Practical 1

Github repository link : https://github.com/kalp104/21ce084.git

|  |  |
| --- | --- |
| Question 1 | Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument. |
|  | Java is one of the most popular and widely used programming language and platform. A platform is an environment that helps to develop and run programs written in any programming language.  Java is fast, reliable and secure. From desktop to web applications, scientific supercomputers to gaming consoles, cell phones to the Internet, Java is used in every nook and corner.  **JDK:** The JDK is a development environment for building applications, applets, and components using the Java programming language. The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform  **JRE:** The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer’s operating system software and provides the class libraries and other resources that a specific Java program needs to run.  **JVM:** The Java Virtual Machine (JVM) is the runtime engine of the Java Platform, which allows any program written in Java or other language compiled into Java bytecode to run on any computer that has a native JVM.  **Javadoc:** Javadoc (originally cased Javadoc) is a documentation generator created by Sun Microsystems for the Java language (now owned by Oracle Corporation) for generating API documentation in HTML format from Java source code  **Command Line Argument:** Command Line Argument in Java is the information that is passed to the program when it is executed. The information passed is stored in the string array passed to the main() method and it is stored as a string. It is the information that directly follows the program's name on the command line when it is running.  **Introduction to Object oriented Concepts:**   * Object:   Objects are always called as instances of a class. Objects are created from class in java or any other languages. Objects are those that have state and behaviour. Objects are abstract data types (i.e., objects behaviour is defined by a set of values and operations)   * Class:   Classes are like object constructors for creating objects. The collection of objects is said to be a class. Classes are said to be logical quantities. Classes don’t consume any space in the memory. Class is also called a template of an object. Classes have members which can be fields, methods and constructors. A class has both static and instance initializers.   * Inheritance:   **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviours of a parent object. It is an important part of [OOPs](https://www.javatpoint.com/java-oops-concepts) (Object Oriented programming system).  The idea behind inheritance in Java is that you can create new [classes](https://www.javatpoint.com/object-and-class-in-java) that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.   * Polymorphism:   It is a concept by which we can perform a single action in different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So, polymorphism means many forms.  There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.  If you overload a static method in Java, it is the example of compile time polymorphism. Here, we will focus on runtime polymorphism in java.   * Abstraction:    It is a process of hiding the implementation details and showing only functionality to the user.  Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.  Abstraction lets you focus on what the [object](https://www.javatpoint.com/object-and-class-in-java) does instead of how it does it.   * Encapsulation:   It is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines. We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it. |
| Question 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" \*/ // 21ce084 // kalp pandya import java.util.Scanner; public class code1\_oz{  public static void main(String[] args) {  String s;  Scanner sc = new Scanner(System.*in*);  s = sc.next();  char[] ch = s.toCharArray();// converting the string into array of char  if(ch[0] == 'o')  {  System.*out*.print(ch[0]);  }  if(ch[1] == 'z')  {  System.*out*.print(ch[1]);   }  System.*out*.println();  } } |
|  |  |
| Question 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\*/ // 21ce084 // kalp pandya import java.util.Scanner; public class code2\_lastdigit {  public static void main(String[] args)  {  int c=0;  int d=0;  Scanner sc = new Scanner(System.*in*);  int a = sc.nextInt();  int b = sc.nextInt();  System.*out*.println(*lastdigit*(a,b));  }  public static boolean lastdigit(int a,int b)  {  int c,d;  c = a % 10;  d = b % 10;  return c==d;//return value ine either true or false  } } |
|  |  |
| Question 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\*/ // 21ce084 // kalp pandya import java.util.Scanner;  public class code\_of\_array {   public static void main (String[] args)  {  int flag=0;  Scanner sc = new Scanner(System.*in*);  int a[] = {1,1,2,3,1};  for(int i=0;i<(a.length-2);i++)  {  if(a[i]==1 && a[i+1]==2 && a[i+2]==3)  {  flag=1;  }   }  if(flag==1){  System.*out*.println("true");  }else if(flag==0){  System.*out*.println("flase");  }  }  } |
|  |  |
| Question 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\*/ // 21ce084 // kalp pandya import java.util.Scanner; public class count\_Q5 {  public static void main(String[] args) {  String a="xxcaazz";  String b="xxbaaz";  int c = *stringMatch*(a,b);  System.*out*.println(c);  }  public static int stringMatch(String a, String b) {  int len = Math.*min*(a.length(), b.length());  int count = 0;   for (int i = 0; i < len - 1; i++) {  String aSub = a.substring(i, i + 2);  // System.out.println("asub : " + aSub);  String bSub = b.substring(i, i + 2);  if (aSub.equals(bSub))  count++;  }  return count;  } } |
|  |  |
| Question 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"]\*/ // 21ce084 // kalp pandya public class target\_string {  public static void main(String[] args) {  String[] s= {"a","b","c","a"};  String a = "a";  int cout =0;  for(int i=0;i<s.length;i++)  {  if(s[i].equals(a))  {  cout++;  }  }  int j=0;  String[] sc = new String[s.length - cout];  for(int i=0;i<s.length;i++) {  if (!s[i].equals(a)) {  sc[j] = s[i];  j++;  }  }  System.*out*.println(sc);   } } |
|  |  |
| Question 7: | // 1 // 1 2 1 // 1 2 4 2 1 // 1 2 4 8 4 2 1 up to 8 rows // 21ce084 // kalp pandya import java.util.Scanner; public class pattern {  public static void main(String[] args) {   for(int i=1;i<=8;i++) // loop of number of raws  {  for(int j=8;j>i;j--) // first empty triangle  {  System.*out*.print(" ");   }  int k=1;  for(int j=1;j<=i;j++)  {  System.*out*.print(k+" ");  k = k \* 2;  }  k=k/4;  for (int j=1;j<i;j++)  {  System.*out*.print(k+" ");  k = k / 2;  }  System.*out*.println();  }  } } |
|  |  |
| Question 8: | // 21ce084 //kalp pandya /\*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.\*/  import java.util.Scanner;  public class prac8 {   public static void main (String[] args)  {  Scanner sc = new Scanner (System.*in*);  String[] ans= { "D","B","D","C","C","D","A","E","A","D"};// answer key  String[][] str = new String[8][10]; // runtime initialization   System.*out*.println("Fill in the eight student's answers to 10 questions" );  for (int i =0;i<str.length;i++) // here string length is 8  {   int count =0;  for (int j=0;j<str[i].length;j++) // here string length is 10  {  str[i][j] = sc.next();  if(str[i][j].equalsIgnoreCase(ans[j])) // Ignor case of string  count++; // counting right answer if it is true  }  System.*out*.println("Student's " + (i+1) + " correct answers are " + count);  }  } } |
|  |  |
| Question 9: | // 21ce084 // kalp pandya /\* The problem is to check whether a given Sudoku solution is correct.\*/ public class prac9 {   public static void main(String args[]) {   int a[][] = { { 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 } };  // ansswer of phsudocode   System.*out*.println(*check*(a));  }   public static boolean check(int arr[][]) {  int i, j;  int count[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };  int count1[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };  boolean b = true;  for (i = 0; i < 9; i++) {  for (j = 0; j < 9; j++) {   if (count[arr[j][i]] > i) {  b = false;  return b;  }  if (count1[arr[i][j]] > i) {  b = false;  return b;  }  count1[arr[i][j]]++;  count[arr[j][i]]++;  }   }  return b;  }  } |
|  |  |
| Question 10: | Caesar Cipher Program in Java  (taken from geeksforgeeks)It is one of the simplest and most used encryption techniques. In this technique, each letter of the given text is replaced by a letter of some fixed number of positions down the alphabet.  For example, with a shift of 1, X would be replaced by Y, Y would become Z, and so on. Julius Caesar was the first one who used it for communicating with his officials. Based on his name, this technique was named as **Caesar Cipher** technique.  An integer value is required to cipher a given text. The integer value is known as shift, which indicates the number of positions each letter of the text has been moved down.  Caesar Cipher Program in Java  We use the following steps to implement the program for **the Caesar Cipher** technique:   1. Take an input string from the user to encrypt it using **the Caesar Cipher** technique. 2. Take an input integer from the user for shifting characters. The input integer should be between 0-25. 3. Traverse input string one character at a time. 4. Depending on the encryption and decryption, we transform each character as per the rule. 5. Returns the newly generated string.   import java.util.Scanner;  public class prac10 {  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));  sc.close();  } }    and the output is : |