Faculty of Engineering & Environment

Module: CM0718 – Program Design and Implementation

Module tutor: Michael J Brockway

Title- Report

22/05/2015

Pritesh Bhole– w14043450

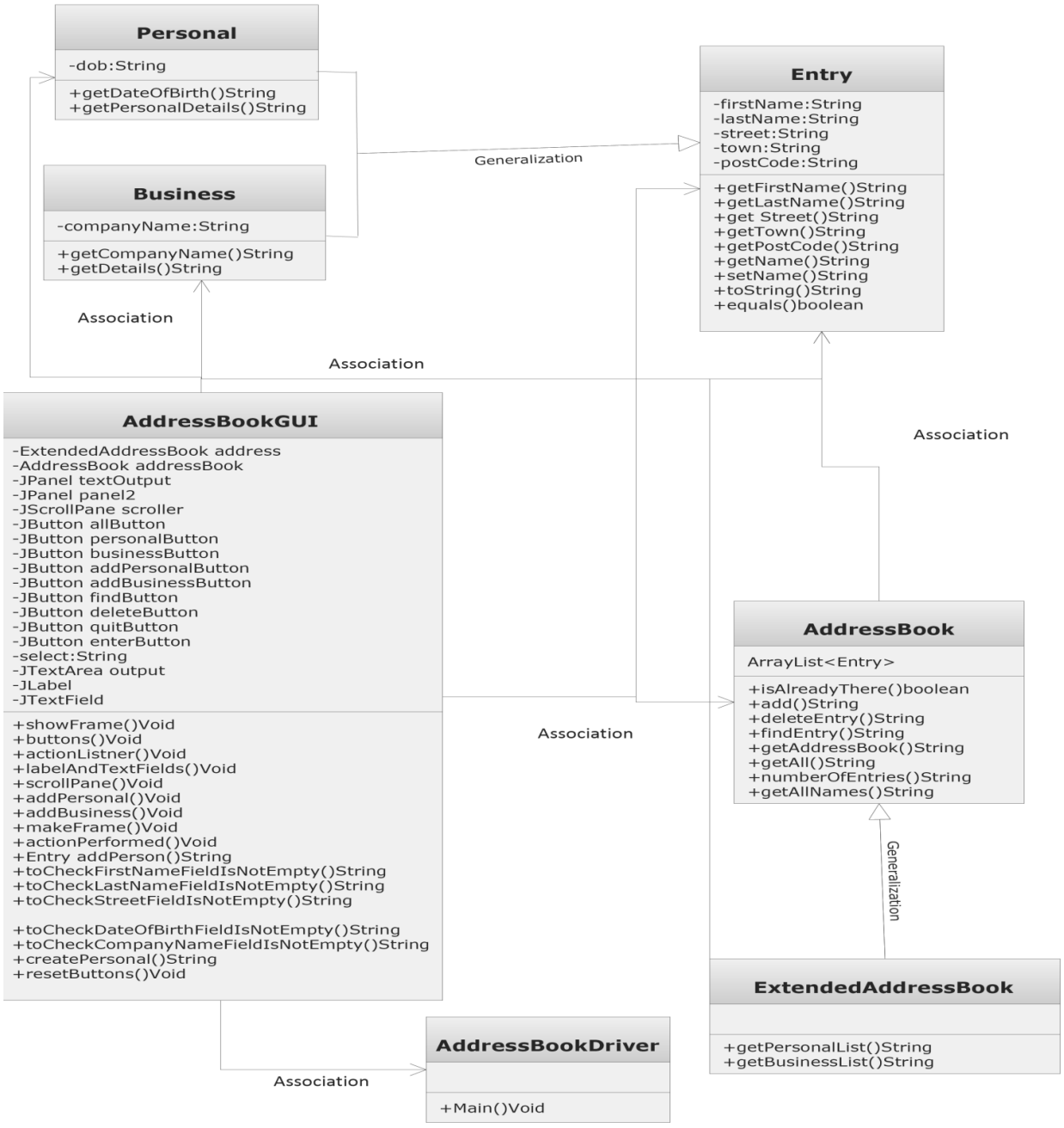
Computing & Information technology

Contents

Introduction

This report outlines design and testing documentation along with critical reflection of Address Book system.

Task 4.c



Submitted by-Pritesh Bhole (w1404350)

Address Book system has seven classes: Entry, Personal, Business, AddressBook, ExtendedAddressBook, AddressBookGUI and AddressBookDriver.

- The Entry class contains person's contact information: first name, last name and address details. It has the methods to full details (toString). This class also has condition to check postcode if it is in UK post code format and adds only a valid post code.
- Personal class is a subclass of Entry class. It inherits all attribute from Entry and also adds date of birth attribute which is needed for adding personal entry. It has a return method (toString) that return person's details along with date of birth.
- Business class is a subclass of Entry class. It inherits all attribute from Entry class and also adds company name attribute which is needed to add business entry. It has a return method (toString) that return person's details along with company name.
- The AddressBook class has an Arraylist that store entry made by user. It records person's first and last name, street number and name, town and postcode and either date of birth or company name. All names in the AddressBook are unique to achieve this Boolean method (isAlreadyThere) is used. Using its methods we can add person (add), remove or find a person by matching the first name and the last name (deleteEntry and findEntry respectively). getAll method returns full list of entries. We can also get the total number of entries in the address book (numberOfEntries).
- ExtendedAddressClass is a subclass of AddressBook class. It inherits array list to return personal list or business list depending on user input. We need this class so that we can return data separately in each case of user input i.e. personal list and business list.
- AddressBookGUI is the class which provides user interface. It has methods that make's frame and all other component inside frame and connects these components through action listeners (actionPerformed) to different methods. It has a method to add person (addPerson) depending on type of entry is selected i.e. personal and business depending on this either personal (createPersonal) or business (createBusiness) object is created and added. Before creating this object it has various methods to check that empty string is not added and weather postcode and date of birth is in valid format. It also has a method that clear text field once the entry is added (toClearTextField).
- AddressBookDriver class contains the main method that runs the entire program to initiate graphical display of system.

Task 4.d

Strategy for testing

To begin with, I have checked constructors in all classes to confirm that object are properly created.

Return method of all classes are tested to make sure that they are returning the data they are expected to return in case of both valid input and invalid input. I have checked date of birth should be valid data before adding by using assertTrue when valid date is entered and assertFalse when in valid is entered. In similar manner I have tested that a valid postcode should get added and for invalid it should return a bad post code error (???). I have also tested toString to make sure it is returning basic details of person added correctly.

Functionality test

To test List all I have added an entry, than by using assertEquals I have tested if it is returning the entries added. Similarly to test list personal and business I have added entries respectively and checked if they are returning value expected i.e. along with basic details for personal date of birth and company name in case of business.

To test Add personal and business functional I have added an entry respectively and then with help assertEquals i have checked them if they are added or not. Moreover, I also tested that system doesn't add duplicate entry and all names are unique by using assertFalse is used to show entry doesn't exist and assertTrue shows entry with user inputted name is already present in address book.

To test remove method I have added an entry and than tried to remove existing entry and checked with assertFalse also, I tried to remove a entry which doesn't exist and checked it with assertTrue. Likewise I did same thing to test find functionality finding a valid user and for entry which is not added and checked result by assertFalse and AssertTrue respectively.

GUI testing

For this I used black box testing approach. I have checked each functionality requirement of system. I have checked whether text field is accepting user input and when enter button is clicked it is storing it. Furthermore, if it is showing the enties when list all button is used or personal/ business depending on entry type. I have also checked that empty text field should not get added and also valid date of birth, first name, last name and post code should be added. I have checked find and remove function by using valid as well as invalid input.

Moreover, I have also done regression testing making sure change in one functionality code is not affecting other. However I haven't done non-functionality testing.

Test Data

addEntry- "pritesh", "bhole", "gowland", "newcastle", "NE4 4NE"

addPersonal- "pritesh", "Bhole", "gowland", "newcastle", "NE4 4NE","09/09/1992"

addBuisness- "pritesh", "Bhole", "gowland", "newcastle", "NE4 4NE","inpanch"

ID	Description	Pre-conditions	Test Data	Expected Result
testConstructor()	Test Constructor in Entry class	Create an object entry1 with addEntry details	Add entry	Not null
testGetFirstName()	Testing if it gets the correct first name	Create an object entry1 with addEntry details	First name	Returns true if first name equals pritesh
testGetLastName()	Testing if it gets the correct last name	Create an object entry1 with addEntry details	Last name	Returns true if last name equals Bhole
testGetTown()	Testing if it gets the correct town	Create an object entry1 with addEntry details	Town	Returns true if town equals Newcastle
testGetStreet()	Testing if it gets the correct street	Create an object entry1 with addEntry details	street	Returns true if street equals gowland
testGetPostCode()	Testing if it gets the valid postcode	Create an object entry1 with addEntry details	"NE4 4NE"	Returns true if postcode equals NE4 4NE

testInvalidPostCode()	Testing for invalid postcode is entered	Create an object entry1 with addEntry details	"" and NE4NE	Return true if postcode is not added
testToString()	Testing for to get correct details	Create an object entry1 with addEntry details	addEntry	Return true if all details are equal to addEntry
testPersonalConstructor()	Test Constructor in Personal class	Create an object personal1 with addPersonal details	addPersonal	Not null
testGetDateOfBirth()	Test if valid date of birth is entered	Create an object personal1 with addPersonal details	"09/09/1992"	Returns true if date of birth equals 09/09/1992
testGetFalseDateOfBirth()	Test if invalid date of birth is entered	Create an object personal1 with addPersonal details	"0909"	Returns true if date of birth not added
testPersonal ClassToString()	Test if it returns valid personal details i.e. basic details and date of birth	Create an object personal1 with addPersonal details	addPersonal	Return true if all details are equal to addPersonal
testBusinessConstructor()	Test Constructor in Business class	Create an object business1 with addBusiness details	addBuisness	Not null
testGetCompanyName()	Test if it returns company name	Create an object business1 with	"inpanch"	Return true if company name is equal to

		addBusiness details		inpanch
testBusiness ClassToString()	Test if it returns valid business details i.e. basic details and company name	Create an object business1 with addBusiness details	addBuisness	Return true if all details are equal to addBusiness
testAddressBookCreated()	Test Constructor in address book class	addEntry	New object addressB1	Not null
testAddMethod()	Tests if a new entry has been added. Also tests if numberOfEntries() returns the correct number of entries in address book	Create new addressB1 object	addEntry	Returns true if person with addEntry details is added successfully
testPersonalAddMethod()	Tests if a new Personal entry has been added. Also tests if numberOfEntries() returns the correct number of entries in address book	Create new addressB1 object	addPersonal	Returns true if person with addPersonal details is added successfully
testBusinessAddMethod()	Tests if a new business entry has been added. Also tests if numberOfEntries() returns the correct number of entries in address book	Create new addressB1 object	addBusiness	Returns true if person with addBusiness details is added successfully
testIsAlreadyThere()	Tests whether the name is unique or not	Create new addressB1 object	addEntry	Return true if name is unique before adding or else doesn't add the person
testRemoveMethod()	To test if person is	Create new	addEntry	Return true if

	removed using first and last name	addressB1 object		person is successfully removed
testIfPersonNotRemoved()	To test if person is not removed using first and last name	Create new addressB1 object	addEntry	Return true if person to be removed is not present
testFindMethod()	Tests if the correct name has been found given first and last names	Create new addressB1 object	addEntry	Return true if person is found
testIfPersonNotFound()	Tests if person is not there in address book	Create new addressB1 object	addEntry	Return true if person to be found is not present in address book
testGetAllMethod()	To test if all entries are returned	Create new extended1 object	addEntry	Returns true if details are equals to addEntry
testGetPersonalList()	Test if all personal entries are returned	Create new extended1 object	addPersonal	Returns true if details are equals to add Personal
testGetBusinessList()	Test if all business entries are returned	Create new extended1 object	addBusiness	Returns true if details are equals to add Business

Task 4.e

Critical evaluation

I started program development by making frame than providing its component with proper action listener and then handling errors. While making frame first I made use of just flow layout and border layout but didn't have it right, so did some more research and found some example and then made use of grid layout along with border layout which gave me the desired frame. After that I created Business and Personal class. Later I created methods of action listener I implemented add method, but found some difficulty in making it work as there were two different contact and have one different field each, so did some more research and found a way out of it. Then while creating list personal and list business function found the necessity of one more class in order to get separate list and hence created extended address book class. After everything was working I then proceeded for error handling. At the beginning for empty field I created a single method to check and give error for empty field, but later while making modification to add error checking for post code and date of birth it became a bit complicated, hence I decided to break it down into more smaller method so that each empty text field can be checked individually and can throw specific error as this will also increase the modifiability factor of system. In this along with empty string I have also checked that only a valid first name, last name, postcode and date of birth should be added.

System Improvements

This system at present doesn't have any functionality to change name or address if a user want to change name or address he/she has to delete old entry and create a new one .Moreover, program doesn't take into account if a person enters space in between letter like (first name = pr i t) it saves as it is without taking out spaces. Size of field doesn't remain constant if the frame size is minimized or maximised. There is no minimum limit to minimize frame.

Conclusion

This assignment has helped me develop deeper understanding of java along with testing knowledge. It has also helped understand proper technique and procedure to be followed while creating an application and also proper technique to write a program taking into consideration modifiability and maintenance factor for future development.