Programming Fundamentals 2

Assignment 2 Report

Ben Capper

20088114

SSD W1

# Development and Implementation

The biggest issue I had with this project was working with the ArrayLists used to construct a sub class such as the special interest ArrayList in Scout. For example, when creating a cub, I have a while loop that asks to enter another interest until no is chosen. To do this I had to target the specific cub which the user had just entered details for. Index number won’t work in this case because if any scout in the list is not a cub there is a type error. This meant another way of targeting a specific group and specific scout from that group was needed. For this I made a method in the ScoutList class which targeted the scout by their cub/beaver/scouter phone number and what group they are a part of using instanceof and casting.

I also had an issue with the update methods. I was trying to update directly from the ScoutSystem class rather than collect the information then pass it into a ScoutList method. This was easily overcome when pointed in the right direction.

After finishing the main section of the project, I decided to add an events section as it follows the same inheritance principles and makes sense in this context. I could then add more constructor ArrayLists to get a better grip on the concept and allowed me to make more advanced calculations as I now had location costs, equipment costs and donations pledged.

# Scout (Abstract – Implements iScoutMember)

This class has been fully implemented. It is an abstract class which implements iScoutMember, making the calcRegistrationFee method necessary in all sub classes. Every scout is constructed with an ArrayList of special interests and this ArrayList can be displayed. Javadoc comments are used throughout the class.

* **Scout()** – Scout constructor.
* **getName()** – Returns the Scouts name.
* **setName()** – Sets the Scouts name to a specified value.
* **getCounty()** – Returns the Scouts county.
* **setCounty()** – Sets the Scouts county to a specified value.
* **getDateOfBirth()** – Returns the Scouts date of birth.
* **setDateOfBirth()** – Sets the Scouts date of birth to a specified value.
* **getAddress()** – Returns the Scouts address.
* **setAddress()** – Sets the Scouts address to a specified value.
* **getPhoneNumber()** – Returns the Scouts phone number.
* **setPhoneNumber()** – Sets the Scouts phone number to a specified value.
* **display()** – Cycles through ArrayList of Interests, creates a String of all interest’s toString, then returns the String
* **getSpecialInterests()** – Returns the ArrayList of Special Interests.
* **setSpecialInterests()** – Assigns an ArrayList of Interests to the Scout.
* **addSpecialInterest()** – Adds a Special Interest to the ArrayList of Interests.
* **toString()** - Returns all Scout information. (variable values)

**5/5**

# BeaverScout (Extends Scout)

This class has been fully implemented. It extends Scout, inheriting all scout attributes. Super is used when constructing a beaver and in the toString method. The attributes are protected rather than private and the calcRegistrationFee method is implemented and returns the correct amount. Javadoc comments are used throughout the class.

* **BeaverScout()** – Beaver constructor.
* **getBeaverPhoneNumber()** – Returns the Beavers phone number.
* **setBeaverPhoneNumber()** – Sets the Beavers phone number to specified value.
* **calcRegistrationFee()** – Returns 100, the Beaver registration cost.
* **toString()** – Returns all Beaver information.

**5/5**

# CubScout (Extends Scout)

This class has been fully implemented. It extends Scout, inheriting all scout attributes. Super is used when constructing a cub and in the toString method. The attributes are protected rather than private and the calcRegistrationFee method is implemented and returns the correct amount. Javadoc comments are used throughout the class.

* **CubScout()** – Cub constructor.
* **getCubPhoneNumber()** – Returns the Cubs phone number.
* **setCubPhoneNumber()** – Sets the Cubs phone number to specified value.
* **calcRegistrationFee()** – Returns 150, the Cub registration cost.
* **toString()** – Returns all Cub information.

**5/5**

# Scouter (Extends Scout)

This class has been fully implemented. It extends Scout, inheriting all scout attributes. Super is used when constructing a scouter and in the toString method. The attributes are protected rather than private and the calcRegistrationFee method is implemented and returns the correct amount. Javadoc comments are used throughout the class.

* **Scouter()** – Scouter constructor
* **getScouterPhoneNumber()** – Returns the Scouters phone number.
* **setScouterPhoneNumber()** – Sets the Scouters phone number to specified value.
* **calcRegistrationFee()** – Returns 10, the Scouter registration cost.
* **toString()** – Returns all Scouter information.

**5/5**

# SpecialInterest

This class allows for the creation of Special Interest objects and stores methods to manipulate those objects. Special Interests are added to a scouts special interest ArrayList. Javadoc comments are used throughout the class.

* **SpecialInterest()** – Special Interest constructor.
* **getInterestCategory()** – Returns the interest category.
* **setInterestCategory()** – Sets the interest category to specified value.
* **getDetails()** – Returns the special interest details.
* **setDetails()** – Sets the special interest details to specified value.
* **getDateBadgeReceived()** – Returns the date the interest badge was received.
* **setDateBadgeReceived()** – Sets the date the badge was received on to the specified value.
* **toString()** – Returns all Special Interest information.

# ScoutList

This class is fully implemented and stores methods which can be used to manipulate the ArrayList of scouts. This includes the ability to create, read, update and delete objects along with other methods used to identify and list certain scouts based on user choice. Instanceof and casting has also been used frequently. Javadoc comments are used throughout the class.

* **addScout()** – Adds a scout to the ArrayList.
* **addInterest()** – Cycles through the ArrayList until it finds a scout with a (group) phone number matching the one entered, then adds the interest to that scouts ArrayList of interests.
* **removeScout()** – Removes a scout from the ArrayList.
* **numberOfScouts()** – Returns the amount of objects in the arrayList.
* **getScout()** – Returns the scout specified by the user by index.
* **getCubByPhone()** – Returns a cub whose cub phone number matches the input.
* **getBeaverByPhone()** – Returns a beaver whose beaver phone number matches the input.
* **getScouterByName()** – Returns a scouter whose scouter phone number matches the input.
* **getScoutByName()** – Returns a scout whose name matches the input.
* **listAllScouts()** – Creates a String of each Scout, then lists all scouts in the ArrayList.
* **listCubScout()** – Returns a String of all cubs in the ArrayList.
* **listBeaverScout()** – Returns a String of all beavers in the ArrayList.
* **listScouter()** – Returns a String of all Scouters in the ArrayList.
* **updateCubScout()** – Updates a cubs information based on user input.
* **updateBeaverScout()** – Updates a beavers information based on user input.
* **updateScouter()** – Updates a scouters information based on user input.
* **save()** – Saves all scouts to a .xml file.
* **load()** – Loads all scouts from a .xml file.

**25-30/30**

# InterestList (Extra Functionality)

This class stores methods which can be used to manipulate an ArrayList of special interests. This is separate to the ArrayList of special interests used to construct a scout and exists to allow for the listing of all interests at once. Javadoc comments are used throughout the class.

* **addSpecialInterest()** – Adds an interest to the ArrayList.
* **numberOfInterests()** – Returns the amount of objects in the ArrayList.
* **getSpecialInterest()** – Returns an interest object based on user input.
* **removeInterest()** – Removes an interest from the ArrayList.
* **listAllInterests()** – Returns a String of all interests in the ArrayList.
* **save()** – Saves all interests in the ArrayList to a .xml file.
* **load()** – Loads all interests in the ArrayList from a .xml file.

# iScoutMember

This class, I think has been fully implemented. It is implemented by the abstract scout class to force its sub classes to implement any methods contained within. Javadoc comments are used throughout the class.

* **calcRegistrationFee()** – Forces each class implementing iScoutMember to include this method. Returns a different amount based on Scout subclass.

**3-5/5**

# Event (Abstract – Implements iEventMember)(Extra Functionality)

This is an abstract class which implements the iEventMember interface, forcing the calcEquipCost and calcLocationCost methods in each of its sub classes. Every type of event object is constructed with an ArrayList of equipment and has a location cost associated with it. Events attributes use protected rather than private. Javadoc comments are used throughout the class.

* **Event()** – Event constructor.
* **getDate()** – Returns the event date.
* **setDate()** – Sets the event date based on input.
* **getLocation()** – Returns the event location.
* **setLocation()** – Sets the event location based on input.
* **getDuration()** – Returns the event duration.
* **setDuration()** – Sets the event duration based on input.
* **getCost()** – Returns the event cost.
* **setCost()** – Sets the event cost based on input.
* **getEquipment()** – Returns the ArrayList of equipment.
* **setEquipment()** – Assign an ArrayList of equipment to the event.
* **addEquipmentToList()** – Add an equipment object to the ArrayList of equipment.
* **displayEquipment()** – Returns a String of every equipment object in the equipment ArrayList.
* **toString()** – Returns all event information.

# ScoutEvent (Extends Event)(Extra Functionality)

This class extends event and is constructed using an ArrayList of scouts who are invited to the event based on user choice. Javadoc comments are used throughout the class.

* **ScoutEvent()** – Scout Event constructor.
* **getScoutsInvited()** – Returns an ArrayList of scouts, invited to this event.
* **setScoutsInvited()** – Assigns an ArrayList of scouts to this ScoutEvent.
* **display()** – Returns a String containing every scout in the scoutsInvited ArrayList.
* **addScoutGroup()** – Adds a scout to the scoutsInvited ArrayList.
* **getActivity()** – Returns the event activity.
* **setActivity()** – Sets the event activity based on input.
* **calcLocationCosts()** – Returns the cost of this events location.
* **calcEquipCost()** – Returns the cost of this events required equipment.
* **toString()** – Returns all ScoutEvent information.

# FundraiseEvent (Extends Event)(Extra Functionality)

This class extends event and is constructed using an ArrayList of donators who have pledged a donation. Javadoc comments are used throughout the class.

* **FundraiseEvent()** – Fundraise Event constructor.
* **getDonators()** – Return the ArrayList of donators.
* **setDonators()** – Assigns an ArrayList of donators to this fundraise event.
* **displayDonator()** – Returns a String containing all donators in the donators ArrayList.
* **addDonatorToList()** – Adds a donator object to the donator ArrayList.
* **getName()** – Returns fundraise event name.
* **setName()** – Sets the fundraise event name.
* **calcLocationCost()** – Returns the cost of this events location.
* **calcEquipCost()** – Returns the cost of this events required equipment.
* **toString()** – Returns all FundraiseEvent information.

# Equipment (Extra Functionality)

This class allows for the creation of equipment objects which are required to hold a scout/fundraise event. Javadoc comments are used throughout the class.

* **Equipment()** – Equipment constructor.
* **getName()** – Returns the equipment name.
* **setName()** – Sets the equipment name based on input.
* **getSupplier()** – Returns the equipment supplier.
* **setSupplier()** – Sets the equipment supplier based on input.
* **getCost()** – Returns the equipment cost.
* **setCost()** – Sets the equipment cost based on input.
* **getAmount()** – Returns the quantity of this piece of equipment.
* **setAmount()** – Sets the quantity of this piece of equipment based on input.
* **toString()** – Returns all equipment information.

# Donator (Extra Functionality)

This class allows for the creation of donator objects which are required to hold a fundraise event. Javadoc comments are used throughout the class.

* **Donator()** – Donator constructor.
* **getName()** – Returns the donators name.
* **setName()** – Sets the donators name based on input.
* **getPhone()** – Returns the donators phone number.
* **setPhone()** – Sets the donators phone number based on input.
* **getAddress()** – Returns the donators address.
* **setAddress()** – Sets the donators address based on input.
* **getEmail()** – Returns the donators email address.
* **setEmail()** – Sets the donators email address based on input.
* **getDonation()** – Returns the donation amount.
* **setDonation()** – Sets the donation amount based on input.
* **toString()** – Returns all donator information.

# EventList (Extra Functionality)

This class stores methods which can be used to manipulate event objects in an ArrayList. Javadoc comments are used throughout the class.

* **addEvent()** – Add an Event to the ArrayList.
* **removeEvent()** – Removes an event from the events ArrayList.
* **numberOfEvents()** – Returns the amount of Events in the ArrayList.
* **getEvent()** – Returns the event at the specified index number.
* **listAllEvents()** – Returns a String containing all events in the ArrayList.
* **listFundEvent()** – Returns a String of all fundraise events in the ArrayList.
* **listScoutEvent()** – Returns a String of all scout events in the ArrayList.
* **listDonators()** – Returns a String of all donators across all fundraise events.
* **updateFundEvent()** – Updates a fundraise events information based on user input.
* **updateScoutEvent()** – Updates a scout events information based on user input.
* **save()** – Saves all events in the ArrayList to a .xml file.
* **load()** – Loads all events in the ArrayList from a .xml file.

# iEventMember (Extra Functionality)

This interface class is implemented by Event and ensures each sub class contains these methods to calculate costs. Javadoc comments are used throughout the class.

* **calcLocationCost()** – Forces any class implementing iEventMember to include this method. Returns the location costs of this event.
* **calcEquipCost()** – Forces any class implementing iEventMember to include this method. Returns the equipment costs of this event

# ScoutSystem / UI Elements

This class is the user interface and is fully implemented. It provides all necessary options in a clean, easy to read format with a lot of extra functionality. Javadoc comments are used throughout the class.

* **mainMenu()** – User interface main menu.
* **runMainMenu()** – Runs methods based on main menu choice.
* **scoutMenu()** – Scout options menu.
* **runScoutMenu()** – Runs methods based on scout options menu.
* **scoutRegisterMenu()** – Scout type menu.
* **runScoutRegisterMenu()** – Runs methods based on scout register menu.
* **scoutListByGroupMenu()** – Scout groups menu.
* **runScoutListByGroupMenu()** – Runs methods based on scout list by group menu.
* **runScoutUpdateGroupMenu()** – Runs methods based on scout list by group menu.
* **eventMenu()** – Event options menu.
* **runEventMenu()** – Runs methods based on event options menu.
* **eventRegisterMenu()** – Event type menu.
* **runEventRegisterMenu()** – Runs methods based on event register menu.
* **runEventUpdateMenu()** – Runs methods based on event register menu.
* **adminMenu()** – Admin options menu.
* **runAdminMenu()** – Runs methods based on admin options menu.
* **repeatInterest()** – Asks the user for input, creates a special interest, then adds it to the interestList.
* **noScoutInList()** – Tells the user there are no scouts registered and offers other options.
* **addInterestToScout()** – Adds a special interest to a scout in the ArrayList.
* **addBeaverScout()** – Creates a beaver object and adds it to the ArrayList.
* **addCubScout()** – Creates a cub object and adds it to the ArrayList.
* **addScouter()** – Creates a scouter object and adds it to the ArrayList.
* **addScoutEvent()** – Creates a scout event object and adds it to the ArrayList.
* **addEquipment()** – Creates an equipment object.
* **addAnotherEquip()** – Creates an equipment object and adds it to the specified event.
* **addDonator()** – Creates a donator object.
* **addAnotherDonator()** – Creates a donator object and adds it to the specified event.
* **addFundEvent()** – Creates a fundraise event object and adds it to the ArrayList of events.
* **searchScoutsByName()** – Search for a scout by name.
* **nameFormat()** – Capitalizes the first letter of a String with the rest lower case.
* **addedToRegisterMessage()** – Confirms a scout has been created.
* **updateBeaverScout()** – Collects updated beaver information from the user then updates an existing specified beaver.
* **updateCubScout()** – Collects updated cub information from the user then updates an existing specified cub.
* **updateScouter()** – Collects updated scouter information from the user then updates an existing specified scouter.
* **updateScoutEvent()** – Collects updated scout event information from the user then updates an existing specified scout event.
* **updateFundEvent()** – Collects updated fundraise event information from the user then updates an existing specified fundraise event.
* **calcTotalFees()** – Calculates and displays the total fees collected from all scouts.
* **calcFeesByGroup()** – Calculates and displays fees collected from each scout group.
* **calcFeesByScout()** – Displays each scout and their fee amount.
* **calcTotalEquipCost()** – Calculates and displays all equipment costs.
* **calcTotalLocationCost()** – Calculates and displays all location costs.
* **summaryReport()** – Calculates and displays all relevant membership and financial details.
* **removeScout()** – Removes a scout from the scout ArrayList.
* **removeEvent()** – Removes an event from the event ArrayList.
* **save()** – Saves all ArrayLists to a .xml file.
* **load()** – Loads all ArrayLists from a .xml file.

**25-30/30**