Problem1

**package** problem1;

**public** **class** Circle **extends** Shape{

**double** radius;

**public** Circle(String color, **double** radius) {

**super**(color);

**this**.radius = radius;

}

@Override

**double** calcualteArea() {

**return** Math.***PI***\***this**.radius \* **this**.radius ;

}

@Override

**double** calculatePerimeter() {

**return** 2\*Math.***PI***\***this**.radius ;

}

}

**package** problem1;

**public** **class** Rectengle **extends** Shape {

**double** width;

**double** height;

**public** Rectengle(**double** width, **double** height,String color ) {

**super**(color);

**this**.width = width;

**this**.height = height;

}

@Override

**double** calcualteArea() {

**return** **this**.width \* **this**.height;

}

@Override

**double** calculatePerimeter() {

**return** 2\***this**.width + 2\* **this**.height;

}

}

**package** problem1;

**public** **class** Shape {

String color;

Shape(String color)

{

**this**.color=color;

}

**double** calcualteArea() {

**return** 0.0 ;

}

**double** calculatePerimeter() {

**return** 0.0 ;

}

}

**package** problem1;

**public** **class** Square **extends** Rectengle {

**public** Square(**double** side, String color) {

**super**(side, side, color);

// **TODO** Auto-generated constructor stub

}

@Override

**double** calcualteArea() {

**return** **this**.width \* **this**.height;

}

@Override

**double** calculatePerimeter() {

**return** 4\***this**.width ;

}

}

**package** problem1;

**public** **class** Solution1 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Shape[] database = {

**new** Shape("Red"),

**new** Rectengle(45,21,"gray"),

**new** Circle("Black",5),

**new** Circle("Brown",51),

**new** Square(3,"Blue"),

**new** Rectengle(84,23,"Yellow"),

};

*printTotal*(database);

}

**public** **static** **void** printTotal(Shape[] shapes)

{

**int** index = 0;

**for** (Shape shape : shapes)

{

String className = shape.getClass().getName();

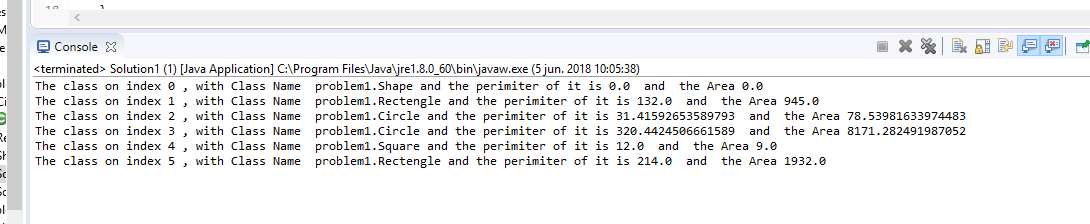
System.***out***.printf("The class on index %s , with Class Name %s and the perimiter of it is %s and the Area %s \n",index,className , shape.calculatePerimeter() , shape.calcualteArea());

index=index+1;

}

}

}



Problem2

**package** problem2;

**import** java.time.LocalDate;

**public** **class** DeptEmployee {

String name ;

LocalDate hireDate;

**double** salary ;

**public** DeptEmployee(String name, **int** yearOfHire, **int** monthOfHire, **int** dayOfHire,**double** salary)

{

**this**.name = name;

**this**.hireDate = LocalDate.*of*(yearOfHire, monthOfHire, dayOfHire);

**this**.salary=salary;

}

**public** **double** computeSalary()

{

**return** **this**.salary;

}

}

**package** problem2;

**public** **class** Professor **extends** DeptEmployee {

**int** numberOfPublications;

**public** Professor(String name, **int** yearOfHire, **int** monthOfHire, **int** dayOfHire, **double** salary,

**int** numberOfPublications)

{

**super**(name, yearOfHire, monthOfHire, dayOfHire, salary);

**this**.numberOfPublications = numberOfPublications;

}

**public** **int** getNumberOfPublications() {

**return** numberOfPublications;

}

**public** **void** setNumberOfPublications(**int** numberOfPublications) {

**this**.numberOfPublications = numberOfPublications;

}

}

**package** problem2;

**public** **class** Secretary **extends** DeptEmployee {

**double** overtimeHours;

**public** Secretary(String name, **int** yearOfHire, **int** monthOfHire, **int** dayOfHire, **double** salary) {

**super**(name, yearOfHire, monthOfHire, dayOfHire, salary);

**this**.overtimeHours = 0;

}

**public** **double** getOvertimeHours() {

**return** overtimeHours;

}

**public** **void** setOvertimeHours(**double** overtimeHours)

{

**this**.overtimeHours = overtimeHours;

}

@Override

**public** **double** computeSalary()

{

**return** **this**.salary + 12\***this**.overtimeHours;

}

}

package problem2;

import java.util.Arrays;

import java.util.stream.Collectors;

import javax.swing.JOptionPane;

public class SolutionProblem2 {

public static void main(String[] args) {

//three instances of Professor

Professor manuel =new Professor("Manuel", 2011, 1, 23, 21000,1);

Professor roberto =new Professor("Roberto", 2012, 12, 23, 12000,2);

Professor edson =new Professor("Edson", 2018, 11, 23,13000,3);

// two instances of Secretary

Secretary sarita =new Secretary("Sarita", 1988, 3, 1, 2200);

sarita.setOvertimeHours(23);

Secretary rosita =new Secretary("Rosita", 2021, 5, 1, 3200);

rosita.setOvertimeHours(12);

DeptEmployee[] department = new DeptEmployee[5];

department[0]=manuel;

department[1]=roberto;

department[2]=edson;

department[3]=sarita;

department[4]=rosita;

double profesors = Arrays.stream(department).filter(o->o.getClass().getSimpleName().equals("Professor")).mapToDouble(o->o.computeSalary()).sum();

double secretaries = Arrays.stream(department).filter(o->o.getClass().getSimpleName().equals("Secretary")).mapToDouble(o->o.computeSalary()).sum();

double all = Arrays.stream(department).mapToDouble(o->o.computeSalary()).sum();

String option = JOptionPane.showInputDialog("What do you want to do ? \nWrite 1 if you like know the sum of database profesor\n"

+ "Write 2 if you like know the sum of database Secretary \n"

+ "Write 3 if you like know the sum of database All\nTo exits other wise");

if (option != null )

{

switch (option)

{

case "1":

JOptionPane.showMessageDialog(null, "The result to all profesors " + String.valueOf(profesors), "",1);

break;

case "2":

JOptionPane.showMessageDialog(null, "The result to all secreataries " + String.valueOf(secretaries), "",1);

break;

case "3":

JOptionPane.showMessageDialog(null, "The result to all profesors " + String.valueOf(all), "",1);

break;

default:

JOptionPane.showMessageDialog(null, "Exits without options " , "",1);

break;

}

}

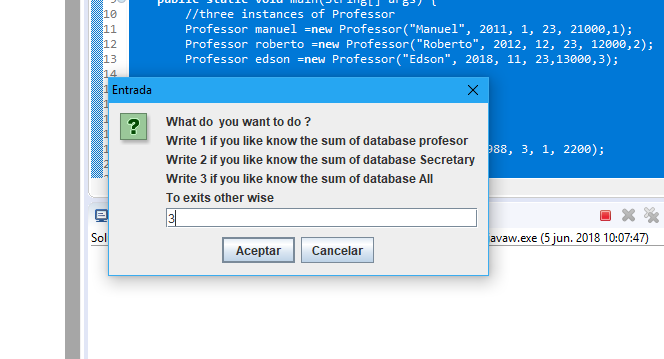
else {

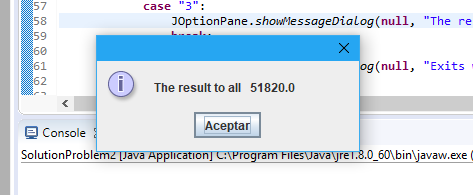
JOptionPane.showMessageDialog(null, "Exits without options " , "",1);

}

}

}





Problem3

**package** Problem3;

**public** **interface** Figure {

**public** **void** getFigure();

}

**package** Problem3;

**public** **class** DownwardHat **implements** Figure {

@Override

**public** **void** getFigure() {

System.***out***.print("\\/");

}

}

**package** Problem3;

**public** **class** FaceMaker **implements** Figure {

@Override

**public** **void** getFigure() {

System.***out***.print(":)");

}

}

**package** Problem3;

**public** **class** UpwardHat **implements** Figure {

@Override

**public** **void** getFigure() {

// **TODO** Auto-generated method stub

System.***out***.print("/\\");

}

}

**package** Problem3;

**public** **class** Vertical **implements** Figure {

@Override

**public** **void** getFigure() {

// **TODO** Auto-generated method stub

System.***out***.print("||");

}

}

**package** Problem3;

**public** **class** Solution3 {

**public** **static** **void** main(String[] args) {

Figure[] database = { **new** UpwardHat(), **new** UpwardHat(), **new** DownwardHat(), **new** FaceMaker(), **new** Vertical(),

};

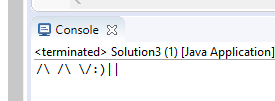
**for** (Figure figure : database) {

figure.getFigure();

}

}

}



Problem4

**package** Problem4;

**public** **class** BasePlusCommissionEmployee **extends** CommissionEmployee {

**double** baseSalary;

**public** BasePlusCommissionEmployee(String firtsName, String lastName, String socialSecurityNumber, **double** grossSales,

**double** commissionRate, **double** baseSalary) {

**super**(firtsName, lastName, socialSecurityNumber, grossSales, commissionRate);

**this**.baseSalary = baseSalary;

}

@Override

**public** String toString() {

**return** "BasePlusCommissionEmployee [baseSalary=" + baseSalary + ", grossSales=" + grossSales

+ ", commissionRate=" + commissionRate + ", firtsName=" + firtsName + ", lastName=" + lastName

+ ", socialSecurityNumber=" + socialSecurityNumber + "]";

}

@Override

**public** **double** getPayment() {

**return** **this**.baseSalary + (**this**.grossSales \* **this**.commissionRate);

}

}

**package** Problem4;

**public** **class** CommissionEmployee **extends** Employee {

**double** grossSales;

**double** commissionRate;

**public** CommissionEmployee(String firtsName, String lastName, String socialSecurityNumber, **double** grossSales,

**double** commissionRate) {

**super**(firtsName, lastName, socialSecurityNumber);

**this**.grossSales = grossSales;

**this**.commissionRate = commissionRate;

}

@Override

**public** String toString() {

**return** "CommissionEmployee --->grossSales=" + grossSales + ", commissionRate=" + commissionRate + ", firtsName="

+ firtsName + ", lastName=" + lastName + ", socialSecurityNumber=" + socialSecurityNumber + "<----";

}

@Override

**public** **double** getPayment() {

// **TODO** Auto-generated method stub

**return** **this**.grossSales \* **this**.commissionRate;

}

}

**package** Problem4;

**public** **class** Computer {

String manufacturer;

String processor;

**int** ramSize;

**double** processorSpeed;

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + ((processor == **null**) ? 0 : processor.hashCode());

**long** temp;

temp = Double.*doubleToLongBits*(processorSpeed);

result = prime \* result + (**int**) (temp ^ (temp >>> 32));

result = prime \* result + ramSize;

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj) {

**return** **true**;

}

**if** (obj == **null**) {

**return** **false**;

}

**if** (getClass() != obj.getClass()) {

**return** **false**;

}

Computer other = (Computer) obj;

**if** (processor == **null**) {

**if** (other.processor != **null**)

{

**return** **false**;

}

} **else** **if** (!processor.equals(other.processor)) {

**return** **false**;

}

**if** (Double.*doubleToLongBits*(processorSpeed) != Double.*doubleToLongBits*(other.processorSpeed)) {

**return** **false**;

}

**if** (ramSize != other.ramSize) {

**return** **false**;

}

**return** **true**;

}

@Override

**public** String toString() {

**return** "Current Status of Computer --->manufacturer=" + manufacturer + ", processor=" + processor + ", ramSize="

+ ramSize + ", processorSpeed=" + processorSpeed + "<----";

}

**public** **double** computePower() {

// return ramSize multiplied by processorSpeed

**return** **this**.ramSize \* **this**.processorSpeed;

}

**public** **double** getProcessorSpeed() {

**return** processorSpeed;

}

**public** **int** getRamSize() {

**return** ramSize;

}

**public** Computer(String manufacturer, String processor, **int** ramSize, **double** processorSpeed) {

**this**.manufacturer = manufacturer;

**this**.processor = processor;

**this**.ramSize = ramSize;

**this**.processorSpeed = processorSpeed;

}

}

**package** Problem4;

**public** **abstract** **class** Employee {

String firtsName;

String lastName;

String socialSecurityNumber;

**public** **abstract** **double** getPayment();

**public** Employee(String firtsName, String lastName, String socialSecurityNumber) {

**super**();

**this**.firtsName = firtsName;

**this**.lastName = lastName;

**this**.socialSecurityNumber = socialSecurityNumber;

}

}

**package** Problem4;

**public** **class** HourlyEmployee **extends** Employee {

**double** wage;

**double** hours;

@Override

**public** String toString() {

**return** "HourlyEmployee -->wage=" + wage + ", hours=" + hours + ", firtsName=" + firtsName + ", lastName="

+ lastName + ", socialSecurityNumber=" + socialSecurityNumber + "<----";

}

**public** HourlyEmployee(String firtsName, String lastName, String socialSecurityNumber, **double** wage, **double** hours) {

**super**(firtsName, lastName, socialSecurityNumber);

**this**.wage = wage;

**this**.hours = hours;

}

@Override

**public** **double** getPayment() {

// **TODO** Auto-generated method stub

**return** **this**.wage \* **this**.hours;

}

}

**package** Problem4;

**public** **class** SalariedEmployee **extends** Employee {

**double** weeklySalary;

**public** SalariedEmployee(String firtsName, String lastName, String socialSecurityNumber, **double** weeklySalary) {

**super**(firtsName, lastName, socialSecurityNumber);

**this**.weeklySalary = weeklySalary;

}

@Override

**public** **double** getPayment() {

**return** **this**.weeklySalary;

}

}

package Problem4;

import java.text.NumberFormat;

import java.util.Arrays;

public class Solution4 {

public static void main(String[] args) {

Employee eduardo = new CommissionEmployee("Juan ", "Francisco", "14623807", 2500, 0.1);

// System.out.println(eduardo);

Employee jhon = new HourlyEmployee("Juan", "Francisco", "31251721", 50000, 12);

// System.out.println(jhon);

Employee frank = new SalariedEmployee("Frank ", "Salazar", "38487", 3000);

// System.out.println(frank );

Employee alejandra = new BasePlusCommissionEmployee("Alenadra", "Saavedra", "8489823", 8478398, 0.1, 95000);

// System.out.println(alejandra );

Employee[] database = { eduardo, jhon, frank, alejandra };

double total\_salary = Arrays.stream(database).mapToDouble(o -> o.getPayment()).sum();

NumberFormat formatter = NumberFormat.getCurrencyInstance();

String moneyString = formatter.format(total\_salary);

System.out.printf("The Total salary of database of employees is %s \n", moneyString);

}

}



Problem 5

**package** Problem5;

**import** java.util.Objects;

**public** **class** Computer {

String manufacturer;

String processor;

**int** ramSize;

**double** processorSpeed;

@Override

**public** **int** hashCode() {

**int** hash = 547;

hash += Double.*doubleToLongBits*(**this**.processorSpeed);

hash += Double.*doubleToLongBits*(**this**.ramSize);

**return** hash;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj) {

**return** **true**;

}

**if** (obj == **null**) {

**return** **false**;

}

**if** (**this**.getClass() != obj.getClass()) {

**return** **false**;

}

Computer other = (Computer) obj;

**if** (processor == **null**) {

**if** (other.processor != **null**)

{

**return** **false**;

}

}

**return** processor.equals(other.processor) && **this**.processorSpeed == other.processorSpeed

&& ramSize == other.ramSize;

}

@Override

**public** String toString() {

**return** "Current Status of Computer <<<manufacturer=" + manufacturer + ", processor=" + processor + ", ramSize="

+ ramSize + ", processorSpeed=" + processorSpeed + ">>";

}

**public** **double** computePower() {

// return ramSize multiplied by processorSpeed

**return** **this**.ramSize \* **this**.processorSpeed;

}

**public** **double** getProcessorSpeed() {

**return** processorSpeed;

}

**public** **int** getRamSize() {

**return** ramSize;

}

**public** Computer(String manufacturer, String processor, **int** ramSize, **double** processorSpeed) {

**this**.manufacturer = manufacturer;

**this**.processor = processor;

**this**.ramSize = ramSize;

**this**.processorSpeed = processorSpeed;

}

}

**package** Problem5;

**public** **class** Solution5 {

**public** **static** **void** main(String[] args) {

Computer AsusamdX = **new** Computer("Asus", "x348", 1024, 3200);

Computer LenovoamdY = **new** Computer("Lenovo", "x348", 1024, 3200);

System.***out***.printf("The computer AsusamdX and LenovoamdY are %s \n",

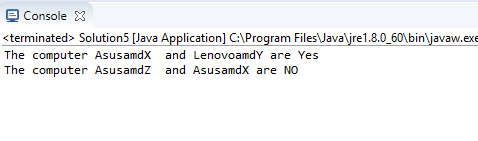
(AsusamdX.equals(LenovoamdY) ? "Yes" : "NO"));

Computer AsusamdZ = **new** Computer("Asus", "x348", 2024, 3200);

System.***out***.printf("The computer AsusamdZ and AsusamdX are %s \n ", (AsusamdX.equals(AsusamdZ) ? "Yes" : "NO"));

}

}



Problem 6.

**package** Problem6;

**public** **class** Computer {

String manufacturer;

String processor;

**int** ramSize;

**double** processorSpeed;

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + ((processor == **null**) ? 0 : processor.hashCode());

**long** temp;

temp = Double.*doubleToLongBits*(processorSpeed);

result = prime \* result + (**int**) (temp ^ (temp >>> 32));

result = prime \* result + ramSize;

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj) {

**return** **true**;

}

**if** (obj == **null**) {

**return** **false**;

}

**if** (getClass() != obj.getClass()) {

**return** **false**;

}

Computer other = (Computer) obj;

**if** (processor == **null**) {

**if** (other.processor != **null**)

{

**return** **false**;

}

} **else** **if** (!processor.equals(other.processor)) {

**return** **false**;

}

**if** (Double.*doubleToLongBits*(processorSpeed) != Double.*doubleToLongBits*(other.processorSpeed)) {

**return** **false**;

}

**if** (ramSize != other.ramSize) {

**return** **false**;

}

**return** **true**;

}

@Override

**public** String toString() {

**return** "Current Status of Computer [manufacturer=" + manufacturer + ", processor=" + processor + ", ramSize="

+ ramSize + ", processorSpeed=" + processorSpeed + "]";

}

**public** **double** computePower() {

// return ramSize multiplied by processorSpeed

**return** **this**.ramSize \* **this**.processorSpeed;

}

**public** **double** getProcessorSpeed() {

**return** processorSpeed;

}

**public** **int** getRamSize() {

**return** ramSize;

}

**public** Computer(String manufacturer, String processor, **int** ramSize, **double** processorSpeed) {

**this**.manufacturer = manufacturer;

**this**.processor = processor;

**this**.ramSize = ramSize;

**this**.processorSpeed = processorSpeed;

}

}

**package** Problem6;

**class** Person **implements** Cloneable {

String name;

Computer computer;

**public** Person(String name, Computer computer) {

**super**();

**this**.name = name;

**this**.computer = computer;

}

**public** Object clone() {

**try** {

Person obj = (Person) **super**.clone();

**return** obj;

} **catch** (CloneNotSupportedException ex) {

System.***out***.println("Fail on cloning");

}

**return** **null**;

}

}

**package** Problem6;

**public** **class** Solution6 {

**public** **static** **void** main(String[] args) {

Computer computer = **new** Computer("Asus", "x3120", 1024, 3200);

Person federico = **new** Person("Federico", computer);

Person leonardo = federico;

Person francisco = (Person) federico.clone();

//// the same addres on memory

System.***out***.println("Leonardo and federicho has a pointer to same adddres ");

System.***out***.println(leonardo);

leonardo.name = "lernardo";

System.***out***.println(leonardo.hashCode());

System.***out***.println(federico.hashCode());

System.***out***.println(leonardo);

System.***out***.print("Who is federico ? ---- >");

System.***out***.println(federico);

System.***out***.println("the next example , francisco has a new addres ");

///

System.***out***.println(francisco.hashCode());

System.***out***.println("Francisco is a diferent object on meomry the values are the same , but the memory pointer is other");

}

}

