Zachary Bell

Dr. Roy

CS 1401

Lab 10

**Activity 1**

|  |
| --- |
| **vacationDestination** |
| - cityName  - distance  - moneySpent  - date  - enjoyed |
| + setCityName(): void  + setDistance(): void  + setMoneySpent(): void  + setDate(): void  + setEnjoyed(): void  + getCityName(): String cityName  + getDistance(): int distance  + getMoneySpent(): double moneySpent  + getDate(): int date  + getEnjoyed(): boolean enjoyed  + print(): void |

Class vacationDestination contains five private variables as well as public methods for setting and getting these variables. The class also contains a method called print() that prints relevant information using all the variables in a user friendly manner.

Pseudo-code: **print()**

1. Give cityName, distance, moneySpent, date
2. If (enjoyed is true) give “vacation great”, else give “vacation not great”

**Activity 2**

Pseudo-code: **readFromFile(filename)**

1. Create scanner with filename
2. While scanner has next line
   1. ++counter
3. Reinitialize scanner
4. Create an array vd with (counter / 5) length
5. For i = 0; i < counter / 5; ++i
   1. vd[i] 🡨 new vacationDestination object
   2. vd[i] set cityname to scanner nextline
   3. vd[i] set distance to scanner nextline
   4. vd[i] set moneySpent to scanner nextline
   5. vd[i] set date to scanner nextline
   6. vd[i] set enjoyed to scanner nextline
6. give vd

Pseudo-code: **sortByDate(vacationDestination[] vacations)**

1. for i = 0; i < vacations.length; ++i
   1. for j = i ; j < vacations.length; ++j
      1. if vacations[i] date > vacations[j] date
         1. vacations[i] 🡨🡪 vacations[j]

Pseudo-code: **earliest(vacationDestination[] vacations)**

1. if vacations length is greater than 0
   1. minimum 🡨 vacations[0] get date
2. else {
   1. minimum 🡨 -1
3. for i = 0; i < vacations length; ++i
   1. if vacations[i] get date < minimum
      1. minimum 🡨 vacations[i] get date
4. give minimum

Pseudo-code: **enjoyed(vacationDestination[] vacations)**

1. for i = 0; i < vacations length; ++i
   1. if (vacations[i] get enjoyed is equal to true)
      1. ++enjoyedCount
2. Give enjoyedCount

Pseudo-code: **recEnjoyed(vacationDestination[] vacations, int n)**

1. If n < 0
   1. Give 0
2. Else if vacations[n] get enjoyed equals true
   1. Give 1 + recEnjoyed(vacations, n – 1)
3. Else {
   1. Give recEnjoyed(vacations, n – 1)

Pseudo-code: **betterClose(vacationDestination[] vacations, int dist)**

1. For i = 0; i < vacations length; ++i
   1. If (vacations[i] distance <= dist and vacations[i] enjoyed equals true
      1. ++nearCount
   2. Else if (vacations[i] distance > dist and vacations[i] enjoyed equals true
      1. ++farCount
2. If nearCount >= farCount
   1. Give true
3. Else give false

**Activity 3**

|  |
| --- |
| **vacDestinationLL** |
| - vacation  - next |
| + setVacation():void  + setNext():void  + getVacations(): return vacationDestination vacation  + getNext(): return vacDestinationLL next  + printLL(): void  + sizeLL(): return int counter  + sizeLLR(): return int  + addTail(): void  + addNth(): void |

This class is for creation of linked list nodes that link in a series and contain the vacationDestination object.

Pseudo-code: **printLL()**

1. temp 🡨 this
2. while temp is not equal to null
   1. temp get vacation then print it
   2. temp 🡨 temp getNext()

Pseudo-code: **sizeLL()**

1. counter 🡨 0
2. temp 🡨 this
3. while temp is not equal to null
   1. temp 🡨temp getNext()
   2. ++counter
4. give counter

Pseudo-code: **sizeLLR()**

1. if next is not equal to null
   1. give 1 + next.sizeLLR()
2. else if this is not equal to null
   1. give 1
3. else give 0

Pseudo-code: **removeHead()**

1. temp 🡨 next
2. this vacation 🡨temp.getVacation()
3. this next 🡨 temp.getNext()

Pseudo-code: **addTail()**

1. if this vacation is equal to null
   1. this setVacation(vacation)
2. else
   1. temp 🡨this
   2. while temp getNext() is not equal to null
      1. temp 🡨 temp getNext()
   3. temp setNext(new vacDestinationLL(vacation)

Pseudo-code: **addNth()**

1. if n is equal to 0
   1. temp 🡨new vacDestinationLL(this getVacation(), this getNext())
   2. this setVacation(vacation)
   3. this setNext(temp)
2. else
   1. temp 🡨next
   2. tempPrev 🡨 this
   3. while n > 1 and temp is not equal to null
      1. - - n
      2. tempPrev 🡨temp
      3. temp 🡨temp.getNext()
   4. newVacation 🡨 new vacDestinationLL(vacation)
   5. tempPrev setNext(newVacation)
   6. newVacation setNext(temp)

**Activity 4**

Pseudo-code: **buildLL()**

1. construct empty node newLinkedList 🡨 new vacDestinationLL()
2. for i = 0; i < vacations length; ++i
   1. vewLinkedList addTail ( vacations[i] )
3. give newLinkedList

Pseudo-code: **chronoLLinsert()**

1. call sortByDate on vacations
2. vacationsLL 🡨 buildLL(vactions)
3. temp 🡨 vacationsLL
4. n 🡨 0
5. while temp is not equal to null
   1. if temp vacation getDate() > extraVacation getDate()
      1. break loop
   2. ++n
   3. Temp 🡨 temp getNext()
6. vacationsLL addNth( extraVacation, n)
7. give vacationsLL

**Exceptions**

Illegal Argument Exception – used to indicate that an illegal argument was used in one of the methods.

IndexOutOfBoundsException – used to indicate if one of my indexes in my loop would go out of bounds.

NullPointerException – used to indicate if one of the values that would be expected not to be null is a null value.

FileNotFoundException – used to indicate that a file wasn’t found when trying to locate and read a file.

IOException – indicated an exception in the IO of a file being read from and written to.