<?xml version="1.0" encoding="UTF-8"?>

<classpath>

<classpathentry kind="con" path="org.eclipse.jdt.launching.JRE\_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-19">

<attributes>

<attribute name="module" value="true"/>

</attributes>

</classpathentry>

<classpathentry kind="src" path="src"/>

<classpathentry kind="output" path="bin"/>

</classpath>

<?xml version="1.0" encoding="UTF-8"?>

<projectDescription>

<name>NoteCount</name>

<comment></comment>

<projects>

</projects>

<buildSpec>

<buildCommand>

<name>org.eclipse.jdt.core.javabuilder</name>

<arguments>

</arguments>

</buildCommand>

</buildSpec>

<natures>

<nature>org.eclipse.jdt.core.javanature</nature>

</natures>

</projectDescription>

package com.gl.NoteCount.service;

public class MergeSortImplementation {

void merge(int arr[],int left,int mid,int right) {

int n1=mid-left+1;

int n2=right-mid;

int leftArray[]=new int[n1];

int rightArray[]=new int[n2];

for(int i=0;i<n1;i++)

leftArray[i]=arr[left+i];

for(int j=0;j<n1;j++)

rightArray[j]=arr[mid+1+j];

int i=0, j=0;

int k=left;

while(i<n1 && j<n2) {

if(leftArray[i]>=rightArray[j]) {

arr[k] = leftArray[i];

i++;

}else {

arr[k] = rightArray[j];

j++;

}

k++;

}

while(i<n1) {

arr[k] = leftArray[i];

i++;

k++;

}

while(j<n2) {

arr[k] = rightArray[j];

j++;

k++;

}

}

public void sort(int[] notes, int left, int right) {

if(left < right) {

int mid = (left+right)/2;

sort(notes, left, mid);

sort(notes, mid+1, right);

merge(notes, left, mid, right );

}

}

}

package com.gl.NoteCount;

import java.util.Scanner;

import com.gl.NoteCount.service.MergeSortImplementation;

import com.gl.NoteCount.service.NotesCount;

public class NoteCountDriver {

public static void main(String[] args) {

MergeSortImplementation mergeSortImplementation = new MergeSortImplementation();

NotesCount notesCount = new NotesCount();

System.out.println("enter the size of currency denominations");

Scanner sc= new Scanner(System.in);

int size = sc.nextInt();

int[] notes = new int[size];

System.out.println("enter the currency denominations value");

for(int i=0;i<size;i++) {

notes[i] = sc.nextInt();

}

System.out.println("enter the amount you want to pay");

int amount = sc.nextInt();

mergeSortImplementation.sort(notes,0,notes.length-1);

notesCount.notesCountImplementation(notes, amount);

sc.close();

}

}

package com.gl.NoteCount.service;

public class NotesCount {

public void notesCountImplementation(int notes[], int amount) {

int[] noteCounter = new int[notes.length];

try {

for(int i=0; i< notes.length;i++) {

if(amount>= notes[i]) {

noteCounter[i] = amount / notes[i];

amount = amount - noteCounter[i]\*notes[i];

if(amount == 0)

break;

}

}

if(amount > 0) {

System.out.println("exact amount cannot be given with the highest denomination");

}else {

System.out.println("your payment approach in order to give min no of notes will be:");

for(int i=0; i< notes.length; i++) {

if(noteCounter[i]!=0) {

System.out.println(notes[i]+":" + noteCounter[i]);

}

}

}

}catch(ArithmeticException e) {

System.out.println(e+ "notes of denominator 0 is invalid");

}

}

}

/\*\*

\*

\*/

/\*\*

\*

\*/

module NoteCount {

}

eclipse.preferences.version=1

encoding/<project>=UTF-8

eclipse.preferences.version=1

org.eclipse.jdt.core.compiler.codegen.inlineJsrBytecode=enabled

org.eclipse.jdt.core.compiler.codegen.targetPlatform=19

org.eclipse.jdt.core.compiler.codegen.unusedLocal=preserve

org.eclipse.jdt.core.compiler.compliance=19

org.eclipse.jdt.core.compiler.debug.lineNumber=generate

org.eclipse.jdt.core.compiler.debug.localVariable=generate

org.eclipse.jdt.core.compiler.debug.sourceFile=generate

org.eclipse.jdt.core.compiler.problem.assertIdentifier=error

org.eclipse.jdt.core.compiler.problem.enablePreviewFeatures=disabled

org.eclipse.jdt.core.compiler.problem.enumIdentifier=error

org.eclipse.jdt.core.compiler.problem.reportPreviewFeatures=warning

org.eclipse.jdt.core.compiler.release=enabled

org.eclipse.jdt.core.compiler.source=19