ASSIGNMENT –9;’

DATE :23-10-2024

REG NO:2022503045

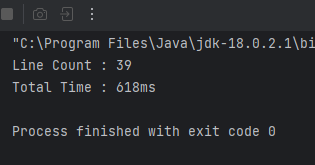
1. Create a program that copies a text file using both byte streams and character streams. Compare the performance.

(i) Line counting

Code(ByteStream):

import java.io.FileInputStream;  
import java.io.FileOutputStream;  
  
public class ByteStream\_3045 {  
 public static void main(String[] args){  
 long startTime=System.*currentTimeMillis*();  
 try(FileInputStream in=new FileInputStream("input.txt"); FileOutputStream out=new FileOutputStream("output.txt");){  
 int c;  
 int line\_count=0;  
 while((c=in.read())!=-1){  
 if((char)c=='\n')  
 line\_count++;  
 out.write(c);  
 }  
  
 System.*out*.println("Line Count : "+line\_count);  
 }  
 catch(Exception e){  
 System.*out*.println("exception caught");  
 }  
 long TotalTime=System.*currentTimeMillis*()-startTime;  
 System.*out*.println("Total Time : "+TotalTime+"ms");  
 }  
}

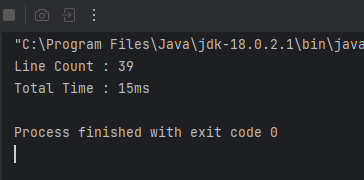
Output :



Code(Charstream):

import java.io.FileReader;  
import java.io.FileWriter;  
  
public class charStream\_3045 {  
 public static void main(String[] args){  
 long startTime=System.*currentTimeMillis*();  
 try(FileReader in=new FileReader("input.txt"); FileWriter out=new FileWriter("output.txt");){  
 int c;  
 int line\_count=0;  
 while((c=in.read())!=-1){  
 if((char)c=='\n')  
 line\_count++;  
 out.write(c);  
 }  
  
 System.*out*.println("Line Count : "+line\_count);  
 }  
 catch(Exception e){  
 System.*out*.println("exception caught");  
 }  
 long TotalTime=System.*currentTimeMillis*()-startTime;  
 System.*out*.println("Total Time : "+TotalTime+"ms");  
 }  
}

Output:



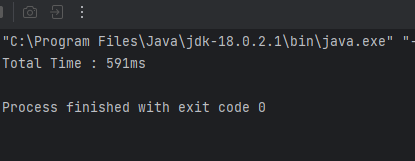
Thus char Stream total time is less than byteStream by 613ms .character Stream performance is high.

(ii)uppercase

Code(Bytestream):

import java.io.FileInputStream;  
import java.io.FileOutputStream;  
  
public class ByteStream\_3045 {  
 public static void main(String[] args){  
 long startTime=System.*currentTimeMillis*();  
 try(FileInputStream in=new FileInputStream("input.txt"); FileOutputStream out=new FileOutputStream("output.txt");){  
 int c;  
  
 while((c=in.read())!=-1){  
 char t=(char)c;  
 out.write(Character.*toUpperCase*(t));  
 }  
  
  
 }  
 catch(Exception e){  
 System.*out*.println("exception caught");  
 }  
 long TotalTime=System.*currentTimeMillis*()-startTime;  
 System.*out*.println("Total Time : "+TotalTime+"ms");  
 }  
}

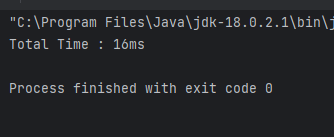
Output:



Code(char Stream):

import java.io.FileReader;  
import java.io.FileWriter;  
  
public class charStream\_3045 {  
 public static void main(String[] args){  
 long startTime=System.*currentTimeMillis*();  
 try(FileReader in=new FileReader("input.txt"); FileWriter out=new FileWriter("output.txt");){  
 int c;  
 while((c=in.read())!=-1){  
  
 out.write(Character.*toUpperCase*((char)c));  
 }  
  
 }  
 catch(Exception e){  
 System.*out*.println("exception caught");  
 }  
  
 long TotalTime=System.*currentTimeMillis*()-startTime;  
 System.*out*.println("Total Time : "+TotalTime+"ms");  
 }  
}

Output:



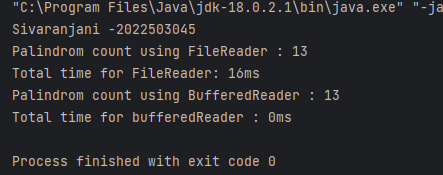
2. Enhance the previous program by using BufferedInputStream /BufferedOutputStream and BufferedReader/BufferedWriter. Compare the performance with non-buffered versions.

(i) palindrome count.

Code:

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
  
  
public class palindrome\_buffer {  
 public void palindromBuffer(){  
 int palindromCount=0;  
 long startTime=System.*currentTimeMillis*();  
 try(FileReader f=new FileReader("input.txt"); BufferedReader bf=new BufferedReader(f);){  
 String str;  
  
 while((str=bf.readLine())!=null){  
 String[] words=str.split(" ");  
 for(String word: words){  
 if(isPalindrome(word.toLowerCase())){  
 palindromCount++;  
  
 }  
 }  
 }  
 }  
 catch(Exception e){  
 System.*out*.println("exception caught : "+e);  
 }  
 long TotalTime=System.*currentTimeMillis*()-startTime;  
 System.*out*.println("Palindrom count using BufferedReader : "+palindromCount);  
 System.*out*.println("Total time for bufferedReader : "+TotalTime+"ms");  
 }  
 public void palindromWithoutBuffer(){  
 int palindromCount=0;  
 long startTime=System.*currentTimeMillis*();  
 try(FileReader f=new FileReader("input.txt");){  
 int c;  
 StringBuilder sb=new StringBuilder();  
  
 while((c=f.read())!=-1){  
 if((char)c!=' ' && (char)c!='\n'){  
 sb.append((char)c);  
 }  
 else{  
 if(isPalindrome(sb.toString().toLowerCase())){  
 palindromCount++;  
   
 }  
 sb=new StringBuilder();  
 }  
  
  
 }  
 }  
 catch(Exception e){  
 System.*out*.println("exception caught : "+e);  
 }  
 long TotalTime=System.*currentTimeMillis*()-startTime;  
 System.*out*.println("Palindrom count using FileReader : "+palindromCount);  
 System.*out*.println("Total time for FileReader: "+TotalTime+"ms");  
  
 }  
 public boolean isPalindrome(String str){  
 int start=0;  
 int end=str.length()-1;  
  
 while(start<=end){  
 if(str.charAt(start)!=str.charAt(end))  
 return false;  
 start++;  
 end--;  
 }  
 return true;  
 }  
 public static void main(String[] args){  
 palindrome\_buffer obj=new palindrome\_buffer();  
 System.*out*.println("Sivaranjani -2022503045");  
 obj.palindromBuffer();  
 obj.palindromWithoutBuffer();  
 }  
}

OUTPUT :



(ii)encrypt the text

Code:

import java.io.BufferedInputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.BufferedOutputStream;

public class encrypt\_buffer\_3045 {

public void encryptBuffer(){

long startTime=System.currentTimeMillis();

try(FileInputStream f=new FileInputStream("input.txt"); BufferedInputStream bf=new BufferedInputStream(f); FileOutputStream out=new FileOutputStream("output.txt"); BufferedOutputStream outb=new BufferedOutputStream(out); ){

int ch;

while((ch=bf.read())!=-1){

outb.write(ch+5);

}

}

catch(Exception e){

System.out.println("exception caught : "+e);

}

long TotalTime=System.currentTimeMillis()-startTime;

System.out.println("Total time for bufferedInput/OutputStream : "+TotalTime+"ms");

}

public void encryptWithoutBuffer(){

long startTime=System.currentTimeMillis();

try(FileInputStream f=new FileInputStream("input.txt"); FileOutputStream out=new FileOutputStream("output.txt");){

int c;

while((c=f.read())!=-1){

out.write(c+7);

}

}

catch(Exception e){

System.out.println("exception caught : "+e);

}

long TotalTime=System.currentTimeMillis()-startTime;

System.out.println("Total time for FileInput/outputstream: "+TotalTime+"ms");

}

public static void main(String[] args){

encrypt\_buffer\_3045 obj=new encrypt\_buffer\_3045();

System.out.println("Sivaranjani -2022503045");

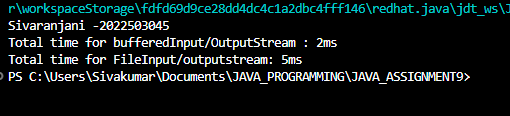
obj.encryptBuffer();

obj.encryptWithoutBuffer();

}

}

Output:



3. Create a program that reads a file and counts the occurrence of a specific character using InputStreamReader.

Input :

import java.io.InputStreamReader;

import java.util.Scanner;

import java.io.FileInputStream;

public class Count\_characters\_3045 {

public int count\_character(char c){

int count=0;

try(FileInputStream f=new FileInputStream("input.txt"); InputStreamReader in\_str=new InputStreamReader(f);){

int t;

count=0;

while((t=in\_str.read())!=-1){

char ch=(char)t;

if(ch==c){

count+=1;

}

}

}

catch(Exception e){

System.out.println("Exception caught :"+e);

}

return count;

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Sivaranjani - 2022503045");

System.out.println("enter the character to count : ");

char c=sc.nextLine().charAt(0);

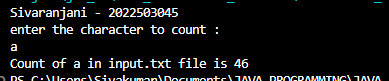
Count\_characters\_3045 obj=new Count\_characters\_3045();

System.out.println("Count of "+c+" in input.txt file is "+obj.count\_character(c));

}

}

Output:



CODE:

import java.io.\*;

import java.util.ArrayList;

import java.util.\*;

class Employee implements Serializable {

public String name;

public int id;

public double salary;

public String designation;

public Employee(String name, int id, double salary, String designation) {

this.name = name;

this.id = id;

this.salary = salary;

this.designation = designation;

}

public void incrementSalary() {

switch (this.designation) {

case "Project-Manager":

this.salary += this.salary \* 0.10; // 10% increase

break;

case "Software-Engineer":

this.salary += this.salary \* 0.08; // 8% increase

break;

case "Data-Analyst":

this.salary += this.salary \* 0.05; // 5% increase

break;

default:

System.out.println("No salary increment for " + this.designation);

break;

}

}

@Override

public String toString() {

return "ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;

}

}

class EmployeeManager {

private ArrayList<Employee> employees = new ArrayList<>();

public void addEmployee(Scanner sc) {

System.out.println("Enter employee details (name, ID, salary, designation):");

String name = sc.next();

int id = Integer.parseInt(sc.next());

double salary = Double.parseDouble(sc.next());

//sc.nextLine(); // Consume newline

String designation = sc.nextLine();

Employee e = new Employee(name, id, salary, designation);

employees.add(e);

System.out.println("Employee added successfully.");

}

public void deleteEmployee(Scanner sc) {

System.out.println("Enter employee ID to delete:");

int id = sc.nextInt();

employees.removeIf(emp -> emp.id == id);

System.out.println("Employee deleted successfully.");

}

public void updateSalary(Scanner sc) {

System.out.println("Enter employee ID to update salary:");

int id = sc.nextInt();

System.out.println("Enter new salary:");

double newSalary = sc.nextDouble();

for (Employee emp : employees) {

if (emp.id == id) {

emp.salary = newSalary;

System.out.println("Salary updated.");

return;

}

}

System.out.println("Employee not found.");

}

public void displayByDesignation(Scanner sc) {

System.out.println("Enter designation to display:");

sc.nextLine(); // Consume newline

String designation = sc.nextLine();

for (Employee emp : employees) {

if (emp.designation.equalsIgnoreCase(designation)) {

System.out.println(emp);

}

}

}

public void applySalaryIncrement() {

for (Employee emp : employees) {

emp.incrementSalary();

}

System.out.println("Salary increments applied based on designations.");

}

public void saveToFile(String fileName) throws IOException {

ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fileName));

oos.writeObject(employees);

oos.close();

System.out.println("Employees saved to file.");

}

public void loadFromFile(String fileName) throws IOException, ClassNotFoundException {

ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fileName));

employees = (ArrayList<Employee>) ois.readObject();

ois.close();

System.out.println("Employees loaded from file.");

}

}

public class Employee\_3045 {

1

public static void main(String[] args) throws Exception {

Scanner sc = new Scanner(System.in);

EmployeeManager manager = new EmployeeManager();

try {

manager.loadFromFile("employees.dat");

} catch (Exception e) {

System.out.println("No existing employee data found."+e);

}

while (true) {

System.out.println("\n1. Add Employee\n2. Delete Employee\n3. Update Salary\n4. Display Employees by Designation\n5. Apply Salary Increment\n6. Save & Exit\nChoose an option: ");

int choice = sc.nextInt();

switch (choice) {

case 1:

manager.addEmployee(sc);

break;

case 2:

manager.deleteEmployee(sc);

break;

case 3:

manager.updateSalary(sc);

break;

case 4:

manager.displayByDesignation(sc);

break;

case 5:

manager.applySalaryIncrement();

break;

case 6:

manager.saveToFile("employees.dat");

System.out.println("Exiting...");

sc.close();

return;

default:

System.out.println("Invalid choice.");

}

}

}

}

OUTPUT:

