**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**CHANDUBHAI S PATEL INSTITUTE OF TECHNOLOGY**

**Name:- Abhi Bhimani**

**ID:- 21CE013**

**CSPIT -CE**

|  |  |
| --- | --- |
|  | **Practical -1** |
| **Practical 1.1** | Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument |
| **Ans:-** | * Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument.   🡪Object means a real word entity such as pen, chair, table etc. Object- Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:  ● Object  ● Class  ● Inheritance  ● Polymorphism  ● Abstraction  ● Encapsulation  🡪**Object**: Any entity that has state and behaviour is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.  🡪**Class**: Collection of objects is called class. It is a logical entity.  🡪**Inheritance**: When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.  🡪**Polymorphism**: When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc. In java, we use method overloading and method overriding to achieve polymorphism. Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.  🡪**Abstraction**: Hiding internal details and showing functionality is known as abstraction. For example:  phone call, we don't know the internal processing.  In java, we use abstract class and interface to achieve abstraction.  🡪**Encapsulation**: Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.  A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.  🡪**Difference between JDK, JRE and JVM**  Understanding the difference between JDK, JRE and JVM is important in Java. We are having brief overview of JVM here.  If you want to get the detailed knowledge of Java Virtual Machine, move to the next page. Firstly, let's see the basic differences between the JDK, JRE and JVM.  🡪**JVM**  JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed. JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent. The JVM performs following main tasks:  ● Loads code  ● Verifies code  ● Executes code  ● Provides runtime environment  **🡪JRE**  JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.  Implementations of JVMs are also actively released by other companies besides Sun Micro Systems  🡪**JDK**  JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools. |
| **Practical 1.2** | Given a string, return a string made of the first 2 chars (if present), however include first char only if it is 'o' and include the second only if it is 'z', so "ozymandias" yields "oz".  startOz("ozymandias") → "oz"  startOz("bzoo") → "z"  startOz("oxx") → "o" |
| **Code** | //Problem 1 : Given a string, return a string made of the first 2 chars (if present),  // however include first char only  // if it is 'o' and include the second only if it is 'z', so "ozymandias" yields "oz".  //CREATED BY 21CE013 ABHI BHIMANI  import java.util.Scanner;  public class P1\_2 {      public static String startOz(String str){          if (str.startsWith("oz"))          return "oz";    else if (str.startsWith("o"))          return "o";    else if (str.startsWith("z",1))          return "z";    else          return "";      }      public static void main(String[] args) {          Scanner sc = new Scanner(System.in);          String s1 = sc.next();          String s2 = sc.next();          String s3 = sc.next();          System.out.println(startOz(s1));          System.out.println(startOz(s2));          System.out.println(startOz(s3));          System.out.println("CREATED BY 21CE013 ABHI BHIMANI");      }  }  //deploy on docker as image from practical 1 file we can download it from anywhere ,(ubernetis) |
| **Output** |  |
| **GitHub Link** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.3** | Given two non-negative int values, return true if they have the same last digit, such as with 27 and 57. Note that the % "mod" operator computes remainders, so 17 % 10 is 7.  lastDigit(7, 17) → true  lastDigit(6, 17) → false  lastDigit(3, 113) → true |
| **Code** | ////CREATED BY 21CE013 ABHI BHIMANI  import java.util.Scanner;  public class P1\_3 {      public static void main(String[] args) {          Scanner sc=new Scanner(System.in);          for(int i=0;i<3;i++) {              int a = sc.nextInt();              int b = sc.nextInt();              if (a % 10 == b % 10) {                  System.out.println("true");              }              else {                  System.out.println("false");              }          }          System.out.println("CREATED BY 21CE013 ABHI BHIMANI");      }  } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.4** | Given an array of ints, return true if the sequence of numbers 1, 2, 3 appears in the array somewhere.  array123([1, 1, 2, 3, 1]) → true  array123([1, 1, 2, 4, 1]) → false  array123([1, 1, 2, 1, 2, 3]) → true |
| **Code** | ////CREATED BY 21CE013 ABHI BHIMANI  import java.util.Scanner;  public class P1\_4 {      public static boolean arrOneTeoThree(int []arr){          for(int i=0;i<arr.length-1;i++){              if(arr[i]==1 && arr[i+1]==2 && arr[i+2]==3){                  return true;              }          }return false;      }      public static void main(String[] args) {          int [] arr1 = { 1,1,2,3,1};          int [] arr2 = {1,1,2,4,1};          int [] arr3 = {1,1,2,1,2,3};          System.out.println(arrOneTeoThree(arr1));          System.out.println(arrOneTeoThree(arr2));          System.out.println(arrOneTeoThree(arr3));          System.out.println("CREATED BY 21CE013 ABHI BHIMANI");      }  } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.5** | Given 2 strings, a and b, return the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings. stringMatch("xxcaazz", "xxbaaz") → 3  stringMatch("abc", "abc") → 2  stringMatch("abc", "axc") → 0 |
| **Code** | //Creatd By Abhi Bhimani 21ce013  public class P1\_5  {      public static int stringMatch(String str1, String str2)      {          String s1, s2;          int cnt = 0;          for(int i = 0; i< (str1.length()) - 1; i++)          {              if(i< (str1.length()) - 2)              {                  s1 = str1.substring(i, i + 2);                  s2 = str2.substring(i, i + 2);              }              else              {                  s1 = str1.substring(i);                  s2 = str2.substring(i);              }              if(s1.equals(s2))                  cnt++;          }          return cnt;      }      public static void main(String[] args)      {          System.out.println(stringMatch("xxcaazz", "xxbaaz"));          System.out.println(stringMatch("abc", "abc"));          System.out.println(stringMatch("abc", "axc"));          System.out.println("CREATED BY 21CE013 ABHI BHIMANI");      }  } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.6** | Given an array of strings, return a new array without the strings that are equal to the target string. One approach is to count the occurrences of the target string, make a new array of the correct length, and then copy over the correct strings.  wordsWithout(["a", "b", "c", "a"], "a") → ["b", "c"]  wordsWithout(["a", "b", "c", "a"], "b") → ["a", "c", "a"]  wordsWithout(["a", "b", "c", "a"], "c") → ["a", "b", "a"] |
| **Code** | /\*   \* Created by:- Abhi Bhimani   \* ID:- 21CE013   \* Aim:- Given an array of strings, return a new array without the strings that are equal to   \* the target string. One approach is to count the occurrences of the target string, make   \* a new array of the correct length, and then copy over the correct strings.   \* wordsWithout(["a", "b", "c", "a"], "a") → ["b", "c"]   \* wordsWithout(["a", "b", "c", "a"], "b") → ["a", "c", "a"]   \* wordsWithout(["a", "b", "c", "a"], "c") → ["a", "b", "a"]   \*/  import java.util.\*;  public class Ass6Targetmaqin {      public static void main(String[] args) {          Scanner sc = new Scanner(System.in);          System.out.print("Enter the String: ");          String s1 = sc.nextLine();          System.out.print("Enter the target string: ");          String s2 = sc.nextLine();          int a = s1.length(),b = 0;          for(int i=0;i<a;i++)          {              if(s1.charAt(i)==s2.charAt(0))              b++;          }          char[] target = new char[a-b];          int j = 0;          for(int i=0;i<a;i++)          {              if(s1.charAt(i)!=s2.charAt(0))              {                  target[j]=s1.charAt(i);                  j++;              }          }          System.out.print("The New string is ");          System.out.println(target);          System.out.println("Created By 21CE013 ABHI BHIMANI");      }  } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.7** | Display numbers in a pyramid pattern.  1  1 2 1  1 2 4 2 1  1 2 4 8 4 2 1  1 2 4 8 16 8 4 2 1  1 2 4 8 16 32 16 8 4 2 1  1 2 4 8 16 32 64 32 16 8 4 2 1  1 2 4 8 16 32 64 128 64 32 16 8 4 2 1 |
| **Code** | ////CREATED BY 21CE013 ABHI BHIMANI  public class P1\_7 {      public static void pyramid() {          int ix = 30;          for (int i = 1; i <= 128; i = i \* 2) {  // x is the number printed  //it gets the value from i,              for (int g = ix; g >= 0; g--) {                  System.out.print(" ");              }              for (int x2 = 1; x2 <= i - 1; x2 = x2 \* 2) {                  System.out.print("   ");                  System.out.print(x2);              }              for (int x = i; x >= 1; x = x / 2) {                  System.out.print("   ");                  System.out.print(x);              }              ix = ix - 4;              System.out.println();          }      }      public static void main(String[] args) {          pyramid();          System.out.println("CREATED BY 21CE013 ABHI BHIMANI");      }  } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.8** | The problem is to write a program that will grade multiple-choice tests. Assume there are eight students and ten questions, and the answers are stored in a twodimensional array. Each row records a student’s answers to the questions, as shown in the following array. Students’ Answers to the Questions: 0 1 2 3 4 5 6 7 8 9  Student 0 A B A C C D E E A D  Student 1 D B A B C A E E A D  Student 2 E D D A C B E E A D  Student 3 C B A E D C E E A D  Student 4 A B D C C D E E A D  Student 5 B B E C C D E E A D  Student 6 B B A C C D E E A D  Student 7 E B E C C D E E A D  The key is stored in a one-dimensional array:  Key to the Questions: 0 1 2 3 4 5 6 7 8 9  Key D B D C C D A E A D  Your program grades the test and displays the result. It compares each student’s answers with the key, counts the number of correct answers, and displays it. |
| **Code** | /\*   \* Created by:- ABHI BHIMANI   \* ID:- 21CE013   \* Aim:- The problem is to write a program that will grade multiple-choice tests. Assume   \* there are eight students and ten questions, and the answers are stored in a two dimensional array. Each row records a student’s answers to the questions, as   \* shown in the following array.   \* Students’ Answers to the Questions:   \* 0 1 2 3 4 5 6 7 8 9   \*                 Student 0 A B A C C D E E A D   \*                 Student 1 D B A B C A E E A D   \*                 Student 2 E D D A C B E E A D   \*               Student 3 C B A E D C E E A D   \*                 Student 4 A B D C C D E E A D   \*                 Student 5 B B E C C D E E A D   \*                 Student 6 B B A C C D E E A D   \*                 Student 7 E B E C C D E E A D   \* The key is stored in a one-dimensional array:   \* Key to the Questions:   \* 0 1 2 3 4 5 6 7 8 9   \* Key D B D C C D A E A D   \*/  import java.util.\*;  public class Ass8StudentAnsMain {      public static void main(String[] args) {          Scanner sc = new Scanner(System.in);          char[][] stu = new char[8][11];          for(int i=0;i<8;i++)          {              System.out.print("Enter the answers of Student "+i+": ");              stu[i][0] = (char)i;              for(int j=1;j<11;j++)              {                  char a = sc.next().charAt(0);                  stu[i][j] = a;              }          }          char key[] = {'D','B','D','C','C','D','A','E','A','D'};          int ans[][] = new int[8][2];          int c=0;          for(int i=0;i<8;i++)          {              ans[i][0] = stu[i][0];              for(int j=1;j<11;j++)              {                  if(stu[i][j]==key[j-1])                  {                      c++;                  }              }              ans[i][1]=c;              c=0;          }          System.out.println();          for(int i=0;i<8;i++)          {              System.out.println("Number of correct answers by Student "+ans[i][0]+" is "+ans[i][1]);          }          System.out.println("CREATED BY 21CE013 ABHI BHIMANI");      }    } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.9** |  |
| **Code** | ////CREATED BY 21CE013 ABHI BHIMANI  import java.io.\*;  import java.util.\*;  public class P1\_9{          static int N = 9;          // Function to check if all elements of the board[][] array store value in the range[1, 9]          static boolean isinRange(int[][] board)          {              for(int i = 0; i < N; i++)        // Traverse board[][] array              {                  for(int j = 0; j < N; j++)                  {                      // Check if board[i][j] lies in the range                      if (board[i][j] <= 0 || board[i][j] > 9)                      {                          return false;                      }                  }              }              return true;          }          // Function to check if the solution of sudoku puzzle is valid or not          static boolean isValidSudoku(int board[][])          {              if (isinRange(board) == false)   // Check if all elements of board[][] stores value in the range[1, 9]              {                  return false;              }              boolean[] unique = new boolean[N + 1];    // Stores unique value from 1 to N              for(int i = 0; i < N; i++)                // Traverse each row of the given array              {                  Arrays.fill(unique, false);       // Initialize unique[] array to false                  for(int j = 0; j < N; j++)            // Traverse each column of current row                  {                      int Z = board[i][j];              // Stores the value of board[i][j]                      if (unique[Z])                    // Check if current row stores duplicate value                      {                          return false;                      }                      unique[Z] = true;                  }              }              for(int i = 0; i < N; i++)               // Traverse each column of the given array              {                  Arrays.fill(unique, false);      // Initialize unique[] array to false                  for(int j = 0; j < N; j++)           // Traverse each row of current column                  {                      int Z = board[j][i];             // Stores the value of board[j][i]                      if (unique[Z])                   // Check if current column stores duplicate value                      {                          return false;                      }                      unique[Z] = true;                  }              }              for(int i = 0; i < N - 2; i += 3)            // Traverse each block of size 3 \* 3 in board[][] array              {                  for(int j = 0; j < N - 2; j += 3)        // j stores first column of each 3 \* 3 block                  {                      Arrays.fill(unique, false);      // Initialize unique[] array to false                      for(int k = 0; k < 3; k++)            // Traverse current block                      {                          for(int l = 0; l < 3; l++)                          {                              int X = i + k;          // Stores row number of current block                              int Y = j + l;          // Stores column number of current block                              int Z = board[X][Y];    // Stores the value of board[X][Y]                              if (unique[Z])          // Check if current block stores duplicate value                              {                                  return false;                              }                              unique[Z] = true;                          }                      }                  }              }              // If all conditions satisfied              return true;          }          public static void main(String[] args)          {              int[][] board = { { 5, 3, 4, 6, 7, 8, 9, 1, 2 },                      { 6, 7, 2, 1, 9, 5, 3, 4, 8 },                      { 1, 9, 8, 3, 4, 2, 5, 6, 7 },                      { 8, 5, 9, 7, 6, 1, 4, 2, 3 },                      { 4, 2, 6, 8, 5, 3, 7, 9, 1 },                      { 7, 1, 3, 9, 2, 4, 8, 5, 6 },                      { 9, 6, 1, 5, 3, 7, 2, 8, 4 },                      { 2, 8, 7, 4, 1, 9, 6, 3, 5 },                      { 3, 4, 5, 2, 8, 6, 1, 7, 9 } };              if (isValidSudoku(board))              {                  System.out.println("Valid");              }              else              {                  System.out.println("Not Valid");              }              System.out.println("CREATED BY 21CE013 ABHI BHIMANI");          }      } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |
| **Practical 1.10** | Implement Caesar Cipher. |
| **Code** | ////CREATED BY 21CE013 ABHI BHIMANI  import java.util.Scanner;  public class P1\_10 {          public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";          public static String encrypt(String plainText, int shiftKey)          {              plainText = plainText.toLowerCase();              String cipherText = "";              for (int i = 0; i < plainText.length(); i++)              {                  int charPosition = ALPHABET.indexOf(plainText.charAt(i));                  int keyVal = (shiftKey + charPosition) % 26;                  char replaceVal = ALPHABET.charAt(keyVal);                  cipherText += replaceVal;              }              return cipherText;          }          public static String decrypt(String cipherText, int shiftKey)          {              cipherText = cipherText.toLowerCase();              String plainText = "";              for (int i = 0; i < cipherText.length(); i++)              {                  int charPosition = ALPHABET.indexOf(cipherText.charAt(i));                  int keyVal = (charPosition - shiftKey) % 26;                  if (keyVal < 0)                  {                      keyVal = ALPHABET.length() + keyVal;                  }                  char replaceVal = ALPHABET.charAt(keyVal);                  plainText += replaceVal;              }              return plainText;          }          public static void main(String[] args)          {              Scanner sc = new Scanner(System.in);              System.out.println("Enter the String for Encryption: ");              String message = new String();              message = sc.next();              System.out.println(encrypt(message, 3));              System.out.println(decrypt(encrypt(message, 3), 3));              System.out.println("CREATED BY 21CE013 ABHI BHIMANI");              sc.close();          }      } |
| **Output** |  |
| **GitHub** | https://github.com/abhii14758/JAVA-PRACTICAL-FILE-1 |