It provides a statement of fact about the entity.

Software Design Specification (SDS): A representation of a software system created to facilitate analysis, planning, implementation, and decision making. A blueprint or model of the software system. The SDS is used as the primary medium for communicating software design information.

JFX: Java Virtual Effects

Demo: A demonstration of the product functions and features

Developer: A person or a group of people who develop software.

GB: Gigabyte, a unit for storage capacity.

GUI: Graphical User Interface

Internet: a global computer network providing a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols.

Java Virtual Machine: A virtual environment that Java Code runs on.

Java: A software programing language

Keyboard: An external input device.

Module: Part of the program which undertakes a specific task.

MongoDB: A database service provider

Monitor: An external output device.

Mouse: An external input device.

OS: Operating System

Password: A string that holds key to access a specific content or feature, usually go with a user account.

RAM: Random Access Memory.

Socket: A network socket is an internal endpoint for sending or receiving data within a node on a computer network.

SRS: Software Requirements Specifications

Stimulus: An action that triggers response.

Supervisor: Professor Hank Stalica.

Shell: The shell is a command interpreter in an operating system such as Unix or

	GNU/Linux, it is a program that executes other programs.
TeamDelta:	A group of individuals who wrote this SDS & will develop this project.
Text Buffer:	A finite but arbitrarily long string of text stored in the working memory of a program, that also provides an interface to edit its contents.
Thread:	The virtual version of a CPU core that handle a specific task.
Unicode:	The computing standard for encoding and representing text from a wide array of human languages.
User account:	An account to recognize a user's identity on the server.
User:	The person who uses the software.
UTF-8:	The leading implementation for encoding text that abides to the Unicode standard.
Windows:	An operating system developed by Microsoft.

6.1 Appendix: References

None

Comment Summary

Page 2

1. Good TOC. No complaints.

Page 6

2. Good purpose section. No complaints. Kind of hard to get this wrong, no matter what.
3. Sounds good. Good little succinct intro to what is going to be presented.

Page 7

4. No problem with dividing up work this way. Your team, your rules.

5. Sort of defeats the purpose of a group project, though, don't ya think? In the future, I'll probably disallow projects organized this way.
6. Ok, good, good, good.

Page 8

7. Really clear diagram. Good for orienting the reader.

Page 9

8. Who is this actor?
9. This is kind of weird. What do I code for this entity?

Page 11

10. This isn't a package diagram. This is a UML class diagram.

11. How does an organizational unit execute anything?

Page 13

12. Good diagram. Goes to identification, dependencies, interface, data, subordinates.

Page 14

13. Confusing. The above isn't a package diagram. What are you describing?

Page 15

14. Oh, it looks like you are describing what is *in* the package.
15. Not a package diagram. This is a class diagram.
    i. Assuming again you mean that this is the class inside your package. This isn't a package diagram though. This would be a diagram you'd include with the class description as an entity.

Page 16

16. Not a package diagram. Very strong UM though. Hoping to see each class diagram with their individual entity descriptions.

Page 18

17. Strong UML.

18. That's not a Purpose, that's a function.
19. How is that a function of a Package? I think you meant to say it organizes the classes that serve that function.

Page 19

20. Nonsensical. How can a package have an algorithm?
21. How does a package of a single picture. What's a picture of product data? This doesn't describe data at all.

Page 20

22. Not a package diagram, but it looks consistent that each team member is providing an overall UML class diagram of the classes in the package they belong to. Ok.
23. Packages don't execute anything.

    For example, how does

    import os;

    execute something?

Page 21

24. Strong UML.

Page 24

25. Strong description of the entity.
26. Good.

27. Good.
28. Good.

Page 26

29. More of the same goodness from the previous entity.

Page 28

30. Pretty good. Thorough.

Page 30

31. Consistently good.

Page 37

32. All of these methods are part of the interface. It's a little redundant to label each one as interface.

Page 38

33. GOOD. Very detailed diagram of the logic used by this entity. Well thought out.

Page 39

34. Good data dictionaries. Good treatment of the topic.

Page 40

35. Nice to see correct usage of the attributes.
36. Ok, I see what you did here. My thinking was you may have labeled this as "method", but I see why you chose Interface here.

Page 42

37. Might be a little on the excessive side here. Could place this in it's own section since the entire server would use this stuff, right?

Page 60

38. Good UML diagram here.

Page 61

39. Resources are external to the design. What's listed here looks like a listing of variables that belong to the class, but there is no description of each variable.

    This goes to the data attribute, not resources, but is incomplete.

40. Where is Intent Method in your interface? How is a method a parameter?
41. Reads like the Function attribute.

Page 62

42. What's this have to do with the data used by the entity?
43. How?
44. How?
45. I don't know what this figure is trying to say, or what it is.

Page 63

46. I don't know what this is.

Page 64

47. This doesn't describe the purpose of the entity at all.
    ii. This is function information.
48. Identification good, Type good, Function Good.
49. Good partial addressing of the data attribute. Need detailed descriptions of each of these in the data section
50. What is the type? What is the parameter name? How can a method return a class?
    iii. Need a full description of the method below.
51. This tells me nothing about how to write the ItemAdapter method.

Page 65

52. What is this?
53. Missing description of the data for the entity.
54. How? If this is a flow chart, it doesn't describe anything in detail. How do I store the picture data? What's the algorithm?

Page 66

55. What's a Tomato picture, Squash picture, etc? What kind of data type is that?
56. What's the parameter list? How do you return a class?
57. Not a purpose for the class.
58. Where in the UML is the MainActivity or ItemAdapter listed as a Dependency?
59. Where's a description of the method?
60. Where's' a description of the data?

Page 67

61. Doesn't tell me anything about how to write this.

Page 68

62. Similar problems as above.
63. Not a purpose.

Page 69

64. Tells me nothing about the methods.
65. How?
66. How?
67. Has nothing to do with the data attribute.
68. Flowchart doesn't convey any information about the algorithms used by the entity.

Page 70

69. Doesn't look like a function...looks like a class according the UML diagram.
70. No interfaces? I see three methods in your UML diagram...
71. This doesn't describe external resources used by the design. This describes the function.
72. Doesn't describe the purpose of the entity.

Page 71

73. How is a serverIP "Numeric"

Page 72

74. Looks like a class based on the diagram.
75. Not a purpose.
76. Three methods listed in the UML, where is a description of the interface?
77. A database is a resource.

Page 73

78. What function?
79. Good, but should there be any initialization values?
80. How does a variable fetch stuff?

Page 74

81. Not a valid Java identifier, btw.
82. Not a purpose, this is a function.
83. 9 methods in the UML diagram. Where are their descriptions?
84. YUP.
85. This is more hardware requirement information. Kind of pointless to include this. You could include this for every entity based on your rational of including it here.

Page 75

86. Incomplete. 5 variables in the UML diagram, but only 4 listed here.

Page 76

87. Why is this a class, but everything else is a function?
88. Not a purpose.
89. Pointless, see previous comment.
90. How? This tells me nothing about the algorithms used by the entity?
91. what data type is channelname_content.txt ?
92. What's the data type?
93. I see two methods in the interface. Where is their descriptions?

Page 77

94. There's a correct use of Purpose.
95. Where in the UML, I don't see any?
96. I see two methods in your interface. Where's the descriptions.

Page 78

97. Pointless: see above.

Page 79

98. Java doesn't allow periods in identifiers.

Page 80

99. Not a purpose, this is a function.
100. JPanel doesn't inherit from your Board class.
101. Where are the descriptions of all the methods in the interface?

Page 81

102. OK.
103. How do I do that checking?

104. Pretty good.

Page 82

105. You have a method in your UML.
106. Not a Purpose. This is it's function.
107. I don't see e listed in your UML class diagram.

Page 83

108. Not a Purpose.
109. Which is it? A dependency or a subordinate?
110. I count 12 methods in the interface according to your UML. Where are the descriptions?
111. Incomplete. Should be marked None if there is nothing to say about this attribute.

Page 84

112. What's the range?
113. Java identifiers don't have periods in them. Why list this here, the names are below and in your UML.

Page 85

114. Where's the Function?
115. Not a Purpose.
116. Don't see any listing for Swing UI in your class diagram. Also, no description of the three methods athat are actually in your diagram.
117. Incomplete.
118. Incomplete.

Page 87

119. I count 6 methods in your interface. Where are their descriptions?

120. Not a purpose. This is it's function.
121. Not an external resource. This is more a description of Function.
122. Not sure what this is describing? Looks like some sort of flow chart, but doesn't have the syntax of a flow chart.

    Might be a bit like a Use Case, but doesn't describe the logic of any algorithm used by the entity.
    iv. Not really helpful.

Page 91

123. Not a purpose. This is it's functions.
124. One method in the interface in your UML diagram.

Page 92

125. Ok, I can follow along a bit as to what is happening here.

    This is a little more helpful, still this feels more like a use case than a description of any complex algorithms.

    Ok.