|  |
| --- |
| **PROBLEM -1** |
| **package** M1; */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John C Stelly  \* Completion date: 09/24/2020  \*  \* ArrayCommonOperations.txt - template file. Save as ArrayCommonOperations.java  \*  \* This program performs common operations on the elements of an one-dimensional array  \* The elemenst of the array are random numbers of type integer, with values between 1 and 100.  \* The calculated values are stored, and displayed at the end of the program  \*  \* Student tasks: complete tasks specified in the file  \*/* **public class** ArrayCommonOperations {  **public static void** main(String [] args)  {  **int** maxNumber;  **int** minNumber;  **int** sum = 0;  **int** num;  **int** evenCheck;  **int** evenCount = 0;  **int** threeDivCheck;  **int** newThreeNumber;   *//\*\*\* Task #1: Define and instantiate an array of integer numbers, with 10 elements* **int**[] someArray = **new int**[10];   *//\*\*\* Task #2: Fill in the array with integer numbers from 1 to 100* **for** (**int** i = 0; i < someArray.**length**; i++) {  num = (**int**) (Math.*random*() \* 101);  someArray[i] = num;  }   *//\*\*\* Task #3: define another array, named temp, and copy the initial array in it.  //\* This allows to preserve the original array* **int**[] temp = **new int**[10];   *//\*\*\* Task #4: Define the variables you need to calculate the following values,  //\* and initialize them with appropriate values.  //SEE ABOVE* System.***out***.print(**"Original Array: ["**);  **for** (**int** i = 0; i < someArray.**length**; i++) {  num = someArray[i];  *//\*\*\* Task #5: Print the original array* System.***out***.print(num + **" "**);  *//\*\*\* Task #6: Calculate the sum of all values* sum = sum + num;  *//\*\*\* Task #7: Count the number of even values* evenCheck = num % 2;  **if** (evenCheck == 0) {  evenCount++;  }  }  System.***out***.println(**"]"**);   *//\*\*\* Task #8: Calculate the minimum value in the array* minNumber = someArray[0];  **for** (**int** i = 1; i < someArray.**length**; i++) {  **int** test = someArray[i];  **if** (test < minNumber) {  minNumber = test;  }  }   *//\*\*\* Task #9: Calculate the maximum value in the array* maxNumber = someArray[0];  **for** (**int** i = 1; i < someArray.**length**; i++) {  **int** test = someArray[i];  **if** (test > maxNumber) {  maxNumber = test;  }  }   *//\*\*\* Task #10: Replace the elements that are divisible by 3, with their value plus 2* **for** (**int** i = 0; i < someArray.**length**; i++) {  num = someArray[i];  threeDivCheck = num % 3;  **if** (threeDivCheck == 0) {  newThreeNumber = num + 2;  temp[i] = newThreeNumber;  }  **else** {  temp[i] = num;  }  }   *//\*\*\* Task #11: Display the new array after replacement* System.***out***.print(**"Divisible by 3 Array: ["**);  **for** (**int** i = 0; i < temp.**length**; i++) {  System.***out***.print(temp[i] + **" "**);  }  System.***out***.println(**"]"**);   *//\*\*\* Task #12: Display the calculated values.* System.***out***.println(**"Sum of array values in array: "** + sum);  System.***out***.println(**"Number of even values in array: "** + evenCount);  System.***out***.println(**"Minimum value in array: "** + minNumber);  System.***out***.println(**"Maximum value in array: "** + maxNumber);  } } |
| **RESULT**A picture containing bird  Description automatically generated |

|  |
| --- |
| **PROBLEM -2** |
| **package** M1;  /\*\*  \* @author EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  *\* Student name: John C Stelly  \* Completion date: 09/24/2020*  \*  \* Person.txt --> Download and save as Person.java  \*  \*  \* Student tasks: complete tasks specified in the file  \*/  **public class** Person {  *//\*\*\* Task #1: define the instance variables* **private** String **name**;  **private int idNumber**;  **private** String **department**;   *//\*\*\* Task #2: write the no-argument constructor that* **public** Person() {  **this**.**name** = **""**;  **this**.**idNumber** = 0;  **this**.**department** = **""**;  }   *//\*\*\* Task #3: write the constructor that passes values for the name and idNumber* **public** Person(String name, **int** idNumber) {  **this**.**name** = name;  **this**.**idNumber** = idNumber;  **this**.**department** = **""**;  }   *//\*\*\* Task #4: write the constructor that initializes all three instance variables* **public** Person (String name, **int** idNumber, String department){  **this**.**name** = name;  **this**.**idNumber** = idNumber;  **this**.**department** = department;  }   *//\*\*\* Task #5: write accessor method for attribute name* **public** String getName() {  **return name**;  }  **public void** setName(String name) {  **this**.**name** = name;  }   *//\*\*\* Task #7: write accessor method for attribute idNumber* **public int** getIdNumber() {  **return idNumber**;  }   *//\*\*\* Task #8: write mutator method for attribute idNumber* **public void** setIdName(**int** idNumber) {  **this**.**idNumber** = idNumber;  }   *//\*\*\* Task #9: write accessor method for attribute department* **public** String getDepartment() {  **return department**;  }   *//\*\*\* Task #10: write mutator method for attribute department* **public void** setDepartment(String department) {  **this**.**department** = department;  }   *//\*\*\* Task #11: write toString method* @Override  **public** String toString() {  **return " Name: "** + **name** + **" \n ID Number: "** + **idNumber** + **" \n Department: "** + **department** + **"\n"**;  } } |
| */\*\**  *\* @author EMILIA BUTU*  *\* version 1.0*  *\* since 2020-07*  *\**  *\* Student name: John C Stelly  \* Completion date: 09/24/2020*  *\**  *\* PersonDriver.txt --> Download and save as PersonDriver.java*  *\**  *\* This class represents the driver for class Person. The role of it is to test the methods of Person class.*  *\**  *\* Student tasks: complete tasks specified in the file*  *\*/*  **public class** PersonDriver {  **public static void** main(String[] args)  {  *//\*\*\* Task #1: define and instantiate three objects of type Person, using the three constructors* System.***out***.println(**"Initial Information about person \n"**);  Person personOne = **new** Person();  Person personTwo = **new** Person(**"Jane Young"**, 32421);  Person personThree = **new** Person(**"Ella Jones"**, 54231, **"IT"**);   *//\*\*\* Task #2: display the information of the three objects of type Person* System.***out***.println(personOne);  System.***out***.println(personTwo);  System.***out***.println(personThree);   *//\*\*\* Task #3* personOne.setName(**"Jimmy Dean"**);  personOne.setIdName(23123);  personOne.setDepartment(**"Sales"**);  personTwo.setDepartment(**"Marketing"**);   *//\*\*\* Task #4: display the updated information of the three persons* System.***out***.println(**"New Information about person \n"**);  System.***out***.println(personOne);  System.***out***.println(personTwo);  System.***out***.println(personThree);   } } |
| **RESULT:**  A screenshot of a cell phone  Description automatically generated |

**PROBLEM -3**

|  |
| --- |
| */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* Person.java  \*  \* Driver for the inheritance application testing the inheritance, and  \* showing how polymorphism and dynamic binding works.  \*  \* Students have to run the program after building all classes  \*  \*/* **public class** Person {  **private** String **name**;   **public** Person() {  **this**.**name** = **""**;  }   **public** Person(String name) {  **this**.**name** = name;  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public** String toString() {  **return "Name: "** + **name**;  }   } |
| */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* Student.java  \*  \* Driver for the inheritance application testing the inheritance, and  \* showing how polymorphism and dynamic binding works.  \*  \* Students have to run the program after building all classes  \*  \*/* **public class** Student **extends** Person {  **private int studentNumber**;  **public** Student(){  **super**(**""**);  **this**.**studentNumber** = 0;   }  **public** Student(String name, **int** studentNumber) {  **super**(name);  **this**.**studentNumber** = studentNumber;  }  **public int** getStudentNumber() {  **return studentNumber**;  }  **public void** setStudentNumber(**int** studentNumber) {  **this**.**studentNumber** = studentNumber;  }   @Override  **public** String toString() {  **return "Name: "** + **super**.getName() + **" \nStudentNumber: "** + **studentNumber**;  } } |
| */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* Employee.java  \*  \* Driver for the inheritance application testing the inheritance, and  \* showing how polymorphism and dynamic binding works.  \*  \* Students have to run the program after building all classes  \*  \*/* **public class** Employee **extends** Person {  **private int empID**;  **public** Employee() {  }  **public** Employee(String name, **int** empID) {  **super**(name);  **this**.**empID** = empID;  }  **public int** getEmpID() {  **return empID**;  }  **public void** setEmpID(**int** empID) {  **this**.**empID** = empID;  }   @Override  **public** String toString() {  **return "Name: "** + **super**.getName() + **"\nEmployee ID: "** + **empID** + **" "**;  } } |
| */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* Undergraduate.java  \*  \* Driver for the inheritance application testing the inheritance, and  \* showing how polymorphism and dynamic binding works.  \*  \* Students have to run the program after building all classes  \*  \*/* **public class** Undergraduate **extends** Student {  **private** String **major**;  **public** Undergraduate() {  **super**(**""**,0);  **major** = **""**;  }  **public** Undergraduate(String name, **int** studentNumber, String major) {  **super**(name, studentNumber);  **this**.**major** = major;  }   **public** String getMajor() {  **return major**;  }   **public void** setMajor(String major) {  **this**.**major** = major;  }  @Override  **public** String toString() {  **return "Name: "** + **super**.getName() + **"\nStudentNumber: "** + **super**.getStudentNumber() + **"\nUndergraduate Major: "** + **major** ;  } } |
| */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* Faculty.java  \*  \* Driver for the inheritance application testing the inheritance, and  \* showing how polymorphism and dynamic binding works.  \*  \* Students have to run the program after building all classes  \*  \*/* **public class** Faculty **extends** Employee{  **private** String **rank**;  **public** Faculty() {  **this**.**rank** = **""**;  }  **public** Faculty(String name, **int** empID, String rank) {  **super**(name, empID);  **this**.**rank** = rank;  }  **public** String getRank() {  **return rank**;  }  **public void** setRank(String rank) {  **this**.**rank** = rank;  }  @Override  **public** String toString() {  **return "Name: "** + **super**.getName() + **"\nEmployee ID: "** + **super**.getEmpID() + **"\nFaculty Rankrank: "** + **rank** ;  } } |
| **RESULT:**Graphical user interface, text, application  Description automatically generated |
|  |

**PROBLEM -4**

|  |
| --- |
| **package** M1;  */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* EmployeeException.txt - save as EmployeeException.java  \*  \* This class represents the blueprint for instantiating EmployeeException objects,  \* with the following attributes: name: String ssn: String salary: double  \* and methos: A constructor with no arguments that sets the attributes at default values A constructor that passes values for all attributes Accessor, mutator and display methods for each attribute An equals method that has an object of type Employee as argument, and returns true if two employees have the same name, salary and sSN  \*  \* Students have to build the file from scratch  \*/* **public class** EmployeeException {  *//\*\*\* Task #1: define the instance variables* **private** String **name**;  **private** String **ssn**;  **private double salary**;  *//\*\*\* Task #2: define constructor with no arguments* **public** EmployeeException() {  *//this.name = "";  //this.ssn = "";  //this.salary = 0.00;* }  *//\*\*\* Task #3: define constructor passing values for all arguments* **public** EmployeeException(String name, String ssn, **double** salary) {  **this**.**name** = name;  **this**.**ssn** = ssn;  **this**.**salary** = salary;  }  *//\*\*\* Tasks #4: define accessor, mutator, and write out methods for name attribute* **public** String getName() {  **return name**;  }  **public void** setName(String name) {  **this**.**name** = name;  }  **public void** showName () {  System.***out***.println(**"Name: "** + **name**);  }  *//\*\*\* Tasks #5: define accessor, mutator, and write out methods for sSN attribute  // Display the SSN as ddd-dd-dddd* **public** String getSsn() {  **return ssn**;  }  **public void** setSsn(String ssn) {  **this**.**ssn** = ssn;  }  **public void** showSsn () {  System.***out***.println(**"SSN: "** + **ssn**.substring(0,3) + **"-"** + **ssn**.substring(3,5) + **"-"** + **ssn**.substring(5,9));  }  *//\*\*\* Tasks #6: define accessor, mutator, and write out methods for salary attribute* **public double** getSalary() {  **return salary**;  }  **public void** setSalary(**double** salary) {  **this**.**salary** = salary;  }  **public void** showSalary () {  System.***out***.println(**"Salary: $"** + **salary**);  }  *//\*\*\* Tasks #7: Define method writeOutput() that display all information about the employee.* **public void** writeOutput() {  showName();  showSsn();  showSalary();  }  *//\*\*\* Tasks #8: Define equals method, having argument of type EmployeeException* **public boolean** equals(EmployeeException testObject){  **return** (**this**.**name**.equalsIgnoreCase(testObject.**name**)) &&  (**this**.**ssn**.equals(testObject.**ssn**)) &&  (**this**.**salary** == testObject.**salary**);   } } |
| **package** M1;  */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* EmployeeExceptionDriver.java  \*  \* This class represents the driver for the EmployeeException class.  \* The driver program uses an array that can hold up to 100 employees (  \* the array will be of EmployeeException type).  \* The user should be free to enter as many employees as needed.  \* The driver class should use two exception classes to signal the user that the SSN entered is not correct. \* SSN needs to be entered as a 9-digit string without separators.  \*/* **import** java.util.Scanner; **public class** EmployeeExceptionDriver {  **public static void** main(String [] args)  {  *//\*\*\* Task #1: define the variables required for the program* EmployeeException[] employee = **new** EmployeeException[100];  **boolean** enterEmployee = **true**;  String enterEmployeeAnswer;  **int** elementCount = 0;  String name, ssn;  **double** salary;  *//\*\*\* Task #2: define and instantiate variable of type Scanner to be able to read from* Scanner kb1 = **new** Scanner(System.***in***);  Scanner kb2 = **new** Scanner(System.***in***);  Scanner kb3 = **new** Scanner(System.***in***);  *//\*\*\* Task #3: create a loop in which you enter the data for employee.* **try** {  **while**(enterEmployee) {  System.***out***.println(**"[ENTER EMPLOYEE DATA]"**);  *//\*\*\* Task #5: read the name of the employee* System.***out***.print(**"Enter employee name: "**);  name = kb1.nextLine();  *//\*\*\* Task #6: read the salary of the employee* System.***out***.print(**"Enter employee salary: "**);  salary = kb2.nextDouble();  *//\*\*\* Task #7: read SSN using the exceptions blocks* System.***out***.print(**"Enter employee SSN: "**);  ssn = kb1.nextLine();  **if** (ssn.length() != 9)  **throw new** SSNLengthException();  **for** (**int** i = 0; i < ssn.length(); i++) {  **int** temp = ssn.charAt(i);  **if** (temp < 48 || temp > 57)  **throw new** SSNStructureException();  }  *//\*\*\* Task #4: inside the loop, instantiate each element of the array with the constructor* employee[elementCount] = **new** EmployeeException(name, ssn, salary);  elementCount++;  System.***out***.println(**"Continue entering employees? (Y for Yes, or N for No)"**);  *//\*\*\* Task #8: ask the user if there are more employees to enter* enterEmployeeAnswer = kb3.next();  **if**(enterEmployeeAnswer.equalsIgnoreCase(**"y"**)) {  enterEmployee = **true**;  }  **else** {  enterEmployee = **false**;  }  }  } **catch**(SSNLengthException e) {  System.***out***.println(e);  } **catch**(SSNStructureException e) {  System.***out***.println(e);  }  *//\*\*\* Task #9: calculate the average salary* **double** average = *averageSalary*(employee, elementCount);  *//System.out.println(average);  //\*\*\* Task #10: display the information about all employees with a note if their salary  // is above average, under average or average.* System.***out***.println(**"\n"**);  **for** (**int** i = 0; i < elementCount; i++) {  System.***out***.println(**"Employee #"** + (i+1));  employee[i].writeOutput();  **if** (employee[i].getSalary() == average) {  System.***out***.println(**"Average"**);  } **else if** (employee[i].getSalary() > average) {  System.***out***.println(**"Above average"**);  } **else if** (employee[i].getSalary() < average) {  System.***out***.println(**"Below average"**);  }  System.***out***.println();  }  System.***out***.println(**"No more employees."**);  }  **public static double** averageSalary(EmployeeException[] employee, **int** elementCount) {  **double** average;  **double** sum = 0;  **int** count = 0;  **for** (**int** i = 0; i < elementCount; i++) {  sum = sum + employee[i].getSalary();  count++;  }  average = sum / count;  **return** average;  } } |
| **package** M1;  */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* SSNLengthException.java  \*  \* This class represents the driver for the EmployeeException class.  \* The driver program uses an array that can hold up to 100 employees (  \* the array will be of EmployeeException type).  \* The user should be free to enter as many employees as needed.  \* The driver class should use two exception classes to signal the user that the SSN entered is not correct. \* SSN needs to be entered as a 9-digit string without separators.  \*/* **public class** SSNLengthException **extends** Exception{  **public** SSNLengthException() {  **super**(**"Exception: SSN must be 9 digits in length"**);  }  **public** SSNLengthException(String message) {  **super**(message);  } } |
| **package** M1;  */\*\*  \** ***@author*** *EMILIA BUTU  \* version 1.0  \* since 2020-07  \*  \* Student name: John Stelly  \* Completion date: 09/26/2020  \*  \* SSNStructureException.java  \*  \* This class represents the driver for the EmployeeException class.  \* The driver program uses an array that can hold up to 100 employees (  \* the array will be of EmployeeException type).  \* The user should be free to enter as many employees as needed.  \* The driver class should use two exception classes to signal the user that the SSN entered is not correct. \* SSN needs to be entered as a 9-digit string without separators.  \*/* **public class** SSNStructureException **extends** Exception {  **public** SSNStructureException() {  **super**(**"Exception: SSN characters must be digits not letters"**);  }  **public** SSNStructureException(String message) {  **super**(message);  } } |
| **RESULT:Graphical user interface, text, application  Description automatically generated** |
| **Exceptions: Structural** |
| Graphical user interface, text, application, email  Description automatically generated |
| **Length Exception** |
| Graphical user interface, text, application  Description automatically generated |