Program 3: 50 points : Recursion Due: Thursday, October 15th, 11:59 p.m.

**Objectives:The focus of this assignment is to create and use a recursive method given a moderately difficult problem.**

**Program Description:**

This project will alter the EmployeeManager to add a search feature, allowing the user to find an Employee by a substring of their name. This will be done by implementing the Rabin-Karp algorithm.

A total of seven classes are required.

Employee (From previous assignment)

HourlyEmployee (From previous assignment)

SalaryEmployee (From previous assignment)

CommissionEmployee (From previous assignment)

EmployeeManager (Altered from previous assignment)

EmployeeDriver (Altered from previous assignment, provided)

InvalidCharacterException (provided)

Required changes to Employee Manager in bold

**EmployeeManager**

|  |
| --- |
| EmployeeManager |
| - employees : 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)  + 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) : 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** |

public Employee[] findAllBySubstring(String find)

This method will return an array of all the Employees in the EmployeeManager that contain the substring passed. Create a new Employee array with the size of the number of current Employees. For every Employee call upon the RabinKarp method giving the search string as the concatenation of that Employee’s first and last name (no spaces). If the substring is found in the Employee add that Employee to the new array. After all have been checked, return the array.

private int charNumericValue(char c)

Given a character, returns the numeric value of the character, starting with A – 0 up to Z – 25. This should treat upper and lower case the same; that is passing it ‘A’ will return 0, passing it ‘a’ will also return 0. If a letter is not passed this method should create and throw an InvalidCharacterException as provided.

private int stringHash(String s)

Given a string, return the hash value of the entire String. Use a base 26 number system to create the hash as described in class. This will be needed only to find the hash of the substring that is being searched for and the base case for finding all substring hashes in the search string.

private int RabinKarpHashes(String s, int[] hashes, int pos, int length)

Finds the hash values of all substrings of size length in the String s, starting at index pos and down. These values are stored in the passed hashes array. **This method must be recursive, using the technique as described in the Rabin-Karp lecture.**

private int linearSearchRecursive(int[] data, int key, int pos)

This is a recursive linear search. Return the position of key in the data array, or -1 if it is not present. **This method must be recursive.**

private int RabinKarp(String name, String find)

Does the preprocessing of finding the hash for the substring, find using the stringHash method and the hashes of substrings in the search string using RabinKarpHashes method. Calls upon linearSearchRecursive to determine if the substring hash is in the collection of hashes and returns the result.

**Other Notes:**

* Classes from the previous assignment will retain the same package structure. The new Exception is declared to be in its own package called “exceptions”.
* Your EmployeeManager must import the InvalidCharacterException in order to use it
* Compile using: javac –d . \*.java

Examples given the following Employees

12348

Doe, John A.

Gender: M

Status: Full Time

Wage: 14.75

Hours Worked: 0.00

12345

Cavanaugh, Patrick B.

Gender: M

Status: Full Time

Wage: 10.00

Hours Worked: 0.00

12346

Bohning, Edith C.

Gender: F

Status: Part Time

Salary: 50000.00

12347

Pohnal, Rickie A.

Gender: F

Status: Full Time

Rate: 2.50

Sales: 0.00

**Search Test 1**

1. Employee Submenu

2. Add Employee

3. Remove Employee

4. Calculate Weekly Payout

5. Calculate Bonus

6. Annual Raises

7. Reset Week

8. Find Employee

9. Quit

Enter Choice: 8

Enter substring of Employee name: ohn

Matches found:

12348

Doe, John A.

Gender: M

Status: Full Time

Wage: 14.75

Hours Worked: 0.00

12346

Bohning, Edith C.

Gender: F

Status: Part Time

Salary: 50000.00

12347

Pohnal, Rickie A.

Gender: F

Status: Full Time

Rate: 2.50

Sales: 0.00

**Search Test 2**

1. Employee Submenu

2. Add Employee

3. Remove Employee

4. Calculate Weekly Payout

5. Calculate Bonus

6. Annual Raises

7. Reset Week

8. Find Employee

9. Quit

Enter Choice: 8

Enter substring of Employee name: Rick

Matches found:

12345

Cavanaugh, Patrick B.

Gender: M

Status: Full Time

Wage: 10.00

Hours Worked: 0.00

12347

Pohnal, Rickie A.

Gender: F

Status: Full Time

Rate: 2.50

Sales: 0.00

**Search Test 3 (No matches)**

1. Employee Submenu

2. Add Employee

3. Remove Employee

4. Calculate Weekly Payout

5. Calculate Bonus

6. Annual Raises

7. Reset Week

8. Find Employee

9. Quit

Enter Choice: 8

Enter substring of Employee name: Beth

Matches found:

**Search Test 4 (Invalid Character)**

1. Employee Submenu

2. Add Employee

3. Remove Employee

4. Calculate Weekly Payout

5. Calculate Bonus

6. Annual Raises

7. Reset Week

8. Find Employee

9. Quit

Enter Choice: 8

Enter substring of Employee name: Jo\*

Invalid character found in search

**In all cases the user is returned to the Main Menu after resolving the search**

**Summary of Files**

You will need to seven (7) files for this assignment.

* Employee.java
* HourlyEmployee.java
* SalaryEmployee.java
* CommissionEmployee.java
* EmployeeManager.java
* EmployeeDriver.java (provided)
* InvalidCharacterException.java (provided)

**Required Elements:**

* Your program file must meet the programming standards defined for this course and contain the appropriate header defined for this course.
* Document your methods

# **Submitting Your Program Files:**

The assignment is automatically collected from your account on the “Loki” machine on the due date. You must put your source code in the correct directory in your account in order for this to work. Your directory name includes the course, section, semester and the assignment number. This is case-sensitive; **all alphas are upper-case**. This directory must be created in your home directory.

Submission Directory:

**CSCI-1620-1-F15-A3**