**PART-I**

**Data Types, Variable, Array, Operator, Control Statement, String**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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   |  |  | | --- | --- | | **C++** | **Java** | | Extends C with object-oriented programming and [generic programming](https://en.wikipedia.org/wiki/Generic_programming). C code can most properly be used. | Strongly influenced by C++/C syntax. | | Compatible with [C](https://en.wikipedia.org/wiki/C_(programming_language)) source code, except for a few [corner cases](https://en.wikipedia.org/wiki/Corner_case). | Provides the Java Native Interface and recently [Java Native Access](https://en.wikipedia.org/wiki/Java_Native_Access) as a way to directly call C/C++ code. | | [Write once, compile anywhere](https://en.wikipedia.org/wiki/Write_once,_compile_anywhere) (WOCA). | [Write once, run anywhere](https://en.wikipedia.org/wiki/Write_once,_run_anywhere)/everywhere (WORA/WORE). | | Allows [procedural programming](https://en.wikipedia.org/wiki/Procedural_programming), [functional programming](https://en.wikipedia.org/wiki/Functional_programming), [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), [generic programming](https://en.wikipedia.org/wiki/Generic_programming), and [template metaprogramming](https://en.wikipedia.org/wiki/Template_metaprogramming). Favors a mix of paradigms. | Allows [procedural programming](https://en.wikipedia.org/wiki/Procedural_programming), [functional programming](https://en.wikipedia.org/wiki/Functional_programming) (since Java 8) and [generic programming](https://en.wikipedia.org/wiki/Generic_programming) (since Java 5), but strongly encourages the [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [programming paradigm](https://en.wikipedia.org/wiki/Programming_paradigm). Includes support for creating [scripting languages](https://en.wikipedia.org/wiki/Scripting_language). | | Runs as native executable machine code for the target [instruction set](https://en.wikipedia.org/wiki/Instruction_set)(s). | Runs on a [virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine). | | Provides object types and type names. Allows reflection via [run-time type information](https://en.wikipedia.org/wiki/Run-time_type_information) (RTTI). | Is [reflective](https://en.wikipedia.org/wiki/Reflection_(computer_programming)), allowing metaprogramming and dynamic code generation at runtime. | | Has multiple binary compatibility standards (commonly Microsoft (for MSVC compiler) and Itanium/GNU (for almost all other compilers)). | Has one binary compatibility standard, [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) for OS and compiler. | | Optional automated [bounds checking](https://en.wikipedia.org/wiki/Bounds_checking) (e.g., the at() method in vector and string containers). | All operations are required to be bound-checked by all compliant distributions of Java. [HotSpot](https://en.wikipedia.org/wiki/HotSpot) can remove bounds checking. | | Native [unsigned arithmetic](https://en.wikipedia.org/wiki/Unsigned_(arithmetic)) support. | Native unsigned arithmetic unsupported. Java 8 changes some of this, but aspects are unclear.[[1]](https://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B#cite_note-1) | | Standardized minimum limits for all numerical types, but the actual sizes are implementation-defined. Standardized types are available via the standard library <cstdint>. | Standardized limits and sizes of all primitive types on all platforms. | | Pointers, references, and pass-by-value are supported for all types (primitive or user-defined). | All types (primitive types and reference types) are always passed by value.[[2]](https://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B#cite_note-2) | | [Memory management](https://en.wikipedia.org/wiki/Memory_management) can be done [manually](https://en.wikipedia.org/wiki/Manual_memory_management) via new / delete, automatically by scope, or by smart pointers. Supports deterministic destruction of objects. [Garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) ABI standardized in C++11, though compilers are not required to implement garbage collection. | Automatic [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). Supports a non-deterministic finalize() method use of which is not recommended.[[3]](https://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B#cite_note-3) | | [Resource management](https://en.wikipedia.org/wiki/Resource_management_(computing)) can be done manually or by automatic lifetime-based resource management ([RAII](https://en.wikipedia.org/wiki/RAII)). | Resource management must generally be done manually, or automatically via finalizers, though this is generally discouraged. Has try-with-resources for automatic scope-based resource management (version 7 onwards).  It can also be done using the internal API sun.misc.Unsafe but that usage is highly discouraged and will be replaced by a public API in an upcoming Java version. | | Supports classes, structs ([passive data structure](https://en.wikipedia.org/wiki/Passive_data_structure) (PDS) types), and unions, and can allocate them on the [heap](https://en.wikipedia.org/wiki/Dynamic_memory_allocation) or the [stack](https://en.wikipedia.org/wiki/Stack-based_memory_allocation). | Classes are allocated on the [heap](https://en.wikipedia.org/wiki/Dynamic_memory_allocation). [Java SE 6](https://en.wikipedia.org/wiki/Java_version_history#Java_SE_6_Update_14) optimizes with [escape analysis](https://en.wikipedia.org/wiki/Escape_analysis) to allocate some objects on the [stack](https://en.wikipedia.org/wiki/Stack-based_memory_allocation). | | Allows explicitly overriding types, and some implicit narrowing conversions (for compatibility with C). | Rigid [type safety](https://en.wikipedia.org/wiki/Type_safety) except for widening conversions. | | The [C++ Standard Library](https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library) was designed to have a limited scope and functions, but includes language support, diagnostics, general utilities, strings, locales, containers, algorithms, [iterators](https://en.wikipedia.org/wiki/Iterator#C.2B.2B), numerics, input/output, random number generators, regular expression parsing, threading facilities, type traits (for static type introspection) and Standard C Library. The [Boost library](https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries)) offers more functions including network I/O.  A rich amount of third-party libraries exist for GUI and other functions like: [Adaptive Communication Environment](https://en.wikipedia.org/wiki/Adaptive_Communication_Environment) (ACE), [Crypto++](https://en.wikipedia.org/wiki/Crypto%2B%2B), various [XMPP](https://en.wikipedia.org/wiki/XMPP) [Instant Messaging](https://en.wikipedia.org/wiki/Instant_Messaging) (IM) libraries,[[4]](https://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B#cite_note-XMPP_Software_.C2.BB_Libraries-4) [OpenLDAP](https://en.wikipedia.org/wiki/OpenLDAP), [Qt](https://en.wikipedia.org/wiki/Qt_(software)), [gtkmm](https://en.wikipedia.org/wiki/Gtkmm). | The standard library has grown with each release. By version 1.6, the library included support for locales, logging, containers and iterators, algorithms, GUI programming (but not using the system GUI), graphics, multi-threading, networking, platform security, introspection, dynamic class loading, blocking and non-blocking I/O. It provided interfaces or support classes for [XML](https://en.wikipedia.org/wiki/XML), [XSLT](https://en.wikipedia.org/wiki/XSLT), [MIDI](https://en.wikipedia.org/wiki/MIDI), database connectivity, naming services (e.g. [LDAP](https://en.wikipedia.org/wiki/LDAP)), cryptography, security services (e.g. [Kerberos](https://en.wikipedia.org/wiki/Kerberos_(protocol))), print services, and web services. SWT offered an abstraction for platform-specific GUIs, but was superseded by [JavaFX](https://en.wikipedia.org/wiki/JavaFX) in the latest releases ; allowing for graphics acceleration and CSS-themable UIs. It although doesn't support any kind of "native platform look" support. | | [Operator overloading](https://en.wikipedia.org/wiki/Operator_overloading) for most operators. Preserving meaning (semantics) is highly recommended. | Operators are not overridable. The language overrides + and += for the String class. | | Single and [Multiple inheritance](https://en.wikipedia.org/wiki/Multiple_inheritance) of classes, including virtual inheritance. | Single inheritance of classes. Supports multiple inheritance via the [Interfaces](https://en.wikipedia.org/wiki/Interface_(Java)) construct, which is equivalent to a C++ class composed of abstract methods. | | Compile-time templates. Allows for [Turing complete](https://en.wikipedia.org/wiki/Turing_complete) meta-programming. | [Generics](https://en.wikipedia.org/wiki/Generics_in_Java) are used to achieve basic type-parametrization, but they do not translate from source code to byte code due to the use of [type erasure](https://en.wikipedia.org/wiki/Type_erasure) by the compiler. | | Function pointers, function objects, lambdas (in [C++11](https://en.wikipedia.org/wiki/C%2B%2B11)), and interfaces. | Functions references, function objects and lambdas were added in [Java 8](https://en.wikipedia.org/wiki/Java_8). Classes (and interfaces, which are classes) can be passed as references as well through SomeClass.class | | No standard inline documentation mechanism. Third-party software (e.g. [Doxygen](https://en.wikipedia.org/wiki/Doxygen)) exists. | Extensive [Javadoc](https://en.wikipedia.org/wiki/Javadoc) documentation standard on all system classes and methods. | | const keyword for defining immutable variables and member functions that do not change the object. Const-ness is propagated as a means to enforce, at compile-time, correctness of the code with respect to mutability of objects (see [const-correctness](https://en.wikipedia.org/wiki/Const-correctness)). | final provides a version of const, equivalent to type\* const pointers for objects and const for primitive types. Immutability of object members achieved via read-only interfaces and object encapsulation. | | Supports the [goto](https://en.wikipedia.org/wiki/Goto) statement. | Supports labels with loops and statement blocks. goto is a reserved keyword but is marked as "unused" in the [Java specification.](https://docs.oracle.com/javase/specs/) | | Source code can be written to be [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) (can be compiled for [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [BSD](https://en.wikipedia.org/wiki/BSD), [Linux](https://en.wikipedia.org/wiki/Linux), [macOS](https://en.wikipedia.org/wiki/MacOS), [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), etc., without modification) and written to use platform-specific features. Typically compiled into native machine code, must be recompiled for each target platform. | Compiled into byte code for the [JVM](https://en.wikipedia.org/wiki/JVM). Byte code is dependent on the Java platform, but is typically independent of [operating system](https://en.wikipedia.org/wiki/Operating_system) specific features. |   INFORMATION ABOUT JDK:  The **Java Development Kit** (**JDK**) is an implementation of either one of the [Java Platform, Standard Edition](https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition), [Java Platform, Enterprise Edition](https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition), or [Java Platform, Micro Edition](https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition) platforms[[1]](https://en.wikipedia.org/wiki/Java_Development_Kit#cite_note-1) released by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation) in the form of a binary product aimed at [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) developers on [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), [Linux](https://en.wikipedia.org/wiki/Linux), [macOS](https://en.wikipedia.org/wiki/MacOS) or [Windows](https://en.wikipedia.org/wiki/Windows). The JDK includes a private JVM and a few other resources to finish the development of a Java Application.[[2]](https://en.wikipedia.org/wiki/Java_Development_Kit#cite_note-2) Since the introduction of the [Java](https://en.wikipedia.org/wiki/Java_(software_platform)) platform, it has been by far the most widely used Software Development Kit ([SDK](https://en.wikipedia.org/wiki/Software_development_kit)).[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] On 17 November 2006, Sun announced that they would release it under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL), thus making it [free software](https://en.wikipedia.org/wiki/Free_software). This happened in large part on 8 May 2007, when Sun contributed the source code to the [OpenJDK](https://en.wikipedia.org/wiki/OpenJDK).[[3]](https://en.wikipedia.org/wiki/Java_Development_Kit#cite_note-3)  Information about jre:  **Java Runtime Environment** (**JRE**) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the [Java Class Library](https://en.wikipedia.org/wiki/Java_Class_Library). The [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation), which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called [HotSpot](https://en.wikipedia.org/wiki/HotSpot).  INFORMATION ABOUT JVM:  A **Java virtual machine** (**JVM**) is [an abstract computing machine](https://en.wikipedia.org/wiki/Virtual_machine#Process_virtual) that enables a computer to run a [Java](https://en.wikipedia.org/wiki/Java_(software_platform)) program. There are three notions of the JVM: specification, implementation, and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into [Java bytecode](https://en.wikipedia.org/wiki/Java_bytecode).  INFORMATION ABOUT JAVADOC:  **Javadoc** (originally cased **JavaDoc**)[[1]](https://en.wikipedia.org/wiki/Javadoc#cite_note-1) is a [documentation generator](https://en.wikipedia.org/wiki/Documentation_generator) created by [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) for the Java language (now owned by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation)) for generating [API](https://en.wikipedia.org/wiki/Application_programming_interface) documentation in [HTML](https://en.wikipedia.org/wiki/HTML) format from [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) source code. The HTML format is used for adding the convenience of being able to [hyperlink](https://en.wikipedia.org/wiki/Hyperlink) related documents together.[[2]](https://en.wikipedia.org/wiki/Javadoc#cite_note-2)  The "doc comments" format[[3]](https://en.wikipedia.org/wiki/Javadoc" \l "cite_note-3) used by Javadoc is the de facto industry standard for documenting Java classes. Some [IDEs](https://en.wikipedia.org/wiki/Integrated_Development_Environment),[[4]](https://en.wikipedia.org/wiki/Javadoc" \l "cite_note-4) like [Netbeans](https://en.wikipedia.org/wiki/Netbeans) and [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)), automatically generate Javadoc HTML. Many file editors assist the user in producing Javadoc source and use the Javadoc info as internal references for the programmer.  Javadoc also provides an API for creating [doclets](https://en.wikipedia.org/wiki/Doclets) and taglets, which allows users to analyze the structure of a Java application. This is how [JDiff](https://en.wikipedia.org/wiki/JDiff) can generate reports of what changed between two versions of an API.  INFORMATION ABOUT COMMAND LINE ARGUMENTS:  We use javac classname.java (for compiling)  We use java classname arg1 arg2 arg3…(for running class file) |
| **2** | 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  package jainil;  import java.util.Scanner;  public class Jainil {    public static boolean lastDigit(int a,int b)  {  if((b%10)==a)  {return true;}  else  {return false;}  }  public static void main(String[] args) {    // TODO code application logic here  Scanner sc = new Scanner(System.in);  int a,b;  System.out.println("enter first no");  a=sc.nextInt();  System.out.println("enter second number");  b=sc.nextInt();  if(lastDigit(a, b)==true)  System.out.println("last digit matched");  else  System.out.println("last digit doesnot match");  }  }  **OUTPUT**  1 |
| **3** | Implement following task using Array. Return an array that contains exactly the same numbers as the given array, but rearranged so that every 3 is immediately followed by a 4. Do not move the 3's, but every other number may move. The array contains the same number of 3's and 4's, every 3 has a number after it that is not a 3 or 4, and a 3 appears in the array before any 4.public int[] fix34(int[] nums)  Program  package pra3;  import java.util.Scanner;  public class Pra3 {  public static int[] fic34(int a[])  {  if(a.length==2)  {  if(a[0]==4&&a[1]==3)  {  a[0]=3;a[1]=4;  }  if(a[0]==3&&a[1]==4)  {  a[0]=3;a[1]=4;  }  }  else{  int i=0,j=0,k=0;  for(i=0;i<a.length-1;i++)  {  if(a[i]==3)  {  if(i==0||a[i+1]!=4||i==a.length)  {  for(k=0;j<a.length;k++)  {  if(a[k]==4)  {  if(k==0||a[k-1]!=3)  {  int temp;  temp=a[i+1];  a[i+1]=a[k];  a[k]=temp;  break;  }  }  }  }  }  }}  return a;    }  public static void main(String[] args) {  // TODO Auto-generated method stub  int size=0;  Scanner sc=new Scanner(System.in);  System.out.println("Enter number of terms");  size=sc.nextInt();  System.out.println("Enter"+size+" numbers");  int a[]=new int[size];  int i=0;  for(i=0;i<size;i++)  {  a[i]=sc.nextInt();  }  int b[]=new int[size];  b=Pra3.fic34(a);  System.out.println();  System.out.println();  for(i=0;i<size;i++)  {  System.out.print(b[i]+" ");  }  }  }  **OUTPUT** |
| **4** | 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  Program:  package pra4;  import java.util.Scanner;  public class Pra4 {  public static void main(String[] args) {  // TODO code application logic here  int n;  int i;  int j;  int k;  Scanner sc=new Scanner(System.in);  System.out.println("enter no of rows");  n=sc.nextInt();  for(i=0;i<n;i++)  {  for(j=n-i;j>0;j--)  System.out.print(" ");  for(j=0;j<i;j++)  {  System.out.printf("%7d",(int)Math.pow(2,j));  }  for(k=j;k>=0;k--)  {  System.out.printf("%7d",(int)Math.pow(2,k));  }  System.out.print("\n");  }  }  }  **OUTPUT** |
| **5** | 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  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.  Program:  package pra5;  import java.util.Scanner;  public class Pra5 {  public static void main(String[] args)  {  Scanner sc= new Scanner(System.in);  // TODO Auto-generated method stub  String a[]=new String[8];  for(int i=0;i<8;i++)  {  a[i]=sc.next();  }  String answerkey="DBDCCDAEAD";  int marks[]=new int[8];  for(int i=0;i<8;i++)  {  marks[i]=0;  }  for(int i=0;i<8;i++)  {  for(int j=0;j<10;j++)  {  if(answerkey.charAt(j)==a[i].charAt(j))  {  marks[i]=marks[i]+1;  }  }  }  for(int i=0;i<8;i++)  {  System.out.println("Marks of student "+i+"="+marks[i]);  }  }  }  **OUTPUT** |
| **6** | 6) The problem is to check whether a given Sudoku solution is correct.  package sudoku;  import java.util.Scanner;  public class Sudoku {  public static void printGrid(int grid[][])  {  for (int row = 0; row < 9; row++)  {  for (int col = 0; col < 9; col++)  System.out.print(grid[row][col]+" ");  System.out.println();  }  }  public static void main(String[] args)  {  int flag=0;  Scanner sc=new Scanner(System.in);  System.out.println("your question here 0 means unassigned");  int grid[][]={{5,3,0,0,7,0,0,0,0},{6,0,0,1,9,5,0,0,0},{0,9,8,0,0,0,0,6,0},{8,0,0,0,6,0,0,0,3},{4,0,0,8,0,3,0,0,1}  ,{7,0,0,0,2,0,0,0,6},{0,6,0,0,0,0,0,0,0},{0,0,0,4,1,9,0,0,5},{0,0,0,0,8,0,0,7,9}};  int a[][]=new int[9][9];  printGrid(grid);  start:  System.out.println("Enter your answer:");  for(int i=0;i<9;i++)  {  for(int j=0;j<9;j++)  {  System.out.println("enter "+i+" "+j+" "+"digit:");  a[i][j]=sc.nextInt();  }  }  int ans[][]={{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}};  aa:  for (int i = 0; i < 9; i++) {  for (int j = 0; j < 9; j++) {  if(a[i][j]!=ans[i][j])  {  flag=1;  break aa;    }  }    }  if(flag==0)  {  System.out.println("Congrats your answer is correct");  }  else  {  System.out.println("sorry do you want to see answer");    {  System.out.println("your wrong input");  printGrid(a);  System.out.println("your correct answer is");  printGrid(ans);  } } }  }  **OUTPUT** |
| **7** | 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  Program:  package pra7;  import java.util.Scanner;  /\*\*  \*  \* @author jainil  \*/  public class Pra7 {  static int stringMatch(String a,String b)  {  int count=0;  for(int i=0;i<a.length()-1;i++)  {  if(((i+1)<a.length())&&((i+1)<b.length()))  {  if ((a.charAt(i)==b.charAt(i))&&(a.charAt(i+1)==b.charAt(i+1)))  {  count++;  }  }  }  return count;  }  public static void main(String[] args) {    Scanner sc=new Scanner(System.in);  int count=0;  String a;  String b;  System.out.println("enter first string");  a=sc.next();  System.out.println("enter second string");  b=sc.next();  count=stringMatch(a, b);  System.out.println(count); }}  **OUTPUT**  9) Implement Caesar Cipher.  Program:  package pra9;  import java.util.Scanner;  /\*\*  \*  \* @author jainil  \*/  public class Pra9 {  public static void main(String[] args) {  // TODO code application logic here  String s1;  String s2=new String();  int key;  System.out.println("Enter data");  Scanner sc=new Scanner(System.in);  s1=sc.nextLine();  System.out.println("Enter key");  key=sc.nextInt();  char a[]=new char[s1.length()];  a=s1.toCharArray();  int b[]=new int[s1.length()];  for(int i=0;i<s1.length();i++)  {  b[i]=(int)a[i];  }  for(int i=0;i<s1.length();i++)  {  b[i]=(b[i]+key);  }  for(int i=0;i<s1.length();i++)  {  System.out.print((char)b[i]);  }  System.out.println("\nNow reversing");  for(int i=0;i<s1.length();i++)  {  s2=s2+(char)(b[i]-key);  }  System.out.println(s2);  }    }  **OUTPUT** |
| **8** | 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"]  Program:  package pra8;  public class Pra8 {    public static String[] wordsWithout(String[] words, String target) {  int found = 0;    for (int i = 0; i < words.length; i++) {  if (words[i].equals(target))  found++;  }  found = words.length - found;  int place = 0;    String[] str = new String[found];  for (int j = 0; j < words.length; j++) {  if (!words[j].equals(target)) {  str[place] = words[j];  place++;  }  }  return str;  }  public static void main(String[] args)  {  String s[]={"a","e","i","o","u"};  s=wordsWithout(s,"a");  for(int i=0;i<s.length;i++)  System.out.println(s[i]);    }  }  **OUTPUT** |
| **9** | Implement Caesar Cipher.  Program:  package pra9;  import java.util.Scanner;  public class Pra9 {  public static void main(String[] args) {  // TODO code application logic here  String s1;  String s2=new String();  int key;  System.out.println("Enter data");  Scanner sc=new Scanner(System.in);  s1=sc.nextLine();  System.out.println("Enter key");  key=sc.nextInt();  char a[]=new char[s1.length()];  a=s1.toCharArray();  int b[]=new int[s1.length()];  for(int i=0;i<s1.length();i++)  {  b[i]=(int)a[i];  }  for(int i=0;i<s1.length();i++)  {  b[i]=(b[i]+key);  }  for(int i=0;i<s1.length();i++)  {  System.out.print((char)b[i]);  }  System.out.println("\nNow reversing");  for(int i=0;i<s1.length();i++)  {  s2=s2+(char)(b[i]-key);  }  System.out.println(s2);  }    }  **OUTPUT**  **PART-II**  **Object Oriented Programming: Classes, Methods, Inheritance** |
| **1** | Design a class named Circle containing following attributes and behavior.  One double data field named radius. The default value is 1.  A no-argument constructor that creates a default circle.  A Single argument constructor that creates a Circle with the specified radius.  A method named getArea() that returns area of the Circle.  A method named getPerimeter() that returns perimeter of it  Program:  package circle;  import java.util.Scanner;  /\*\*  \*  \* @author jainil  \*/  public class Circle {  double radius;  Circle()  {  radius=1;  }  Circle(double d)  {  radius=d;  }  double getarea()  {  return 3.14\*radius\*radius;  }  double getperimeter()  {  return 2\*3.14\*radius;  }      public static void main(String[] args) {  // TODO code application logic here  int d;  Scanner sc=new Scanner(System.in);  System.out.println("enter radius");  d=sc.nextInt();  Circle c=new Circle(d);  System.out.println("Area is "+c.getarea());  System.out.println("Perimeter is "+c.getperimeter());    }  }  **OUTPUT** |
| **2** | ) Design a class named Account that contains:  A private int data field named id for the account (default 0).  A private double data field named balance for the account (default 500₹).  A private double data field named annualInterestRate that stores the current interest rate (default 7%). Assume all accounts have the same interest rate.  A private Date data field named dateCreated that stores the date when the account was created.  A no-arg constructor that creates a default account.  A constructor that creates an account with the specified id and initial balance.  The accessor and mutator methods for id, balance, and annualInterestRate.  The accessor method for dateCreated.  A method named getMonthlyInterestRate() that returns the monthly interest rate.  A method named getMonthlyInterest() that returns the monthly interest.  A method named withdraw that withdraws a specified amount from the account.  A method named deposit that deposits a specified amount to the account.  Program:  package account;  import static java.lang.System.exit;  import java.util.Date;  import java.util.Scanner;  /\*\*  \*  \* @author jainil  \*/  public class Account {  /\*\*  \* @param args the command line arguments  \*/  static int count=0;  private int id=0;  private double balance=0;  private double annualInterestRate=7;  private Date dateCreated=new Date();    Account()  {  id=count;  balance=0;  annualInterestRate=7;  count++;      }  Account (int idd,double balancee)  {  id=idd;  balance=balancee;    }  public static int getCount() {  return count;  }  public static void setCount(int count) {  Account.count = count;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public double getBalance() {  return balance;  }  public void setBalance(double balance) {  this.balance = balance;  }  public double getAnnualInterestRate() {  return annualInterestRate;  }  public void setAnnualInterestRate(double annualInterestRate) {  this.annualInterestRate = annualInterestRate;  }  public Date getDateCreated() {  return dateCreated;  }  public void setDateCreated(Date dateCreated) {  this.dateCreated = dateCreated;  }  Scanner sc=new Scanner(System.in);  void getdata()  {  System.out.println("Enter id");  id=sc.nextInt();  System.out.println("enter balance");  balance=sc.nextDouble();  System.out.println("enter rate");  annualInterestRate=sc.nextDouble();  }  void printdata()  {  System.out.println("id:"+id+" \n"+"balance:"+balance+" \n"+"annual interest rate"+annualInterestRate);    System.out.println("date:"+dateCreated.getDate()+" "+(dateCreated.getMonth()+1)+" "+(dateCreated.getYear()+1900));  }  void printdate()  {  System.out.println("date"+dateCreated.getDate()+" "+(dateCreated.getMonth()+1)+" "+(dateCreated.getYear()+1900));  }  void withdraw(double a)  {  balance=balance-a;  }  void deposit(double a)  {  balance=balance+a;  }  double getMonthlyInterestRate()  {  System.out.println("Monthly interest rate is");  return annualInterestRate/12;  }  double getMonthlyInterest()  {  double c=balance\*annualInterestRate\*(1.0/12.0)\*0.01;  return c;  }  public static void main(String[] args) {  // TODO code application logic here    Account a[]=new Account[10];  for (int i = 0; i < a.length; i++) {  a[i]=new Account();    }  aa:  while (true) {    System.out.println("total account in bank are 10");  System.out.println("1.for inserting balance");  System.out.println("2.for withrawing balance");  System.out.println("3.for balance sheet");  System.out.println("4.for new explicit registration");  System.out.println("5.fot rate change");  System.out.println("6.for monthly rate");  System.out.println("7.to end");  int choice;  Scanner sc=new Scanner(System.in);  choice=sc.nextInt();  switch (choice) {  case 1:  {int x;  System.out.println("enter id");  x=sc.nextInt();  System.out.println("enter inserting balance");  float y;  y=sc.nextInt();  a[x].balance=a[x].balance+y;  break;}  case 2:  {int x;  System.out.println("enter id");  x=sc.nextInt();  System.out.println("enter withdrawing balance");  float y;  y=sc.nextInt();  a[x].balance=a[x].balance-y;  break;}  case 3:  {  int x;  System.out.println("enter id");  x=sc.nextInt();  a[x].printdata();  break;  }  case 4:  {  Account acc = new Account();  acc.getdata();  acc.printdata();break;  }  case 5:  {  int x;  System.out.println("enter id");  x=sc.nextInt();  System.out.println("enter rate");  float y;  y=sc.nextInt();  a[x].annualInterestRate=y;break;  }  case 6:  {  int x;  System.out.println("enter id");  x=sc.nextInt();  System.out.println(a[x].getMonthlyInterest());break;  }  case 7:  {  break aa;  }  default:  throw new AssertionError();  }      }    }  }  **OUTPUT** |
| **4** | (Subclasses of Account) In Programming Exercise 2, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn. Draw the UML diagram for the classes and then implement them. Write a test program that creates objects of Account, SavingsAccount, and CheckingAccount and invokes their toString() methods  Program:  package demo;  import static java.lang.System.exit;  import java.util.Date;  import java.util.Scanner;  class Account {  static int count=0;  private int id=0;  private double balance=0;  private double annualInterestRate=7;  private Date dateCreated=new Date();  private int limit=0;  public void setLimit(int limit) {  this.limit = limit;  }  public int getLimit() {  return limit;  }  Account()  {  id=count;  balance=0;  annualInterestRate=7;  count++;      }  Account (int idd,double balancee)  {  id=idd;  balance=balancee;    }  public static int getCount() {  return count;  }  public static void setCount(int count) {  Account.count = count;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public double getBalance() {  return balance;  }  public void setBalance(double balance) {  this.balance = balance;  }  public double getAnnualInterestRate() {  return annualInterestRate;  }  public void setAnnualInterestRate(double annualInterestRate) {  this.annualInterestRate = annualInterestRate;  }  public Date getDateCreated() {  return dateCreated;  }  public void setDateCreated(Date dateCreated) {  this.dateCreated = dateCreated;  }  Scanner sc=new Scanner(System.in);  void getdata()  {  System.out.println("Enter id");  id=sc.nextInt();  System.out.println("enter balance");  balance=sc.nextDouble();  System.out.println("enter rate");  annualInterestRate=sc.nextDouble();  }  void printdata()  {  System.out.println("id:"+id+" \n"+"balance:"+balance+" \n"+"annual interest rate"+annualInterestRate);    System.out.println("date:"+dateCreated.getDate()+" "+(dateCreated.getMonth()+1)+" "+(dateCreated.getYear()+1900));  }  void printdate()  {  System.out.println("date"+dateCreated.getDate()+" "+(dateCreated.getMonth()+1)+" "+(dateCreated.getYear()+1900));  }  void withdraw(double a)  {  balance=balance-a;  if(balance<limit)  {  System.out.println("Limit exceeded sorry");  balance=balance+a;  }  }  void deposit(double a)  {  balance=balance+a;  }  double getMonthlyInterestRate()  {  System.out.println("Monthly interest rate is");  return annualInterestRate/12;  }  double getMonthlyInterest()  {  double c=balance\*annualInterestRate\*(1.0/12.0)\*0.01;  return c;  }        }  class SavingAccount extends Account  {    }  class CheckingAccount extends Account  {    }  public class Demo  {  public static void main(String[] args) {    // TODO code application logic here  System.out.println("Saving account");  SavingAccount a=new SavingAccount();  a.getdata();    System.out.println("enter withdawing amount");  double q;  Scanner sc = new Scanner(System.in);  q=sc.nextInt();  a.withdraw(q);  a.printdata();  System.out.println("monthly rate"+a.getMonthlyInterest());  System.out.println("checking account");  CheckingAccount b=new CheckingAccount();    b.getdata();  System.out.println("enter withdawing amount");  double w;  b.setLimit(-5000);  w=sc.nextInt();  b.withdraw(w);  b.printdata();  System.out.println("monthly rate"+b.getMonthlyInterest()+" "+b.getLimit());  }  }  **OUTPUT** |
| **5** | Develop a Program that illustrate method overloading concept  Program:  package practical5;  import java.util.Scanner;  public class Practical5 {  int radius;  double area()  {  return (3.14\*radius\*radius);  }  double area(int r)  {  return (3.14\*r\*r);  }  double area(double r)  {  return (3.14\*r\*r);  }  void area(double ...a)  {  for (double d : a) {  System.out.println("Area is "+(3.14\*d\*d));  }  }  void area(int ...a)  {  for (int d : a) {  System.out.println("Area is "+(3.14\*d\*d));  }  }  public static void main(String[] args) {  // TODO code application logic here  Practical5 p = new Practical5();  int a;  System.out.println("Enter radius");  Scanner scanner = new Scanner(System.in);  a=scanner.nextInt();  p.radius=a;  System.out.println("Area 1 is"+p.area());  System.out.println("Area by one int parameter");  System.out.println("Area 2 is"+p.area(25));  System.out.println("Area with double parameter");  System.out.println("Area 3 is"+p.area(26.3));  System.out.println("Array passing by many int values");  p.area(25.0,30.5,50.3,60.2);  System.out.println("Area by passing many double");  p.area(10,20,30,40,50,60);  }    }  **OUTPUT** |
| **3** | Use the Account class created as above to simulate an ATM machine. Create 10 accounts with id AC001…..AC010 with initial balance 300₹. The system prompts the users to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, display menu with multiple choices. 1. Balance inquiry 2. Withdraw money [Maintain minimum balance 300₹] 3. Deposit money 4. Money Transfer 5. Create Account 6. Deactivate Account 7. Exit Hint: Use ArrayList, which is can shrink and expand with compared to Array.  Program:  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package atm;  import static java.lang.System.exit;  import java.util.ArrayList;  import java.util.Scanner;  /\*\*  \*  \* @author jainil  \*/  class Automatictellermachine  {  float balance=0;  public void balanceinquiry()  {  System.out.println(balance);  }  public void withdraw(float a)  {  balance=balance-a;  if(balance<300)  {  System.out.println("you cannot withdraw rs "+a +" maintain 300 rs");  balance=balance+a;  }  else  {  System.out.println(a+ " withdrawn ");  }  }  public void deposit(float a)  {  balance=balance+a;  }  public float transfer(float a)  {  if((balance-300)>=(a))  {  balance=balance-a;  return a;  }  else  {  System.out.println("not enough balance maintain 300 rs please");  return 0;  }  }    public Automatictellermachine() {  }  public Automatictellermachine(float balance) {  this.balance = balance;  }  public float getBalance() {  return balance;  }  public void setBalance(float balance) {  this.balance = balance;  }      }  public class Atm {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  int i;  int q;  ArrayList<Automatictellermachine> a = new ArrayList<Automatictellermachine> (100);  while(true){  System.out.println("1. Balance inquiry ");  System.out.println("2. Withdraw money [Maintain minimum balance 300₹] ");  System.out.println("3. Deposit money ");  System.out.println("4. Money Transfer ");  System.out.println("5. Create Account ");  System.out.println("6. Deactivate Account ");  System.out.println("7. Exit ");  System.out.println("enter valid choice");  Scanner sc = new Scanner(System.in);  i=sc.nextInt();  switch (i) {  case 1:  System.out.println("enter id for balance inquiry");  q=sc.nextInt();  if(a.size()>q)  a.get(q).balanceinquiry();  else  {  System.out.println("USER NOT PRESENT");  }  break;  case 2:  System.out.println("enter id for withdraw");  q=sc.nextInt();    float o;  if(q<a.size()){  System.out.println("enter money");  o=sc.nextFloat();  a.get(q).withdraw(o);  }  else  {  System.out.println("invalid id");  }      break;  case 3:  System.out.println("enter id for deposit");  q=sc.nextInt();    float oo;  if(q<a.size()){  System.out.println("enter money");  oo=sc.nextFloat();  a.get(q).deposit(oo);  }  else  {  System.out.println("invalid id");  }    break;  case 4:  int payer;  int getter;  float amount;  System.out.println("enter id who want to trasfer(payer)");  payer=sc.nextInt();  System.out.println("enter id of receivant (money getter)");  getter=sc.nextInt();  if((payer<a.size())&&(getter<a.size()))  {  System.out.println("enter amount");  amount=sc.nextFloat();  a.get(getter).deposit(a.get(payer).transfer(amount));  }  else  {  System.out.println(a.size());  System.out.println("invalid id");  }    break;  case 5:  a.add(new Automatictellermachine());  break;  case 6:    System.out.println("enter id to remove");  q=sc.nextInt();  if(q<a.size())  {a.remove(q);}  else  {  System.out.println("invalid id");  }    break;  case 7:  exit(1);    break;  default:  System.out.println("enter valid choice please!!");  }}}}  **OUTPUT** |
|  | **PART-III Package & Interface** |
| **1** | WAP that illustrate the use of interface reference. Interface Luminious Object has two method lightOn() and lightOff(). There is one class Solid extended by 2 classes Cube and Cone. There is one class LuminiousCone extends Cone and implements Luminoius Interface. LumminuiousCube extends Cube and implements Luminious Interface. Create a object of LuminiousCone and LuminousCube and use the concept of interface reference to invoke the methods of interface.  PROGRAM:  package robot;  interface LuminousObject  {  default void lighton()  {  System.out.println("Light on in interface");  }  default void lightoff()  {  System.out.println("Light off in interface");  }  }  class Solid  {    }  class cube extends Solid  {    }  class cone extends Solid  {    }  class LuminiousCone extends cone implements LuminousObject  {  public void reference()  {  LuminousObject.super.lighton();  LuminousObject.super.lightoff();  }  public void lighton()  {  System.out.println("light is on in cone");  }  public void lightoff()  {  System.out.println("Light is off in cone");  }  }  class LuminousCube extends cube implements LuminousObject  {    public void reference()  {  LuminousObject.super.lighton();  LuminousObject.super.lightoff();  }  public void lighton()  {  System.out.println("light is on in cube");  }  public void lightoff()  {  System.out.println("Light is off in cube");  }  }  public class Demo {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  LuminiousCone lCone = new LuminiousCone();  LuminousCube lCube = new LuminousCube();  lCone.reference();  lCone.lighton();  lCone.lightoff();  lCube.reference();  lCube.lighton();  lCube.lightoff();  } }  **OUTPUT** |
| **2** | WAP that illustrate the interface inheritance. Interface P is extended by P1 and P2 interfaces. Interface P12 extends both P1 and P2. Each interface declares one method and one constant. Create one class that implemetns P12. By using the object of the class invokes each of its method and displays constant.  PROGRAM:  package pract2;  import java.awt.PointerInfo;  interface p  { int a=1;  void a();}  interface p1 extends p  { int b=2;  void b();}  interface p2 extends p  { int c=3;  void c();}  interface p12 extends p1,p2  {}  public class Pract2 implements p12{  public void a()  { System.out.println("a = " + a); }  public void b()  { System.out.println("b = " + b); }  public void c()  {  System.out.println("c = " + c);  }  public static void main(String[] args) {  // TODO code application logic here  Pract2 pract2 = new Pract2();  pract2.a();  pract2.b();  pract2.c();  System.out.println("a = " + a);  System.out.println("b = " + b);  System.out.println("c = " + c);  }  }  **OUTPUT** |
| **3** | Create an abstract class Robot that has the concretre subclasses , RobotA, RobotB, RobotC. Class RobotA1 extends RobotA, RobotB1 extends RobotB and RobotC1 extends RobotC. There is interface Motion that declares 3 methods forward(), reverse() and stop(), implemented by RobotB and RobotC. Sound interface declare method beep() implemented by RobotA1, RobotB1 and RobotC1. Create an instance method of each class and invoke beep() and stop() method by all objects.  PROGRAM:  package robot;  interface Motion  {  void forward();  void reverse();  void stop();  }  interface Sound  {  void beep();  }  abstract class Robot  {  int position;  public Robot() {  }  public Robot(int position) {  this.position = position;  }    public void beep()  {  System.out.println(" beep in Robot Parent" );  }  public void forward() {  System.out.println("forward Robot Parent");  position++;  }    public void reverse() {  System.out.println("reverse Robot Parent");  position--;}    public void stop() {  System.out.println("stop Robot Parent position is "+position); }  }  class RobotA extends Robot implements Sound  {  public RobotA() {  }  public void beep()  {  System.out.println("beep in Robot A");  }  }  class RobotB extends Robot implements Motion,Sound  {  public RobotB() {  }  public void beep()  {  System.out.println("beep in Robot B" );  }  public void forward() {  System.out.println("forward B");  position++;  }  public void reverse() {  System.out.println("reverse B"); position--;}    public void stop() {  System.out.println("stop B position is "+position );}    }  class RobotC extends Robot implements Motion,Sound  {  public RobotC() {  }  public void beep()  {  System.out.println("beep in Robot C");  }  public void forward() {  System.out.println("forward C"); position++;  }    public void reverse() {  System.out.println("reverse C"); position--;}    public void stop() {  System.out.println("stop C position is "+position); }  }  class RobotA1 extends RobotA  {}  class RobotB1 extends RobotB  {}  class RobotC1 extends RobotC  {}  public class demo {  public static void main(String[] args) {  // TODO code application logic here  RobotA1 robotA1 = new RobotA1();  RobotB1 robotB1 = new RobotB1();  RobotC1 robotC1 = new RobotC1();  robotA1.beep();  robotA1.forward();  robotA1.reverse();  robotA1.stop();  robotB1.beep();  robotB1.forward();  robotB1.reverse();  robotB1.stop();  robotC1.beep();  robotC1.forward();  robotC1.reverse();  robotC1.stop();  }  }  **OUTPUT** |
| **4** | Develop a Program that illustrate method overriding concept  PROGRAM:  package override;  interface a  { public void interfacemethod();}  abstract class Abs implements a  {  public Abs() {  System.out.println("Abstract object formed!!");  }  abstract public void disp();  }  class ABC{  public ABC() {  }  public void disp()  {  System.out.println("disp() method of parent class");  }  public void abc()  {  System.out.println("abc() method of parent class");  }  }  class Override extends ABC{  public Override() { }  public void disp(){ System.out.println("disp() method of Child class"); }  public void xyz(){  System.out.println("xyz() method of Child class"); }  public int method()  {  try  {System.out.println("try wants to return 1000");return 1000;}  finally  {System.out.println("finnally callled overridind return");return 5000;}  }  public static void main( String args[]) {  System.out.println("Abs is abstact class implementing a.ABC is parent and Override is child of ABC");  //Parent class reference to child class object  System.out.println("ABC abc=new ABC();");  ABC abc=new ABC();  abc.disp();  abc.abc();    System.out.println("Override o = new Override();");  Override o = new Override();  o.abc();  o.disp();  o.xyz();    System.out.println("ABC obj = new Override();");  ABC obj = new Override();  obj.disp();  obj.abc();  //obj.xyz();    System.out.println("TRICKY Override -> Override oo=new Override()");  Override oo=new Override(){public void abc(){System.out.println("OVERRIDE ABC EXPLICITLY IN MAIN BY OVERRIDE");}public void disp(){System.out.println("OVERRIDE EXPLICITLY DISPLAY IN MAIN BY OVERRIDE");}public void xyz(){System.out.println("OVERRIDE EXPLICITLY XYZ IN MAIN BY OVERRIDE");}};  oo.abc();  oo.disp();  oo.xyz();  System.out.println("TRICKY ABSTRACT OBJECT IMPLEMENTING INTERFACE a ->Abs abs = new Abs()");  Abs abs = new Abs(){public void disp(){System.out.println("DECLARATION OF DISP IN MAIN BY OVERRIDE");} public void interfacemethod(){System.out.println("INTERFACE METHOD DEFINATION IN MAIN BY OVERRIDING");}};  abs.disp();    System.out.println("TRICKY OVERIDING THE RETURNED VALUE!!");  Override ooo = new Override();  System.out.println("returned value is "+ooo.method());  }}  **OUTPUT** |
| **5** | Write a java program which shows importing of classes from other user define packages.  Packages->Source Packages-> pack1 -> pack1.java  Packages->Source Packages-> packages->demo.java  Packages->Source Packages-> packages->pack2.java  **PUBLIC STATIC VOID MAIN IS** **IN demo.java**  ***Program pack1.java:***  package pack1;  public class pack1 {  void defaultmethod()  {  System.out.println("Hello from pack1 default method");  }  public void method()  {  System.out.println("Hello from pack1 public method");  }  private void privatemethod()  {  System.out.println("Hello from pack1 private method");  }  protected void protectedmethod()  {  System.out.println("Hello from pack1 protected method");  }  }  ***Program demo.java:***  package pakages;  import pack1.pack1;  import pakages.pack2;  public class demo {  void defaultmethod()  {  System.out.println("Hello from packages default method");  }  public void method()  {  System.out.println("Hello from packages public method");  }  private void privatemethod()  {  System.out.println("Hello from packages private method");  }  protected void protectedmethod()  {  System.out.println("Hello from packages protected method");  }  public static void main(String[] args) {  pack1 p = new pack1();  pack2 pa = new pack2();  demo pac = new demo();  p.method();  pa.defaultmethod();  pa.method();  pa.protectedmethod();  pac.defaultmethod();  pac.privatemethod();  pac.protectedmethod();  pac.method();  }    }  ***Program pack2.java:***  package pakages;  public class pack2 {  void defaultmethod()  {  System.out.println("Hello from packages.pack2 default method");  }  public void method()  {  System.out.println("Hello from packages.pack2 public method");  }  private void privatemethod()  {  System.out.println("Hello from packages.pack2 private method");  }  protected void protectedmethod()  {  System.out.println("Hello from packages.pack2 protected method");  }  }  **OUTPUT** |
| **6** | Write a program that demonstrates use of packages & import statements.  Packages->Source Packages-> pack1 -> pack1.java  Packages->Source Packages-> packages->demo.java  Packages->Source Packages-> packages->pack2.java  **PUBLIC STATIC VOID MAIN IS** **IN demo.java**  ***Program pack1.java:***  package pack1;  public class pack1 {  void defaultmethod()  {  System.out.println("Hello from pack1 default method");  }  public void method()  {  System.out.println("Hello from pack1 public method");  }  private void privatemethod()  {  System.out.println("Hello from pack1 private method");  }  protected void protectedmethod()  {  System.out.println("Hello from pack1 protected method");  }  }  ***Program demo.java:***  package pakages;  import pack1.pack1;  import pakages.pack2;  public class demo {  void defaultmethod()  {  System.out.println("Hello from packages default method");  }  public void method()  {  System.out.println("Hello from packages public method");  }  private void privatemethod()  {  System.out.println("Hello from packages private method");  }  protected void protectedmethod()  {  System.out.println("Hello from packages protected method");  }  public static void main(String[] args) {  pack1 p = new pack1();  pack2 pa = new pack2();  demo pac = new demo();  p.method();  pa.defaultmethod();  pa.method();  pa.protectedmethod();  pac.defaultmethod();  pac.privatemethod();  pac.protectedmethod();  pac.method();  }    }  ***Program pack2.java:***  package pakages;  public class pack2 {  void defaultmethod()  {  System.out.println("Hello from packages.pack2 default method");  }  public void method()  {  System.out.println("Hello from packages.pack2 public method");  }  private void privatemethod()  {  System.out.println("Hello from packages.pack2 private method");  }  protected void protectedmethod()  {  System.out.println("Hello from packages.pack2 protected method");  }  }  **OUTPUT** |
| **7** | Write a program that illustrates the significance of interface default method.  Program:  package interfacedefault;  interface a  {  default void display()  {  System.out.println("interface default method");  }  }  public class Interfacedefault implements a{  public static void main(String[] args) {  // TODO code application logic here  Interfacedefault b = new Interfacedefault();  b.display();  }    }  **OUTPUT** |
|  | **PART-IV Exception Handling** |
| 1 | WAP to show the try - catch block to catch the different types of exception.  Program:  package javaapplication33;  import java.io.DataInputStream;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.util.Scanner;  class Division  {  float divide(int a, int b) throws ArithmeticException  {  return(a/b);  }  }  public class ExceptionDemo1  {  static int demoproc(String s) {  try {  if (s == null) {  throw new NullPointerException("String is null.");  } else {  return s.length();  }  } catch (NullPointerException e) {  System.out.println("NullPointerException Caught inside demoproc method...");  throw e; // rethrow the exception  }  }  public static void main(String[] args)  {  try  {  int a = 5/0;  System.out.println("Value of a = "+a);  }  catch(ArithmeticException e)  {  System.out.println("Exception = "+e);  }  finally  {  //finally block is executed compulsorily irrespective of the occurrence of an error  System.out.println("finally block executes ");  }  System.out.println("Rest of the code...");  try  {  int[] a = new int[5];  a[2] = a[10]/0+2; // In this line a[10] is evaluated first, so handle ArrayIndexOutOfBounds first and then ArithmeticException... see below example  System.out.println("After exception occurs...");  }  catch(ArrayIndexOutOfBoundsException aioob)  {  System.out.println("Array index Out of Bounds..");  }  try {  int[] a = new int[5];    try {  a[2] = a[10] / 0 + 2;  } catch(ArrayIndexOutOfBoundsException aioe)  {  System.out.println("Access of invalid index in array - index out of range...");  }  catch(ArithmeticException ae)  {  System.out.println("Enter valid denominator..");  }finally {  System.out.println("Within Inner finally block.....");  }  } catch(Exception e)  {  System.out.println("Unexpected exception occured... :::"+e);  } finally {  System.out.println("Finally block executed.....");  }  System.out.println("After fially block .......");  try  {  Class c = Class.forName("p1.ExceptionDemo3");  System.out.println("2nd line");  }  catch(ClassNotFoundException e)  {  System.out.println("Class not found..");  }  try  {  FileInputStream fis = new FileInputStream("C:\\Arithmetic.java");  DataInputStream ds = new DataInputStream(fis);  }  catch(FileNotFoundException f)  {  System.out.println("File not found..");  }  char c;  String s = null;  try  {  int n = s.length();  }  catch(NullPointerException e)  {  System.out.println(e);  }int n1,n2;  float ans;  Scanner ss = new Scanner(System.in);  System.out.println("\nEnter number 1 :");  n1 = ss.nextInt();  System.out.println("\nEnter number 2 :");  n2 = ss.nextInt();  Division d = new Division();  try  { ans = d.divide(n1, n2);  System.out.println("Ans = "+ans);  }  catch(ArithmeticException ae)  { System.out.println("Denominator cant be zero."); // Write User-friendly message here}  try {  String sss = null;  int len = demoproc(sss);  System.out.println("Lenght of the String is = " + len);  } catch (NullPointerException e) {  System.out.println("NullPointerException is caught in caller method if callee method not able to handle it.. " + e);  } } }  OUTPUT |
| 2 | WAP to generate user defined exception using “throw” and “throws” keyword.  Program:  class ThrowDemo {  static int demoproc(String s) throws NullPointerException{  try {  if (s == null) {  throw new NullPointerException("String is null.");  } else {  return s.length();  }  } catch (NullPointerException e) {  System.out.println("NullPointerException Caught inside demoproc method...");  throw e; // rethrow the exception  }  }  public static void main(String args[]) {  try {  String s = null;  int len = demoproc(s);  System.out.println("Lenght of the String is = " + len);  } catch (NullPointerException e) {  System.out.println("NullPointerException is caught in caller method if callee method not able to handle it.. " + e);  }  }  }  OUTPUT |
| 3 | Write a program that raises two exceptions. Specify two ‘catch’ clauses for the two exceptions. Each ‘catch’ block handles a different type of exception. For example the exception could be ‘ArithmeticException’ and ‘ArrayIndexOutOfBoundsException’. Display a message in the ‘finally’ block.  Program:  package javaapplication34;  public class JavaApplication34 {  public static void main(String[] args) {  try  { int a = 5/0;  System.out.println("Value of a = "+a); }  catch(ArithmeticException e)  { System.out.println("Exception = "+e); }  try  { int[] a = new int[5];  a[2] = a[10]/0+2; // In this line a[10] is evaluated first, so handle ArrayIndexOutOfBounds first and then ArithmeticException... see below example  System.out.println("After exception occurs..."); }  catch(ArrayIndexOutOfBoundsException a)  { System.out.println("Array index Out of Bounds.."); }    finally  {  //finally block is executed compulsorily irrespective of the occurrence of an error  System.out.println("finally block executes after arithmetic exception occured and array out of bound occured");  }  }  }  OUTPUT |
|  | **PART-V**  **File Handling & Streams** |
| 1 | WAP to show how to create a file with different mode and methods of File class to find path, directory etc.  Program:  import java.io.\*;  import java.util.Date;  public class f2 {  public static void main(String args[])  {  File f = new File("D:\\f2.txt");  try {  if (f.createNewFile()) {  System.out.print("File Created");  } else {  System.out.print("File Already exists");  }  } catch (IOException e) {  System.out.print(e.getMessage());  }  System.out.println("canRead()=" + f.canRead());  System.out.println("canWrite()=" + f.canWrite());  f.setWritable(true);  System.out.println("canWrite()=" + f.canWrite());  System.out.println("canRead()=" + f.canRead());  System.out.println("setReadOnly()=" + f.setReadOnly());  System.out.println("canWrite()=" + f.canWrite());  System.out.println("canRead()=" + f.canRead());  System.out.println("getParent()=" + f.getParent());  System.out.println("getpath()=" + f.getPath());  System.out.println("getAbsolutePath()=" + f.getAbsolutePath());  System.out.println("File Exists()=" + f.exists());  System.out.println("isFile()=" + f.isFile());  System.out.println("isDirectory()=" + f.isDirectory());  System.out.println("LastModified()=" + new Date(f.lastModified()));  System.out.println("File length()=" + f.length());  //rename a File  File f1 = new File("D:\\f11.txt");  File f2 = new File("D:\\f2.txt");  try {  if (f1.createNewFile()&&f2.createNewFile()) {  System.out.println("File Created");  } else {  System.out.println("File Already exists");  }  } catch (IOException e) {  System.out.print(e.getMessage());  }  if (f1.renameTo(f2)) {  System.out.println("File Renamed");  } else {  System.out.println("File can't renamed");  }  //if file exists delete it  File f5 = new File("D:\\f2.txt");  if (f5.exists()) {  System.out.println("File Exists");  if (f5.delete()) {  System.out.println("File Deleted Successfully");  } else {  System.out.println("File can't be Deleted");  }  }  //make directories mkdirs  File f3 = new File("D:\\f3");  if (f3.mkdirs()) {  System.out.println("Directories created");  } else {  System.out.println("Directories can't created");  }  //make directory mkdir a folder is going to create  File f4 = new File("D:\\f4");  if (f4.mkdir()) {  System.out.println("Directory created");  } else {  System.out.println("Directory can't created");  }  //file object is directory or file  System.out.println("is directory " + f4.isDirectory());  }}  OUTPUT |
| 2 | Write a program to show a tree view of files and directories under a specified drive/volume.  Program:  package javaapplication37;  import java.io.\*;  public class directory {  String space=" ";  void fill(String a)  {  File f= new File(a);  String p[]=f.list();  if(p!=null)  for(int i=0;i<p.length;i++)  {  File z=new File(a+"\\"+p[i]);  if(z.isFile()==true)  {System.out.println(space+p[i]);}  if(z.isDirectory()==true)  {  String k=space;  System.out.println(space+p[i]);  space=space+" ";  fill(a+'\\'+p[i]);  space=k;  }}}  public static void main(String [] args)  {new directory().fill("c:\\");}  }  OUTPUT |
| 3 | Write a program to transfer data from one file to another file so that if the destination file does not exist, it is created.  Program:  package file;  import java.io.\*;  public class Filee {  public static void main(String[] args) throws Exception {  FileInputStream in = new FileInputStream(new File("srcfile.txt"));  File f1=new File("destnfile.txt");  if(f1.exists()!=true)  {  f1.createNewFile();  }  FileOutputStream out = new FileOutputStream(new File("destnfile.txt"));    byte[] buf = new byte[1024];  int len;    while ((len = in.read(buf)) > 0) {  out.write(buf, 0, len);  }  in.close();  out.close();  BufferedReader in1 = new BufferedReader(new FileReader("destnfile.txt"));  String str;    while ((str = in1.readLine()) != null) {  System.out.println(str);  }  in.close();  }  }  OUTPUT |
| 4 | WAP to show use of character and byte stream.  Program:  package fille;  import java.io.\*;  import java.util.Scanner;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Fille {  public static void main(String[] args) {  File f = new File("binaryfile.txt");  Scanner sc = new Scanner(System.in);  try{  if(!f.exists())  { f.createNewFile(); }  FileOutputStream fos = new FileOutputStream(f);  int i;  System.out.println("WRITE A BINARY FILE type exit to end");  String j=new String("");  j=sc.nextLine();  while(j.compareTo("exit")!=0){  fos.write((j+"\n").getBytes());  j=sc.nextLine();  }  FileInputStream fis = new FileInputStream(f);  System.out.println("\n\nREAD A BINARY FILE");  while((i=fis.read())!=-1)  { System.out.print((char)i); }  fos.close();  fis.close();  }catch (IOException ex) { System.out.println(ex); }  File f2 = new File("characterfile.txt");  try {  System.out.println("\n\nWRITTING A TEXT FILE:TYPE exit TO EXIT");  BufferedWriter bufferedWriter =new BufferedWriter(new FileWriter(f2));  String s;  s=sc.nextLine();  while(s.compareTo("exit")!=0)  {  bufferedWriter.write(s);  s=sc.nextLine();  bufferedWriter.newLine();  }  bufferedWriter.close();  String line = null;  System.out.println("\n\nREADING A TEXT FILE");  BufferedReader bufferedReader =new BufferedReader(new FileReader(f2));  while((line = bufferedReader.readLine()) != null) {  System.out.println(line);}  bufferedReader.close();  }  catch(Exception ex) {  }  //copying binary file  System.out.println("COPYING BINARY FILE");  try{  FileInputStream in = new FileInputStream(new File("binaryfile.txt"));  File f1=new File("destnbinaryfile.txt");  if(f1.exists()!=true)  {  f1.createNewFile();  }  FileOutputStream out = new FileOutputStream(new File("destnbinaryfile.txt"));    byte[] buf = new byte[1024];  int len;    while ((len = in.read(buf)) > 0) {  out.write(buf, 0, len);  }  in.close();  out.close();  BufferedReader in1 = new BufferedReader(new FileReader("destnbinaryfile.txt"));  String str;    while ((str = in1.readLine()) != null) {  System.out.println(str);  }  in.close();  }  catch (Exception ex) {}  }}  OUTPUT |
| 5 | WAP to read console input and write them into a file. (BufferedReader /BufferedWriter).  Program:  package aaa;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.FileNotFoundException;  import java.io.FileReader;  import java.io.FileWriter;  import java.io.IOException;  import java.util.Scanner;  public class Aaa {  public static void main(String[] args) {  String fileName;  Scanner sc = new Scanner(System.in);  System.out.println("enter file name");  fileName=sc.next();  try {  System.out.println("WRITTING A TEXT FILE:TYPE exit TO EXIT");  BufferedWriter bufferedWriter =new BufferedWriter(new FileWriter(fileName));  String s;  s=sc.nextLine();  while(s.compareTo("exit")!=0)  {  bufferedWriter.write(s);  s=sc.nextLine();  bufferedWriter.newLine();  }  bufferedWriter.close();  }  catch(Exception ex) {  }  String line = null;  System.out.println("READING A FILE");  try { BufferedReader bufferedReader =new BufferedReader(new FileReader(fileName));  while((line = bufferedReader.readLine()) != null) {  System.out.println(line); }  bufferedReader.close();  }  catch (Exception ex) { } } }  OUTPUT |
| 6 | WAP to demonstrate methods of wrapper class.  package wrap;  public class Wrap {  public static void main(String[] args) {  Integer a=new Integer(20);  System.out.println(a);    int i=a;  System.out.println(i);    int j=a.intValue();  byte b=a.byteValue();  float f=a.floatValue();  short s=a.shortValue();  long l=a.longValue();  System.out.println(j);  System.out.println(b);  System.out.println(f);  System.out.println(s);  System.out.println(l);      String str=a.toString();  System.out.println(str);      Integer n=Integer.valueOf(str);  System.out.println(n);      int c=Integer.parseInt(str);  System.out.println(c);  }    }  OUTPUT |
|  | **PART-VI**  **Multithreading** |
| 1 | Write a program to create thread which display “Welcome to CHARUSAT!!” message. A. by extending Thread class B. by using Runnable interface.  Program:  package wel;  import static java.lang.Thread.sleep;  class a extends Thread  {  public void run()  {  try  { sleep(1000); }  catch (Exception ex) { }  System.out.println("welcome to charusat by thread");  }  }  class b implements Runnable  {  public void run()  {  try  {  sleep(1000);  }  catch (Exception ex) { }  System.out.println("welcome to charusat by runnable interface");    }  }  public class Wel {  public static void main(String[] args) {  // TODO code application logic here  a obj = new a();  obj.start();  b objb = new b();  new Thread(objb).start();  }    }  OUTPUT |
| 2 | Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.  Program:  import java.util.Scanner;  class Multi3 extends Thread {  int st, en, sum = 0;  Multi3(int start, int end) {  st = start;  en = end;  sum = 0;  }  public void run() {  for (int i = st; i <= en; i++) {  sum = sum + i;  }  }  }  public class a {  public static void main(String args[]) throws InterruptedException  {  int sum = 0;  int N, n, i, j, k;  Scanner sc = new Scanner(System.in);  System.out.println("Enter N to find sum:");  N = sc.nextInt();  System.out.println("Enter number of threads:");  n = sc.nextInt();  Multi3[] m = new Multi3[n];  k = N / n;  j = 1;  for (i = 0; i < n - 1; i++) {  m[i] = new Multi3(j, j + k);  j = j + k + 1;  }  m[n - 1] = new Multi3(j, N);    for (i = 0; i < n; i++) {  m[i].start();  }  for (i = 0; i < n; i++) {    m[i].join();  }  sum = 0;  for (i = 0; i < n; i++) {  sum = sum + m[i].sum;  }    System.out.println("Sum=" + sum);  }  }  OUTPUT: |
| 3 | Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.  Program:  package inc;  class a extends Thread  {  int i=0;  public void run()  {while(true){  i++;  try  {  System.out.println(i);  sleep(1000);  }  catch (Exception ex) {    }}  }  }  public class Inc {  public static void main(String[] args) {  a obj = new a();  obj.start();      }    }  OUTPUT: |
| 4 | Write a program to create three threads ‘FIRST’, ‘SECOND’, ‘THIRD’. Set the priority of the ‘FIRST’ thread to 3, the ‘SECOND’ thread to 5(default) and the ‘THIRD’ thread to 7.  Program:  /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package thr;  class a extends Thread  {  public void run()  {  System.out.println("thread "+this.getName()+this.getPriority());  }  }  public class Thr {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  // TODO code application logic here  a obj = new a();  a obj1 = new a();  a obj2 = new a();  obj.setName("FIRST");  obj1.setName("SECOND");  obj2.setName("Third");  obj.setPriority(3);  obj1.setPriority(5);  obj2.setPriority(7);  obj.start();  obj1.start();  obj2.start();  }    }  OUTPUT: |
| 5 | Write a program to solve producer-consumer problem using thread synchronization.  Program:  class Q{  int n;  boolean valueSet = false;  synchronized int get(){  if(!valueSet)  try{  wait();  }catch(InterruptedException ie){    }  System.out.println("Got:"+n);  valueSet = false;  notify();  return n;  }  synchronized void put(int n){  if(valueSet)  try{  wait();  }catch(InterruptedException ie){    }  this.n = n;  valueSet = true;  System.out.println("Put:"+n);  notify();  }  }  class Producer implements Runnable{  Q q;  Producer(Q q){  this.q =q;  new Thread(this, "Producer").start();  }    public void run(){  int i=0;    while(true){  q.put(i++);  }  }  }  class Consumer implements Runnable{  Q q;  Consumer(Q q){  this.q =q;  new Thread(this, "Consumer").start();  }    public void run()  int i=0;    while(true){  q.get();  }  }  }  public class PC{  public static void main(String[] args){  Q q = new Q();  new Producer(q);  new Consumer(q);    System.out.println("Press Control - C to stop.");  }  }  OUTPUT: |
|  | **PART-VII Collection Framework and Generic** |
| 1 | Create a generic method for sorting an array of Comparable objects.  Program:  package sortingsimpleiii;  import java.util.Scanner;  public class SortingSimpleIII  {  public static <T extends Comparable<T>> void bubblesort(T[] elements)  {  for(int i=0;i<elements.length;i++)  for(int j=0;j<i;j++)  {  if(elements[i].compareTo(elements[j])==-1)  {  T temp=elements[i];  elements[i]=elements[j];  elements[j]=temp;  }  }  }    public static void main(String[] args) {  int n;  Scanner sc = new Scanner(System.in);  System.out.println("\nhow many Bytes you wanted to sort");  n=sc.nextInt();  Byte a[]=new Byte[n];    System.out.println("enter "+n+" bytes ");  for (int i = 0; i < n; i++) {  a[i]=sc.nextByte();  }  bubblesort(a);  for (Byte byte1 : a) {  System.out.print(byte1+" ");  }  System.out.println("\nhow many Float you wanted to sort");  n=sc.nextInt();  Float aa[]=new Float[n];  System.out.println("enter "+n+" Float ");  for (int i = 0; i < n; i++) {  aa[i]=sc.nextFloat();  }  bubblesort(aa);  for (Float fl : aa) {  System.out.print(fl+" ");  }  System.out.println("\nhow many Integer you wanted to sort");  n=sc.nextInt();  Integer aaa[]=new Integer[n];  System.out.println("enter "+n+" Integer ");  for (int i = 0; i < n; i++) {  aaa[i]=sc.nextInt();  }  bubblesort(aaa);  for (Integer byte1 : aaa) {  System.out.print(byte1+" ");  }  }  }  OUTPUT: |
| 2 | Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.  Program:  package word;  import java.util.Scanner;  import java.util.\*;  public class Word {  public static void main(String[] args) {  Scanner sc=new Scanner(System.in);  String str;  str=sc.nextLine();  StringTokenizer s = new StringTokenizer(str);  int count=s.countTokens();  String k[]=new String[s.countTokens()];  int i=0;  while(s.countTokens()!=0)  {  k[i]=s.nextToken();  i++;  }  System.out.println("SORTING USING TREESET ");  TreeSet<String> treeset =new TreeSet<String>();  for (int j = 0; j < count; j++) {  treeset.add(k[j]);  }    for (String string : treeset) {  System.out.println(string);  }  System.out.println("SORTING USING TREEMAP");  TreeMap<String,Integer> treemap = new TreeMap<String,Integer>();    for (int j = 0; j < count; j++) {  treemap.put(k[j],j);  }  for(Map.Entry<String,Integer> en:treemap.entrySet())  {  System.out.println(en.getValue()+"->"+en.getKey());  } } }  OUTPUT: |
| 3 | Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains method to test if a word is in the keyword set.  Program:  package keywords;  import java.io.BufferedReader;  import java.io.File;  import java.io.FileReader;  import java.util.HashSet;  import java.util.Scanner;  import java.util.StringTokenizer;  public class Keywords {  public static void main(String[] args) {  String s[]={"abstract","assert","boolean","break","byte","case","catch","char","class","continue","default","do",  "double","else","enum","extends","final","finally","float","for","if","implements","import","instanceof",  "int","interface","long","native","new,package","private","protected","public","return","short","static",  "super","switch","synchronized","this","throw","throws","transient","try","void","volatile","while"};  HashSet<String> h=new HashSet<String>();  for (String string : s) {  h.add(string);  }  String a;  int count=0;  System.out.println("ENTER FILE PATH:");  Scanner sc = new Scanner(System.in);  a=sc.next();    File f = new File(a);  try{  BufferedReader bufferedReader =new BufferedReader(new FileReader(f));  String line;  while((line = bufferedReader.readLine()) != null) {  StringTokenizer st = new StringTokenizer(line);  String k[]=new String[st.countTokens()];  int i=0;  while(st.countTokens()!=0)  {  k[i]=st.nextToken();  i++;  }  for (String str : k) {  if(h.contains(str))  { count++; }  }  System.out.println(line);  }System.out.println("number of keywords used="+count);  bufferedReader.close();  }  catch (Exception ex) { }  }  }  OUTPUT: |