Program 4: 60 points :ArrayList/Generics/Insertion Sort

**Objectives: The focus of this assignment is the creation of an ArrayList Data Structure that allows the use of any type through Generics.**

**Program Description:**

This project will alter the EmployeeManager to use an ArrayList instead of an array

A total of ten classes are required.

ArrayList – Implementation of the ArrayList data structure of Comparables



InvalidEmployeeNumberException (Provided)

InvalidSizeException (Provided)

MaximumCapacityException (Provided)

InvalidCharacterException (From previous assignment)

Employee (Altered from previous assignment)

HourlyEmployee (Altered from previous assignment)

SalaryEmployee (Altered from previous assignment)

CommissionEmployee (Altered from previous assignment)

EmployeeManager (Altered from previous assignment)

EmployeeDriver (Altered from previous assignment)

It is up to you to create the implementation of the ArrayList as well as make the necessary changes to classes from the previous assignment

**UML DIAGRAM AND DISCUSSION FOR ArrayList**

|  |
| --- |
| ArrayList<E extends Comparable<E>> |
| * DEFCAP : int final = 50 * origCap : int * numElements : int * list : Object[] |
| <<constructor>>ArrayList()  <<constructor>>ArrayList(size : int) throws InvalidSizeException  + addItem(item : E) throws MaximumCapacityException  + getItem(index int) : E throws IndexOutOfBoundsException  + setItem(index int, item E) throws IndexOutOfBoundsException  + removeItem(index int) : E throws IndexOutOfBoundsException  + findItem(item E) : int  + isEmpty() : Boolean  + lengthIs() : int  + clear()  + toString() : String  + sort()  - enlarge() throws MaximumCapacityException |

Notes on ArrayList

The ArrayList class implements an ArrayList data structure, capable of handling types that implement Comparable<E>. The logic for most of the methods was covered in class, you’ll have to convert these to take advantage of the Generic type that is passed.

Constructors

The constructor that takes no size will set the origCap to DEFCAP and this will be the initial size.

The constructor that accepts a size parameter must throw an InvalidSizeException if the size passed is greater than the DEFCAP or less than 1.

Methods

public void addItem(E item) throws MaximumCapacityException

Attempts to add a new item to the ArrayList. If the array is at capacity it attempts to enlarge, then adds the item to the end.

public E getItem(int index)

This method is to retrieve an item in the list given an index into the ArrayList. If that index does not exist within the ArrayList an IndexOutOfBoundsException is thrown with the message: “Index out of Range”.

public setItem(int index, E item)

Attempts to place the passed item into the given index. If that index does not exist within the ArrayList an IndexOutOfBoundsException is thrown with the message: “Index out of Range”.

public E removeItem(int index)

Attempts to remove the item at the given index from the ArrayList. If that index does not exist within the ArrayList an IndexOutOfBoundsException is thrown with the message: “Index out of Range”.

public int findItem(E item)

Performs a linear search to fine the item passed. Returns the index of the item if found, returns -1 if not found.

public Boolean isEmpty()

Returns a true if the ArrayList is empty, otherwise false.

public int lengthIs()

Returns the current number of elements in the ArrayList.

public void clear()

Clears contents of the ArrayList, sets size of array to the original capacity.

public String toString()

Returns a String containing all elements in the ArrayList, separated by a new line.

public void sort()

Sorts the contents of the ArrayList using the Insertion Sort.

private void enlarge()

Will attempt to enlarge the ArrayList by the value of origCap. If this would put it past its maximum size defined by DEFCAP it will enlarge to the size of DEFCAP. If it is already at size of DEFCAP it will throw a MaximumCapacityException.

**Any method that declares to throw a Checked Exception MUST be called upon within a try statement or rethrown, even if we know the Exception will not be thrown.**

Changes to Employee

The Employee class must be made to implement the Comparable interface, this will require providing the compareTo() method (changes in bold).

|  |
| --- |
| Employee {abstract} **implements Comparable<Employee>** |
| - String firstName  - String lastName  - char middleInitial  - boolean fulltime  - char gender  - int employeeNum |
| **<<constructor>> Employee (fn : String, ln : String, m : char, g : char, empNum : int, ft : boolean ) throws InvalidEmployeeNumberException**  + getEmployeeNumber() : int  + setEmployeeNumber(empNum : int) **throwsInvalidEmployeeNumberException**  + getFirstName() : String  + getLastName() : String  + setFirstName(fn: String)  + setLastName(ln : String)  + setMiddleI(m : char)  + setGender(g : char)  + equals(e2 : Object) : Boolean  + toString() : String  + calculateWeeklyPay() : double {abstract}  + annualRaise() {abstract}  + holidayBonus() : double {abstract}  + resetWeek() {abstract}  **+ compareTo(Employee e) : int** |

setEmployeeNumber(int empNum)

Instead of the setEmployeeNumber reprompting for a valid Employee Number we will use an Exception to inform the calling method it was an invalid Employee number. If an invalid Employee number is passed to the method throw a new InvalidEmployeeNumberException, passing the invalid number to its constructor.

Constructor

The constructor calls upon setEmployeeNumber which now throws the checked Exception of InvalidEmployeeNumber, it should not be caught, rather declared such that the constructor passes the Exception on to the calling method.

public int compareTo(Employee e)

By implementing the Comparable interface, Employee becomes what is known as a Comparable type. All Comparable types have the compareTo() method which determines the logical ordering of instances of the class. This will be based on an Employee’s Employee Number. If the calling Employee’s number is greater than the passed Employee’s number return 1, if it is less than return -1, if they are equal return 0. The use of this will be necessary for the ArrayList’s sort.

HourlyEmployee, SalaryEmployee, CommissionEmployee

The constructor for Employee throws a checked InvalidEmployeeNumberException. Since we are not going to handle it here, rather pass the Exception on to the driver, the constructors for the subtypes must declare they throw an InvalidEmployeeNumberException.

Changes to EmployeeManager

The EmployeeManager is being updated to utilize an ArrayList as the structure for storing Employees; since an ArrayList does all management of the data the EmployeeManager no longer needs to track the number of Employees, so the currentEmployees data member can be removed. An additional change needs to be made to reflect the change made to Employee. A sort method is also being added (changes in bold).

|  |
| --- |
| EmployeeManager |
| **- employees : ArrayList<Employee>**  - employeeMax : final int = 10  ~~-~~**~~currentEmployees : int~~** |
| **<<constructor>> EmployeeManager()**  **+ addEmployee( type : int, fn : String, ln : String, m : char, g : char, en : int, ft : boolean, amount : double) throws InvalidEmployeeNumberException**  **+ removeEmployee( index : int)**  **+ listAll()**  **+ listHourly()**  **+ listSalary()**  **+ listCommision()**  **+ resetWeek()**  **+ calculatePayout() : double**  **+ getIndex( empNum : int ) : int**  **+ annualRaises()**  **+ holidayBonuses() : double**  **+ increaseHours( index : int, amount : double)**  **+ increaseSales( index : int, amount : double)**  **+ findAllBySubstring(find : String) : ArrayList<Employee>**  - RabinKarp(name : String, find : String) : int  - stringHash(s : String) : int  - charNumericValue(c : char) : int  - RabinKarpHashes(s : String, hashes : int[], pos : int, length : int) : int  - linearSearchRecursive(nameHashes : int[], findHash : int, pos : int) : int  **+ sort()** |

Constructor

The constructor will create an ArrayList of **initial size of 3.** Since the constructor of the ArrayList that takes a size throws a checked InvalidSizeException this will need to be in a try statement. Catching this Exception should then create the ArrayList to the default size using the other constructor.

addEmployee

The constructors of the Employee subtypes throw a checked InvalidEmployeeNumberException. Since we are not going to handle it here, rather pass the Exception on to the driver, the addEmployee method must declare that it throws an InvalidEmployeeNumberException.

We still want to notify the user if an Employee cannot be added due to reaching the employeeMax. Since currentEmployees is no longer a data member this can be done by properly handling the MaximumCapacityException thrown by ArrayList’s addItem() method.

findAllBySubstring

The findAllBySubstring method now returns an ArrayList of Employees.

Other changes to methods

All other methods need to be altered to use the ArrayList instead of the array from the previous iteration.

public void sort()

Simply calls upon the ArrayList’s sort() method.

EmployeeDriver

As we add more functionality to the project the Driver will have to be updated. From here on out these updates must be made by you. A new option for sorting must be added to the Main Menu, make this option 9, moving Exit to option 10.

**Also, alter the menu method such that the user will be reprompted if an InputMismatchException occurs. Add this same behavior anywhere a non-String value is entered.**

Since addEmployee may now throw an InvalidEmployeeNumberException this must be caught in the driver. If it is, this is where the user should now be reprompted for a different EmployeeNumber and retry the call to addEmployee until it does not throw the Exception.

**Other Notes:**

* Classes from the previous assignment will retain the same package structure. The new Exceptions are declared to be in their own package called “exceptions”. The ArrayList will be in a package called “dataStructures”.
* Testing the revised version of the project should be nearly identical to the previous two iterations. In usage the only difference will be the ability to sort and handling InputMismatchExceptions
* IndexOutOfBoundsException belongs to the java.lang package and therefore does not need to be created or imported
* Remember that since the array in the ArrayList is of type Object you will often need to cast the elements to E in order to correctly satisfy return types and use Employee specific methods.
* Compile using: javac –d . \*.java
* At times when dealing with Generics (in our case casting Object references down to our type parameter) causes a compiler warning. This warning can be avoided by adding the following line above the definition of your ArrayList

@SuppressWarnings("unchecked")

**Tested**

As nearly all methods of the EmployeeManager are changed all functionality will be tested.