

Advanced Programming lab Practical File (CSP 6025)

Submited to: Mr. Anuj Mahajan Submitted by: Tavneet Singh Enrollment No. : (Enter your no. here)

Tavneet singh (Enter your no. here))

**Advanced Programming Lab Practical File**

**Index**

|  |  |  |
| --- | --- | --- |
| **S no.** | **Practical name** | **Page no.** |
| 1 | Input values using Scanner and display on the screen | 4 |
| 2 | Demonstrate loop and conditional statements: for-loop, while loop, if-else condition, if-else if-else, switch, break and continue statements. | 5-7 |
| 3 | Demonstrate the use of primitive datatypes available n Java and conversion between them using typecasting (if possible) to understand lossy/non-lossy conversions. | 8-11 |
| 4 | Create the following patterns, (number of rows shall be entered by user at run time):  \* 1 \* 1  \*\* 1 2 \* \* \* 2 1 2  \*\*\* 1 2 3 \* \* \* \* \* 3 2 1 2 3  \*\*\*\* 1 2 3 4 \* \* \* 2 1 2  \*\*\*\*\* 1 2 3 4 5 \* 1 | 12-15 |
| 5 | Create Even Series, odd series, Fibonacci Series, Prime Series, Sin series, Cosine series. | 16-18 |
| 6 | Demonstrate various functions available in Maths class | 19-21 |
| 7 | Demonstrate the use of Biginteger and BigDecimal classes to perform operations on very large integer and decimals. | 22-23 |
| 8 | Demonstrate the creation and usage of static/non-static functions. | 24 |
| 9 | Demonstrate the use of Getters and Setters functions. | 25-27 |
| 10 | Demonstrate various kinds of comments available in java including documentation comments. | 28 |
| 11 | Demonstrate the use of enumerate data type | 29-30 |
| 12 | Demonstrate the use of various arithmetic, logical, bitwise, unary, ternary, assignment operators and understand their precedence and hierarchy(L to R, R to L). | 31-33 |
| 13 | Demonstrate the use 1D, 2D array and various functions available in Array class. | 34-37 |
| 14 | Demonstrate the concept of Array of objects, create Students class. Ask user at run time for the no. of students to be created. Create Students objects, initialize their values using setter functions. Display all the student’s details using getters functions. | 38-40 |
| 15 | WAP to create 100 random numbers( range 1 to 100) and store then in array, sort the array and display the sorted array using enhanced for loop | 41-45 |
| 16 | Demonstrate the Matrix addition, subtraction, multiplication using 2D arrays | 46-49 |
| 17 | WAP to solve MagicBoard problem(CodeChef) using 2D arrays. | 50-54 |
| 18 | WAP to create Tic-Tac-Toe game (2 players version) | 55-60 |
| 19 | WAP to create Tic-Tac-Toe game ( VS computer version) | 61-64 |
| 20 | WAP to create Rock-Pape Scissors game | 65-66 |
| 21 | Demonstrate the functions available in StringBuilder/StringBuffer classes. | 67-68 |
| 22 | Write program to demonstrate various String/String Buffer/String Builder functions for: Reversing a string( and check Palindrome), Conversion to UpperCase/LowerCase,  Equality check of given String, Inserting/Deleting a character in string, Fetching the index of a particular character/substring, Replacing the occurrence of a substring/character in a String, comparing strings Lexicographically, splitting a string based on a given token etc. | 69-70 |
| 23 | You are given a string at run time You are required to capitalize the first letter of every word in that string. | 71 |
| 24 | You are given a string at run time. Capitalize all the characters at all the odd indexes in the string e.g., at index 1, 3, 5, etc. | 72 |
| 25 | You are given a string at run time. You are required to fetch the remaining String after the occurrence of a certain sequence/character. E.g., fetch the remaining String after the occurrence of “and” in the String. | 73 |
| 26 | You are given a string at run time. Count the occurrences of a given word/sequence in that string. | 74-75 |
| 27 | You are given a string at run time. Delete all the duplicate words from the String(ignore case) and return modified String. | 76 |
| 28 | You are Given a String at run time. Find total vowels in the given String and also display the count of each vowel in the string. | 77-78 |
| 29 | You are given a string at run time. Replace all the characters at the even index by the character present at the next even index, Replace the character at the last even index by “1” e.g., “HELLO” shall return LEOL1. | 79-80 |
| 30 | Demonstrate parent child relationship among classes add call the method available by virtue of inheritance in child class. | 81 |
| 31 | Demonstrate method overloading add overriding add the use of super keyword in assessing parent class constructor add overridden methods. | 82-83 |
| 32 | Demonstrate that a parent class reference variable can be used to point to the subclass object but using a parent class reference variable we can call only those methods which are available in the parent class. | 84-85 |
| 33 | Demonstrate access specified by keywords: public, private protected and default. | 86-87 |
| 34 | (a) Demonstrate the various methods available in ArayList class.  (b) Create an ArrayList of objects(say Student/Employee through the ArrayList to display all the information of objects. | 88-91 |
| 35 | **The University Problem**  You are requested to input the details of the entire University using the concept of Arrays/ArrayList and composition( an object of another object(). At the lowest level three shall be class Student. The field of student classes are rollNo, name & age. These fields have corresponding getters(for fetching value) and setters (for setting value functions. There is a displayDetails() function which shall print all details of a student by using getters to fetch the value from the fields. | 92-97 |
| 36 | WAP to make Database connectivity and insert records into database. | 98-99 |
| 37 | WAP to input a string at runtime, if the string contains the keyword “smvdu” throw a user defined named “SMVDUException”, if the string contains the keyword “university” Throw another exception named “UniversityException”. If the given string contains Both the keywords, throw both the exceptions, your program must handle both of these Exceptions. You may ignore the case of the input string. | 100-101 |
| 38 | WAP to create three threads(including the main thread) either by extending the Thread class or implementing the Runnable interface. The first thread prints Fibonacci series upto 1000, the second thread prints prime no. series upto 1000, the third thread prints multiples of 5 upto 1000. Each thread sleeps for 100 ms after printing a number. Observe the output of the program after repeated runs. Is it the same? | 102-103 |
| 39 | WAP to demonstrate the use of try-catch-finally block, use of multiple catch statements, nested try-catch. | 104 |
| 40 | WAP to implement comparable interface and override .hashCode() and .equals() function. | 105-107 |
| 41 | WAP to show usage of hash table, hash map and linked hash map. | 108-109 |
| 42 | WAP to show usage of TreeSet, LinkedHashSet and HashSet. | 110-112 |

**Practical 1**

**Aim:** Input values using Scanner and display on the screen

**Program:**

*// Program to input values Using Scanner and display on screen*  
  
**import** java.util.Scanner;  
  
**class** TakingInput {  
 **public** **static** **void** main(String[] args) {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the total grant required to complete your research (in Rs.)"**);  
  
 **int** grant = take.nextInt();  
 System.out.println(**"Grant asked by you is Rs."** + grant);  
 }  
}

}

Text

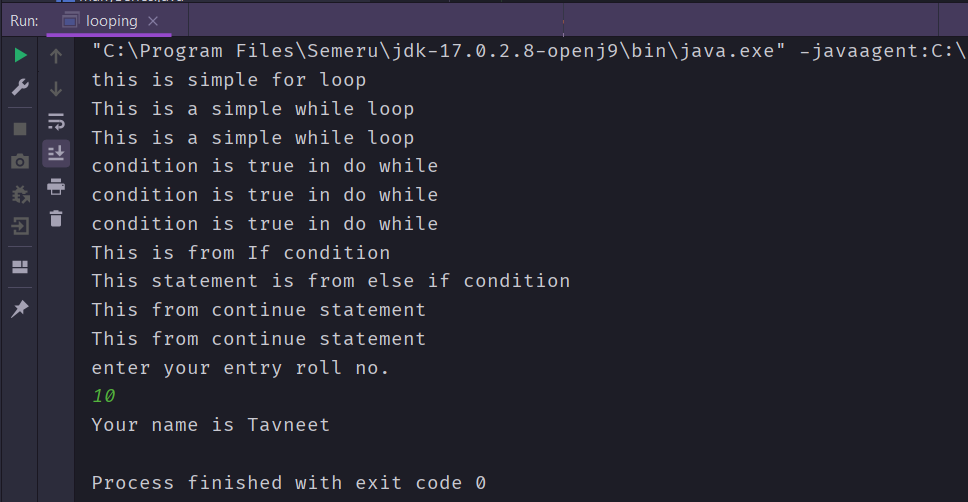
Description automatically generated**Output:**

**Practical 2**

**Aim:** Demonstrate loop and conditional statements: for-loop, while loop, if-else condition, if-else, switch, break and continue statements.

**Program:**

*// Demonstrate loop and conditional statements: for-loop,*  
*// while loop, if-else condition, if-else, switch, break*  
*// and continue statements.*  
  
**import** java.util.Scanner;  
  
**class** All {  
  
 **void** ForLoop() {  
 **for** (**int** i = 1; i < 2; i++) {  
 System.out.println(**"this is simple for loop"**);  
 }  
 }  
  
 **void** WhileLoop() {  
 **int** j = 1;  
 **while** (j < 3) {  
 System.out.println(**"This is a simple while loop"**);  
 j++;  
 }  
 }  
  
 **void** DoWhile() {  
 **int** k = 1;  
 **do** {  
 System.out.println(**"condition is true in do while"**);  
 k++;  
 } **while** (k < 4);  
 }  
  
 **void** IfCondition() {  
 **int** l = 1;  
 **if** (l < 2) {  
 System.out.println(**"This is from If condition"**);  
 } **else** {  
 System.out.println(**"This is from else condition"**);  
 }  
 }  
  
 **void** IfElseCondition() {  
 **int** m = 2;  
 **if** (m < 2) {  
 System.out.println(**"This statement is from If condition"**);  
 } **else** **if** (m < 3) {  
 System.out.println(**"This statement is from else if condition"**);  
 } **else** {  
 System.out.println(**"This statement is from else condition"**);  
 }  
 }  
  
 **void** SwitchStatement() {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"enter your entry roll no."**);  
  
 **int** rollno = take.nextInt();  
 **switch** (rollno) {  
 **case** 10:  
 System.out.println(**"Your name is Tavneet"**);  
 **break**;  
 **case** 9:  
 System.out.println(**" Your Name is Satya"**);  
 **break**;  
 **default**:  
 System.out.println(**"Not in our records"**);  
 }  
 }  
  
 **void** ContinueStatement() {  
 **int** n = 1;  
 **while** (n < 4) {  
 **if** (n == 2) {  
 n++;  
 **continue**;  
 }  
  
  
 System.out.println(**"This from continue statement"**);  
 n++;  
 }  
 }  
}  
  
**class** looping {  
 **public** **static** **void** main(String[] args) {  
 All a = **new** All();  
 a.ForLoop();  
 a.WhileLoop();  
 a.DoWhile();  
 a.IfCondition();  
 a.IfElseCondition();  
 a.ContinueStatement();  
 a.SwitchStatement();  
  
 }  
}

**Output:**

**Practical 3**

**Aim:** Demonstrate the use of primitive datatypes available in Java and conversion between them using typecasting (if possible) to understand lossy/non-lossy conversions.

**Program:**

*// Demonstrate the use of primitive datatypes available*  
*// in Java and conversion between them using typecasting*  
*// (if possible) to understand lossy/non-lossy conversions.*  
**public** **class** PremitiveDataTypes {  
 **public** **static** **void** main(String[] args) {  
 **int** int\_var = 45;  
 **int** int\_var\_b = 20;  
 **double** double\_var = 148.597545d;  
 **double** double\_var\_b = 24.4785d;  
 **long** long\_var = 98785954;  
 **long** long\_var\_b = 874581146;  
 **float** float\_var = 2.718f;  
 **float** float\_var\_b = 3.1415926f;  
 **byte** byte\_var = 112;  
 **byte** byte\_var\_b = 12;  
 **short** short\_var = 7777;  
 **short** short\_var\_b = 451;  
 **char** char\_var\_unicode = 84;  
 **char** char\_var\_alphabet = **'a'**;  
 **boolean** boolean\_var = **true**;  
  
 System.out.println(**"Usage of int Primitive Data type by doing Arithmetic Operations on it\n1) Addition on int: "** + (int\_var + 1) + **"\n 2) Subtraction on int: "** + (int\_var - 2) + **"\n 3) Multiplication on int: "** + (int\_var \* int\_var\_b) + **"\n 4) Division on int: "** + (int\_var / int\_var\_b) + **"\n"**);  
 System.out.println(**"Usage of double Primitive Data type by doing Arithmetic Operations on it\n1) Addition on double: "** + (double\_var + double\_var\_b) + **"\n 2) Subtraction on double: "** + (int\_var - 2.48d) + **"\n 3) Multiplication on double: "** + (double\_var \* double\_var\_b) + **"\n 4) Division on double: "** + (double\_var / double\_var\_b) + **"\n"**);  
 System.out.println(**"Usage of float Primitive Data type by doing Arithmetic Operations on it\n1) Addition on float: "** + (float\_var + float\_var\_b) + **"\n 2) Subtraction on float: "** + (float\_var - 24.4845f) + **"\n 3) Multiplication on float: "** + (float\_var \* float\_var\_b) + **"\n 4) Division on float: "** + (float\_var / float\_var\_b) + **"\n"**);  
 System.out.println(**"Usage of long Primitive Data type by doing Arithmetic Operations on it\n1) Addition on long: "** + (long\_var + long\_var\_b) + **"\n 2) Subtraction on long: "** + (int\_var - int\_var\_b) + **"\n 3) Multiplication on long: "** + (double\_var \* double\_var\_b) + **"\n 4) Division on double: "** + (long\_var / long\_var\_b) + **"\n"**);  
 System.out.println(**"Byte can oly hold value from -128 to 127\n All Airthmetic operations can be applied on byte data type:\n"** + **"1) Addition on byte: "** + (byte\_var + byte\_var\_b) + **"\n 2) Subtraction on byte: "** + (byte\_var - byte\_var\_b) + **"\n 3) Multiplication on byte: "** + (byte\_var \* byte\_var\_b) + **"\n 4) Division on Byte: "** + (byte\_var / byte\_var\_b) + **"\n"**);  
 System.out.println(**"Short data type is half the size of int and twice the size of byte data type\nUsage of short Primitive Data type by doing Arithmetic Operations on it\n 1) Addition on double: "** + (short\_var + 4584) + **"\n 2) Subtraction on short: "** + (short\_var - short\_var\_b) + **"\n 3) Multiplication on short: "** + (short\_var \* short\_var\_b) + **"\n 4) Division on short: "** + (short\_var / short\_var\_b) + **"\n"**);  
 System.out.println(**"char data type is a 16-bit integer representing a Unicode-encoded character (eg: \\u0045 for char C = 45)\n"** + **"Operations on Char datatype:\n"** + **"1)Normal printing value of char variable (Unicode): "** + char\_var\_unicode + **"\n 2) Normal printing value of char variable (containing normal character):"** + char\_var\_alphabet + **"\n 3)Printing value of char variable (Unicode by incrementing it):"** + (char\_var\_unicode + 2) + **"\n"**);  
 System.out.println(**"Boolean is used to control flow of program.\n Boolean only have two values true and false"**);  
 **if** (boolean\_var == **true**) {  
 System.out.println(**"Here value of Boolean variable is: "** + boolean\_var + **"\n"**);  
 boolean\_var = **false**;  
 }  
 System.out.println(**"Here we have change the value of Boolean variable and have set it to "** + boolean\_var + **"\n"**);  
 System.out.println(**"Now take a look at type casting primitive data types\n"**);  
 System.out.println(**"There are two types of type casting:\n 1) Downcasting or Narrowing Type Casting \n 2) Widening Type Casting\n"**);  
 System.out.println(**"In Downcasting we convert a higher primitive datatype like int into lower datatype like short or byte primitive datatype or from double to int, etc.\n"** + **"In this we loose some information as the datatype to which we are converting to doesn't have that much of Space to store all the information.\n"**);  
 System.out.println(**"Example of Downcasting/ loosy Conversion:"**);  
 System.out.println(**"Converting double to int:\n"** + **"Double value: "**+double\_var+**"\nConverted int value: "**+((**int**) double\_var));  
 System.out.println(**"Converting int to byte:\n"**+**"Int value: "**+((**int**)double\_var)+**"\nConverted byte value: "**+((**byte**)((**int**) double\_var))+**"\n"**);  
 System.out.println(**"In Widening Type Casting we convert a lower primitive datatype like int into higher datatype like double or from datatype byte to int.\n"** + **"In this we dont loose any information as the datatype to which we are converting to much extra Space to store all the information.\n"**);  
 System.out.println(**"Example of Widening/ non-loosy Conversion:"**);  
 System.out.println(**"Converting int to double:\n"** + **"Int value: "**+int\_var+**"\nConverted Double value: "**+((**double**) int\_var));  
 System.out.println(**"Converting short to int:\n"**+**"Short Value: "**+short\_var+**"\nConverted Short Value: "**+((**int**)short\_var));  
 }  
}

**Text

Description automatically generatedOutput:**

**Text

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated with low confidence

**Practical 4**

**Aim:** Create the following patterns, (number of rows shall be entered by user at run time):

\* 1 \* 1

\*\* 1 2 \* \* \* 2 1 2

\*\*\* 1 2 3 \* \* \* \* \* 3 2 1 2 3

\*\*\*\* 1 2 3 4 \* \* \* 2 1 2

\*\*\*\*\* 1 2 3 4 5 \* 1

**Program:**

**import** java.util.Scanner;  
  
**public** **class** Patterns {  
 *// Function to print Star L shaped Pattern*  
 **public** **void** Pattern\_Star\_L(**int** r) {  
 **int** no\_of\_rows = r;  
  
 **for** (**int** i = 1; i <= no\_of\_rows; ++i) {  
 **for** (**int** j = 1; j <= i; ++j) {  
 System.out.print(**"\* "**);  
 }  
 System.out.println();  
 }  
 }  
  
 *// Function to print Number L shaped pattern*  
 **public** **void** Pattern\_Number\_L(**int** r) {  
 **int** no\_of\_rows = r;  
  
 **for** (**int** i = 1; i <= no\_of\_rows; ++i) {  
 **for** (**int** j = 1; j <= i; ++j) {  
 System.out.print(j + **" "**);  
 }  
 System.out.println();  
 }  
 }  
  
 *// Function to print Star Pyramid Pattern*  
 **public** **void** Pattern\_Star\_Pyramid(**int** r) {  
 **int** no\_of\_rows = r;  
 **int** space = no\_of\_rows - 1;  
 **for** (**int** j = 1; j <= no\_of\_rows; j++) {  
 **for** (**int** i = 1; i <= space; i++) {  
 System.out.print(**" "**);  
 }  
 space--;  
 **for** (**int** i = 1; i <= 2 \* j - 1; i++) {  
 System.out.print(**"\*"**);  
 }  
 System.out.println(**""**);  
 }  
 space = 1;  
 **for** (**int** j = 1; j <= no\_of\_rows - 1; j++) {  
 **for** (**int** i = 1; i <= space; i++) {  
 System.out.print(**" "**);  
 }  
 space++;  
 **for** (**int** i = 1; i <= 2 \* (no\_of\_rows - j) - 1; i++) {  
 System.out.print(**"\*"**);  
 }  
 System.out.println(**""**);  
 }  
 }  
  
 *// Function to print Number Pyramid Pattern*  
 **public** **void** Number\_Pyramid\_Pattern(**int** r) {  
 **int** no\_of\_rows = r;  
 **int** space = no\_of\_rows - 1;  
 **for** (**int** i = 1; i <= no\_of\_rows; i++) {  
 **int** n = 8;  
 **for** (**int** j = 1; j <= n - i; j++) {  
 System.out.print(**" "**);  
 }  
 **for** (**int** k = i; k >= 1; k--) {  
 System.out.print(k);  
 }  
 **for** (**int** l = 2; l <= i; l++) {  
 System.out.print(l);  
 }  
 System.out.println();  
 }  
 **for** (**int** i = space; i >= 1; i--) {  
 **int** n = 10;  
 **for** (**int** j = 0; j <= n - i; j++) {  
 System.out.print(**" "**);  
 }  
 **for** (**int** k = i; k >= 1; k--) {  
 System.out.print(k);  
 }  
 **for** (**int** l = 2; l <= i; l++) {  
 System.out.print(l);  
 }  
 System.out.println();  
 }  
 }  
  
 *// Main driver function*  
 **public** **static** **void** main(String[] args) {  
 Patterns patterns = **new** Patterns();  
 Scanner sc = **new** Scanner(System.in);  
  
 System.out.println(**"Enter no of rows for star L shaped pattern:"**);  
 **int** rows\_for\_star\_L = sc.nextInt();  
 patterns.Pattern\_Star\_L(rows\_for\_star\_L);  
  
 System.out.println(**"\nEnter no of rows for number L shaped pattern:"**);  
 **int** rows\_for\_number\_L = sc.nextInt();  
 patterns.Pattern\_Number\_L(rows\_for\_number\_L);  
  
 System.out.println(**"\nEnter no of rows for star shaped pyramid pattern:"**);  
 **int** rows\_for\_star\_Pyramid = sc.nextInt();  
 patterns.Pattern\_Star\_Pyramid(rows\_for\_star\_Pyramid);  
  
 System.out.println(**"\nEnter no of rows for number shaped pyramid pattern:"**);  
 **int** rows\_for\_Number\_Pyramid = sc.nextInt();  
 patterns.Number\_Pyramid\_Pattern(rows\_for\_Number\_Pyramid);  
 }  
}

**Text

Description automatically generatedOutput:**

**Text

Description automatically generated**

**Practical 5**

**Aim:** Create Even Series, odd series, Fibonacci Series, Prime Series, Sin series, Cosine series.

**Program:**

**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.Scanner;  
  
**public** **class** ManySeries {  
 *// function for generating even series*  
 **public** **void** evenSeries(**int** l) {  
 **int** limit = l;  
 List<Integer> series = **new** ArrayList<Integer>();  
 System.out.println(**"Even series upto "** + limit + **" is:"**);  
 **for** (**int** i = 0; i <= limit; i = i + 2) {  
 series.add(i);  
 }  
 System.out.println(series);  
 }  
  
 *// function for generating odds series*  
 **public** **void** oddSeries(**int** l) {  
 **int** limit = l;  
 List<Integer> series = **new** ArrayList<Integer>();  
 System.out.println(**"Odd series upto "** + limit + **" is:"**);  
 **for** (**int** i = 1; i <= limit; i = i + 2) {  
 series.add(i);  
 }  
 System.out.println(series);  
 }  
  
 *// function for generating fibonacciSeries*  
 **public** **void** fibonacciSeries(**int** l) {  
 **int** limit = l;  
 **int** firstTerm = 0;  
 **int** secondTerm = 1;  
 List<Integer> series = **new** ArrayList<Integer>();  
 System.out.println(**"Fibonacci series upto "** + limit + **" is:"**);  
 **for** (**int** i = 1; i <= limit; i++) {  
 series.add(firstTerm);  
 *// now computing the next terms*  
 **int** nextTerm = firstTerm + secondTerm;  
 firstTerm = secondTerm;  
 secondTerm = nextTerm;  
 }  
 System.out.println(series);  
 }  
  
 *// function for generating sine series*  
 **public** **void** sineSeries(**int** l, **int** sine\_term) {  
 **float** no\_of\_terms = l;  
 **float** radian = sine\_term;  
 **int** i, j, f, s = 1;  
 **float** sum = 0;  
 radian = (**float**) (radian \* 3.14159 / 180);  
 **for** (i = 1; i <= no\_of\_terms; i = i + 2) {  
 f = 1;  
 **for** (j = 1; j <= i; j++)  
 f = f \* j;  
 sum = (**float**) (sum + s \* (Math.pow(radian, i) / f));  
 s = s \* -1;  
 }  
 System.out.print(**"The value of sin("** + sine\_term + **") is "** + sum + **"\n"**);  
  
 }  
  
 **public** **void** cosineSeries(**int** l, **int** cosine\_term) {  
 **float** no\_of\_terms = l;  
 **float** radian = cosine\_term;  
 **int** i, j, f, s = 1;  
 **float** sum = 0;  
 radian = (**float**) (radian \* 3.14159 / 180);  
 **for** (i = 1; i <= no\_of\_terms; i = i + 2) {  
 f = 1;  
 **for** (j = 0; j <= i; j++)  
 f = f \* j;  
 sum = (**float**) (sum + s \* (Math.pow(radian, i) / f));  
 s = s \* -1;  
 }  
 System.out.print(**"The value of sin("** + cosine\_term + **") is "** + sum);  
 }  
  
 **public** **static** **void** main(String[] args) {  
 ManySeries manySeries = **new** ManySeries();  
 Scanner sc = **new** Scanner(System.in);  
  
 System.out.println(**"Enter the number upto which you want even series:"**);  
 **int** limit\_even = sc.nextInt();  
 manySeries.evenSeries(limit\_even);  
  
 System.out.println(**"\nEnter the number upto which you want odd series:"**);  
 **int** limit\_odd = sc.nextInt();  
 manySeries.oddSeries(limit\_odd);  
  
 System.out.println(**"\nEnter the number upto which you want Fibonacci series:"**);  
 **int** limit\_fibonacci = sc.nextInt();  
 manySeries.fibonacciSeries(limit\_fibonacci);  
  
 System.out.println(**"\nEnter the number of terms for which you want Sine Series:"**);  
 **int** limit\_sine\_series = sc.nextInt();  
  
 System.out.println(**"\nEnter the number value for which you want Sine series:"**);  
 **int** sine\_term = sc.nextInt();  
 manySeries.sineSeries(limit\_sine\_series, sine\_term);  
  
 System.out.println(**"\nEnter the number of terms for which you want Cosine Series:"**);  
 **int** limit\_cosine\_series = sc.nextInt();  
  
 System.out.println(**"\nEnter the number value for which you want Cosine series:"**);  
 **int** cosine\_term = sc.nextInt();  
 manySeries.sineSeries(limit\_sine\_series, cosine\_term);  
  
 }  
}

**Text

Description automatically generatedOutput:**

**Text

Description automatically generated**

**Practical 6**

**Aim:** Demonstrate various functions available in Maths class

**Program:**

**import** java.util.Scanner;  
  
  
**public** **class** MathsFuncDemo {  
   
 **public** **static** **void** main(String[] args){  
 Scanner sca1 = **new** Scanner(System.in);  
 System.out.println(**"Enter the first numerical value"**);  
   
 Double val1 = sca1.nextDouble();  
  
 Scanner sca2 = **new** Scanner(System.in);  
 System.out.println(**"Enter the second numerical value"**);  
   
 Double val2 = sca2.nextDouble();  
  
 System.out.println(**"\nFollowing are some of the functions provided by Math Class:"**);  
  
 System.out.println(**"Smallest of two entered numbers is: "**+ Math.min(val1, val2));  
  
 System.out.println(**"largest of two entered numbers is: "**+ Math.max(val1, val2));  
  
 System.out.println(**"\nCube root of "**+ val1 +**" number is: "** + Math.cbrt(val1));  
   
 System.out.println(**"Cube root of "**+ val2 +**" number is: "** + Math.cbrt(val2));  
  
 System.out.println(**"\nSquare root of "**+ val1 +**" number is: "** + Math.sqrt(val1));  
  
 System.out.println(**"Square root of "**+ val1 +**" number is: "** + Math.sqrt(val2));  
  
 System.out.println(**"\nlog of the "**+ val1+**" is: "**+ Math.log(val1));  
   
 System.out.println(**"log of the "**+ val2+**" is: "**+ Math.log(val2));  
  
 System.out.println(**"\nlog to value 10 of "**+ val1+**" is: "**+ Math.log10(val1));  
  
 System.out.println(**"log to value 10 of "**+ val2+**" is: "**+ Math.log10(val1));  
  
 System.out.println(**"\nNatural log of the sum of the argument and 1 of "**+ val1+**" is: "**+ Math.log1p(val1));  
  
 System.out.println(**"Natural log of the sum of the argument and 1 of "**+ val2+**" is: "**+ Math.log1p(val2));  
  
 System.out.println(**"\n"**+val1+**" Raised to the power of "**+val2+**" is: "**+ Math.pow(val1,val2));  
  
 System.out.println(**"\nExponential of "**+val1+**" is "**+ Math.getExponent(val1));  
  
 System.out.println(**"Exponential of "**+val2+**" is "**+ Math.getExponent(val2));  
  
 System.out.println(**"\nExp of "**+val1+**" is "**+ Math.exp(val1));  
  
 System.out.println(**"Expm1 of "**+val1+**" is "**+ Math.expm1(val1));  
  
 System.out.println(**"\nTrigonometric functions provided by Maths class:"**);  
  
 System.out.println(**"sin values of "**+val1+**" and "**+val2+**"are: "**+ Math.sin(val1)+**" and "**+Math.sin(val2)+**" respectively."**);  
  
 System.out.println(**"cosine values of "**+val1+**" and "**+val2+**" are: "**+ Math.cos(val1)+**" and "** +Math.cos(val2)+**" respectively."**);  
  
 System.out.println(**"tan values of "**+val1+**" and "**+val2+**" are: "**+ Math.tan(val1)+**" and "**+Math.tan(val2)+**" respectively."**);  
  
 System.out.println(**"\narc sin values of "**+val1+**" and "**+val2+**" are: "**+ Math.asin(val1)+**" and "**+Math.asin(val2)+**" respectively."**);  
  
 System.out.println(**"arc cosine values of "**+val1+**" and "**+val2+**" are: "**+ Math.acos(val1)+**" and "**+Math.acos(val2)+**" respectively."**);  
  
 System.out.println(**"arc tan values of "**+val1+**" and "**+val2+**" are: "**+ Math.atan(val1)+**" and "**+Math.atan(val2)+**" respectively."**);  
  
 System.out.println(**"\nsinh values of "**+val1+**" and "**+val2+**" are: "**+ Math.sinh(val1)+**" and "**+Math.sinh(val2)+**" respectively."**);  
  
 System.out.println(**"cosineh values of "**+val1+**" and "**+val2+**" are: "**+ Math.cosh(val1)+**" and "**+Math.cosh(val2)+**" respectively."**);  
  
 System.out.println(**"tanh values of "**+val1+**" and "**+val2+**" are: "**+ Math.tanh(val1)+**" and "**+Math.tanh(val2)+**" respectively."**);  
 }  
   
}

**Output:Text

Description automatically generated**

**Text

Description automatically generated**

**Practical 7**

**Aim:** Demonstrate the use of Biginteger and BigDecimal classes to perform operations on very large integer and decimals.

**Program:**

**import** java.math.\*;  
**import** java.util.Scanner;  
  
**public** **class** BigIntDecDemo {  
 **public** **static** **void** main(String[] args) {  
 System.out.println(**"Enter the first big integer:"**);  
 Scanner binte1 = **new** Scanner(System.in);  
 String s1 = binte1.nextLine();  
 s1 = s1.trim();  
  
 BigInteger Bi1 = **new** BigInteger(s1);  
  
 System.out.println(**"Enter the second big integer:"**);  
 Scanner binte2 = **new** Scanner(System.in);  
 String s2 = binte2.nextLine();  
 s2 = s2.trim();  
  
 BigInteger Bi2 = **new** BigInteger(s2);  
  
 System.out.println(**"\nMultiplication of entered numbers is "** + Bi2.multiply(Bi1));  
 System.out.println(**"Addition of entered numbers is "** + Bi2.add(Bi1));  
 *// big decimal demo*  
  
 System.out.println(**"\nEnter the first big decimal:"**);  
 Scanner bdec1 = **new** Scanner(System.in);  
 String s3 = bdec1.nextLine();  
 s3 = s3.trim();  
  
 BigDecimal Bd1 = **new** BigDecimal(s3);  
  
 System.out.println(**"Enter the second big decimal:"**);  
 Scanner bdec2 = **new** Scanner(System.in);  
 String s4 = bdec2.nextLine();  
 s4 = s4.trim();  
  
 BigDecimal Bd2 = **new** BigDecimal(s4);  
  
 System.out.println(**"\nMultiplication of entered numbers is "** + Bd2.multiply(Bd1));  
   
 System.out.println(**"Addition of entered numbers is "** + Bd2.add(Bd1));  
 }  
}

**Text

Description automatically generatedOutput:**

**Practical 8**

**Aim:** Demonstrate the creation and usage of static/non-static functions.

**Program:**

**public** **class** FunctionTypes {  
 **public** **static** **void** staticFunction() {  
 System.out.println(**"I am a static function\nAnd I dont need any object to create of my class in order to access it via another static function/method."**);  
 }  
  
 **public** **static** **void** staticAddFunction(**int** a, **int** b) {  
 System.out.println(**"Sum of numbers passed to this static add function is: "** + (a + b));  
 }  
  
 **public** **void** nonStaticFunction() {  
 System.out.println(**"I am a static function\nAnd I need an object to create of my class in order to access it via another static function/method."**);  
 }  
  
 **public** **static** **void** main(String[] args) {  
 staticFunction();  
 staticAddFunction(17, 10);  
 FunctionTypes functionTypes = **new** FunctionTypes();  
 functionTypes.nonStaticFunction();  
 }  
}

A picture containing graphical user interface

Description automatically generated**Output:**

**Practical 9**

**Aim:** Demonstrate the use of Getters and Setters functions.

**Program:** Here in the program, we are using getter and setters to add records of the students. We have two files one is ArrayOfObjects.java and Students.java. in Students,java we have setters and getters which we are using to save and retrieve values respectively.

**Students.java :**

**public** **class** Students{  
String Name;  
**int** Rollno;  
**int** Classno;  
  
 **public** **void** setName(String Name) {  
 **this**.Name=Name;  
 }  
  
  
 **public** **void** setRollno(**int** Rollno) {  
 **this**.Rollno=Rollno;  
 }  
  
 **public** **void** setClassno(**int** Classno) {  
 **this**.Classno= Classno;  
 }  
  
 **public** String getName() {  
 **return** **this**.Name;  
 }  
  
 **public** **int** getRollno() {  
 **return** **this**.Rollno;  
 }  
  
 **public** **int** getClassno(){  
 **return** **this**.Classno;  
 }  
}

**ArrayOfObjects.java :**

**import** java.util.Scanner;  
  
**public** **class** ArrayOfObjects {  
 **public** **static** **void** main(String[] args) {  
 Students[] StudArray; *// Here we have Declared array of references of type Students.*  
 *// Here that StudArray contains the references that pont to objects of type students.*  
  
 System.out.println(**"Enter how many numer of students Record do you want to enter:"**);  
  
 Scanner srcn = **new** Scanner(System.in);  
 **int** sno = srcn.nextInt();  
  
 StudArray = **new** Students[sno]; *// here we are assigning size/memory to our array of references of type students.*  
  
 System.out.println(**"Enter the record of "** + sno + **" students one by one:\n"**);  
  
 Scanner Sname = **new** Scanner(System.in);  
 Scanner Sroll = **new** Scanner(System.in);  
 Scanner Sclass = **new** Scanner(System.in);  
  
 **for** (**int** i = 0; i < sno; i++) {  
 StudArray[i] = **new** Students(); *// Here actual object of type Students gets created*  
 *// and the reference variable StudArray[1] in cae of first iteration of for loop points to that students object.*  
 System.out.println(**"\nEnter the name of "** + (i + 1) + **" student: "**);  
 String NameOfStudent = Sname.nextLine();  
 StudArray[i].setName(NameOfStudent); *//Here we have used Setter function of Students object which is .setName()*  
 *// to set the name of student.*  
  
 System.out.println(**"Enter the Roll Number of "** + (i + 1) + **" student: "**);  
 **int** RollnoOfStudent = Sroll.nextInt();  
 StudArray[i].setRollno(RollnoOfStudent);*//Here we have used Setter function of Students object which is.setRollno()*  
 *// to set the name of student.*  
  
 System.out.println(**"Enter the Class of "** + (i + 1) + **" student: "**);  
 **int** ClassOfStudent = Sclass.nextInt();  
 StudArray[i].setClassno(ClassOfStudent);*//Here we have used Setter function of Students object which is.setClassno()*  
 *// to set the name of student.*  
 }  
  
 System.out.println(**"Records Entered by You are: \n"**);  
  
 **int** i = 1;  
 **for** (Students S : StudArray) {  
 System.out.println(**"Name of "** + i + **" Student: "** + S.getName());  
 System.out.println(**"Roll no of "** + i + **" Student: "** + S.getRollno());  
 System.out.println(**"Class of "** + i + **" Student: "** + S.getClassno());  
 i += 1;  
  
 */\*Here above we have used three Getter functions of Students object which is .getName(), .getRollno(), .getClassno()  
 to get the name, roll number, Class of the students. \*/*  
  
 }  
  
 }  
}

**Text

Description automatically generatedOutput:**

**Practical 10**

**Aim:** Demonstrate various kinds of comments available in java including documentation comments.

**Program:**

**public** **class** JavaComments {  
 *// This is just a basic comment in the program*  
 *// By using these // we can only put comment on one line*  
  
 */\* This is a multi line comment in java  
 we use these /\* to put comments in multiple lines.\*/*  
  
 */\*\*  
 \* This is a documentation comment in the program  
 \* By using this we can place multiline documentation comments in our java program.  
 \*/*  
  
 **public** **static** **void** main(String[] args) {  
 System.out.println(**"This program shows multiple comments available in java like\n1) Normal comments via // with which we can only put single line comment.\n2) Multiple line comments via /\* Multi line comment\*/\n3) Documentation comments in java via /\*\* Some Documentation \*/ "**);  
 }  
  
}

**Text

Description automatically generatedOutput:**

**Practical 11**

**Aim:** Demonstrate the use of enumerate data type

**Program:**

**import** java.util.Scanner;  
  
*// enum of type Planet*  
**enum** Planet {  
 MERCURY(3.303e+23, 2.4397e6),  
 VENUS(4.869e+24, 6.0518e6),  
 EARTH(5.976e+24, 6.37814e6),  
 MARS(6.421e+23, 3.3972e6),  
 JUPITER(1.9e+27, 7.1492e7),  
 SATURN(5.688e+26, 6.0268e7),  
 URANUS(8.686e+25, 2.5559e7),  
 NEPTUNE(1.024e+26, 2.4746e7),  
 PLUTO(1.27e+22, 1.137e6);  
  
 *// mass is in Kg*  
 **private** **final** **double** mass;  
  
 *//radius of planet is in meter*  
 **private** **final** **double** radius;  
  
 */\*This below constructor is the private  
 constructor of enum Planet here\*/*  
 Planet(**double** mass, **double** radius) {  
 **this**.mass = mass;  
 **this**.radius = radius;  
 }  
  
 */\*These below functions return the mass  
 and radius of every Planet in emun Planet\*/*  
 **public** **double** mass() {  
 **return** mass;  
 }  
  
 **public** **double** radius() {  
 **return** radius;  
 }  
  
 *//universal gravitational constant (m3 kg-1 s-2)*  
 **public** **static** **final** **double** G = 6.67300E-11;  
  
 **public** **double** surfaceGravity() {  
 **return** G \* mass / (radius \* radius);  
 }  
  
 **public** **double** surfaceWeight(**double** otherMass) {  
 **return** otherMass \* surfaceGravity();  
 }  
}  
  
*// Main driver program*  
**public** **class** EnumeratedProgram {  
 **public** **static** **void** main(String[] args) {  
 System.out.println(**"Enter you weight on here on Earth:"**);  
 Scanner sc = **new** Scanner(System.in);  
 **double** weight\_on\_earth = sc.nextDouble();  
  
 **double** mass = weight\_on\_earth / Planet.EARTH.surfaceGravity();  
  
 **for** (Planet p : Planet.values()) {  
 System.out.printf(**"Your weight on "** + p + **" is "** + p.surfaceWeight(mass) + **"\n"**);  
 }  
 }  
}

Text

Description automatically generated**Output:**

**Practical 12**

**Aim:** Demonstrate the use of various arithmetic, logical, bitwise, unary, ternary, assignment operators and understand their precedence and hierarchy(L to R, R to L).

**Program:**

**import** java.util.Scanner;  
  
**public** **class** OperatorsProgram {  
 **public** **void** urinaryOperator(**double** n) {  
 **int** a = 6, b = -6;  
 **boolean** bool = **true**;  
 System.out.println(**"Incrementing Entered value by one(+1) by using urinary operator n++ (postfix ++): "** + n++);  
 System.out.println(**"Decrementing Entered value by one(-1) by using urinary operator n-- (postfix --): "** + n--);  
 System.out.println(**"Incrementing Entered value by one(+1) by using urinary operator ++n (prefix ++): "** + ++n);  
 System.out.println(**"Decrementing Entered value by one(-1) by using urinary operator --n (prefix --): "** + --n);  
  
 System.out.println(**"\nApplying ~ operator on "** + a + **" gives value: "** + ~a);  
 System.out.println(**"Applying ~ operator on "** + b + **" gives value: "** + ~b);  
  
 System.out.println(**"\nApplying ! operator on boolean variable bool having value \""** + bool + **"\" gives value \""** + !bool + **"\""**);  
 }  
  
 **public** **void** airthmeticOperator(**double** a, **double** b) {  
 System.out.println(**"Applying + Airthmetic operator on "** + a + **" and "** + b + **" gives us value:"** + (a + b));  
 System.out.println(**"Applying - Airthmetic operator on "** + a + **" and "** + b + **" gives us value:"** + (a - b));  
 System.out.println(**"Applying \* Airthmetic operator on "** + a + **" and "** + b + **" gives us value:"** + (a \* b));  
 System.out.println(**"Applying / Airthmetic operator on "** + a + **" and "** + b + **" gives us value:"** + (a / b));  
 System.out.println(**"Applying % Airthmetic operator on "** + a + **" and "** + b + **" gives us value:"** + (a % b));  
 }  
  
 **public** **void** logicalAndBitwiseOperator(**double** a, **double** b) {  
 System.out.println(**"Applying logical operator && on a and b for(a>5&&b>=5)"** + (a > 5 && b >= 5));  
 System.out.println(**"Applying logical operator & on a and b for(a>5&b>=5)"** + (a > 5 & b >= 5));  
  
 System.out.println(**"\nApplying logical operator || on a and b for(a>5||b>=5)"** + (a > 5 || b >= 5));  
 System.out.println(**"Applying logical operator | on a and b for(a>5|b>=5)"** + (a > 5 | b >= 5));  
 }  
  
 **public** **void** ternaryOperator(**double** a, **double** b) {  
 System.out.println(**"Applying ternary operator ? condition on a and b for(a>b?a:b) "** + **"\nThen max of a and b is "** + (a > b ? a : b));  
 System.out.println(**"Applying ternary operator ? condition on a and b for(a<b?a:b) "** + **"\nThen min of a and b is "** + (a < b ? a : b));  
 }  
  
 **public** **void** assignmnetOperator(**double** a, **double** b) {  
 System.out.println(**"\nApplying assignment operator on a via(a+=7) "** + (a += 7));  
 System.out.println(**"Applying assignment operator on a via(b-=5) "** + (b -= 5));  
  
 }  
  
 **public** **static** **void** main(String[] args) {  
 System.out.println(**"Enter any number on which you want to apply Urinary operators:"**);  
 Scanner sc = **new** Scanner(System.in);  
 **double** num = sc.nextDouble();  
 OperatorsProgram operatorsProgram = **new** OperatorsProgram();  
 operatorsProgram.urinaryOperator(num);  
  
 System.out.println(**"\nEnter any number on which you want to apply Arithmetic,logical, bitwise, ternary and assignment operators:"**);  
 System.out.println(**"Enter first number:"**);  
 **double** a = sc.nextDouble();  
 System.out.println(**"Enter second number:"**);  
 **double** b = sc.nextDouble();  
 operatorsProgram.airthmeticOperator(a, b);  
 operatorsProgram.logicalAndBitwiseOperator(a, b);  
 operatorsProgram.ternaryOperator(a, b);  
 operatorsProgram.assignmnetOperator(a, b);  
 }  
}

Text

Description automatically generatedText

Description automatically generated**Output:**

**Practical 13**

**Aim:** Demonstrate the use 1D, 2D array and various functions available in Array class.

**Program:**

**import** java.util.Arrays;  
**import** java.util.Scanner;  
  
**class** ArrayDimension {  
 **int**[] arryi;  
 **int**[][] arrys;  
  
 **void** OneDArrayMake() {  
  
  
 System.out.println(**"How many numbers you want to enter in 1D Array :"**);  
 Scanner inp = **new** Scanner(System.in);  
  
 **int** scai = inp.nextInt();  
 arryi = **new** **int**[scai];  
  
 System.out.println(**"Enter "** + scai + **" values one by one and press enter after each entry:"**);  
 **for** (**int** i = 0; i < scai; i++) {  
 Scanner inparryi = **new** Scanner(System.in);  
 arryi[i] = inparryi.nextInt();  
 }  
  
 System.out.println(**"Values you Entered:"**);  
 **int** i = 0;  
 **for** (**int** b : arryi) {  
 System.out.println(**"Value enterd at arryi["** + i + **"] is: "** + b);  
 i++;  
 }  
  
 }  
  
 **void** OneDArrFunction() {  
 System.out.println(**"\nLength of the entered one Dimension array is: "** + arryi.length);  
 System.out.println(**"Maximum value entered in one Dimension array is: "** + Arrays.stream(arryi).max());  
 System.out.println(**"Minimum value entered in one Dimension array is: "** + Arrays.stream(arryi).min());  
 System.out.println(**"Hash code of 1D Array is : "** + Arrays.hashCode(arryi) + **"\n"**);  
 }  
  
  
 **void** TwoDArrayMake() {  
  
  
 System.out.println(**"\nHow many columns you want to have in your 2D array:"**);  
 Scanner inpc = **new** Scanner(System.in);  
 **int** scaicol = inpc.nextInt();  
  
 System.out.println(**"Enter how many rows you want to have in your 2D array:"**);  
 Scanner inpr = **new** Scanner(System.in);  
 **int** scairow = inpr.nextInt();  
  
 arrys = **new** **int**[scairow][scaicol];  
  
 System.out.println(**"Enter "** + scaicol \* scairow + **" values one by one and press enter after each entry:"**);  
 **for** (**int** r = 0; r < scairow; r++) {  
 **for** (**int** c = 0; c < scaicol; c++) {  
 System.out.println(**"Enter the value you want to put at ("** + r + **','** + c + **") :"**);  
 Scanner inparrys = **new** Scanner(System.in);  
 arrys[r][c] = inparrys.nextInt();  
 }  
 }  
  
 System.out.println(**"\nValues you Entered in 2D array is: "**);  
 **int** i = 0;  
 **int** j = 0;  
 **for** (**int**[] r : arrys) {  
 **for** (**int** c : r) {  
 System.out.println(**"Entered value at ("** + i + **","** + j + **") is: "** + c);  
 j++;  
 }  
 i++;  
 j = 0;  
 }  
 }  
  
 **void** TwoDArrFunction() {  
 System.out.println(**"\nLength of the entered 2D array is: "** + arrys.length);  
  
 System.out.println(**"Hash code of entered 2D Array is :"** + Arrays.hashCode(arrys));  
 }  
  
}  
  
  
**public** **class** ArrayDemo {  
 **public** **static** **void** main(String[] args) {  
 ArrayDimension AOD = **new** ArrayDimension();  
 AOD.OneDArrayMake();  
 AOD.OneDArrFunction();  
 AOD.TwoDArrayMake();  
 AOD.TwoDArrFunction();  
 }  
  
}

}

Text

Description automatically generated**Output:**

Text

Description automatically generatedText

Description automatically generated

**Practical 14**

**Aim:** Demonstrate the concept of Array of objects, create Students class. Ask user at run time for the no. of students to be created. Create Students objects, initialize their values using setter functions. Display all the student’s details using getters functions.

**Program:** Here in the program, we are using getter and setters to add records of the students. We have two files one is ArrayOfObjects.java and Students.java. in Students,java we have setters and getters which we are using to save and retrieve values respectively.

**Students.java :**

**public** **class** Students{  
String Name;  
**int** Rollno;  
**int** Classno;  
  
 **public** **void** setName(String Name) {  
 **this**.Name=Name;  
 }  
  
  
 **public** **void** setRollno(**int** Rollno) {  
 **this**.Rollno=Rollno;  
 }  
  
 **public** **void** setClassno(**int** Classno) {  
 **this**.Classno= Classno;  
 }  
  
 **public** String getName() {  
 **return** **this**.Name;  
 }  
  
 **public** **int** getRollno() {  
 **return** **this**.Rollno;  
 }  
  
 **public** **int** getClassno(){  
 **return** **this**.Classno;  
 }  
}

**ArrayOfObjects.java :**

**import** java.util.Scanner;  
  
**public** **class** ArrayOfObjects {  
 **public** **static** **void** main(String[] args) {  
 Students[] StudArray; *// Here we have Declared array of references of type Students.*  
 *// Here that StudArray contains the references that pont to objects of type students.*  
  
 System.out.println(**"Enter how many numer of students Record do you want to enter:"**);  
  
 Scanner srcn = **new** Scanner(System.in);  
 **int** sno = srcn.nextInt();  
  
 StudArray = **new** Students[sno]; *// here we are assigning size/memory to our array of references of type students.*  
  
 System.out.println(**"Enter the record of "** + sno + **" students one by one:\n"**);  
  
 Scanner Sname = **new** Scanner(System.in);  
 Scanner Sroll = **new** Scanner(System.in);  
 Scanner Sclass = **new** Scanner(System.in);  
  
 **for** (**int** i = 0; i < sno; i++) {  
 StudArray[i] = **new** Students(); *// Here actual object of type Students gets created*  
 *// and the reference variable StudArray[1] in cae of first iteration of for loop points to that students object.*  
 System.out.println(**"\nEnter the name of "** + (i + 1) + **" student: "**);  
 String NameOfStudent = Sname.nextLine();  
 StudArray[i].setName(NameOfStudent); *//Here we have used Setter function of Students object which is .setName()*  
 *// to set the name of student.*  
  
 System.out.println(**"Enter the Roll Number of "** + (i + 1) + **" student: "**);  
 **int** RollnoOfStudent = Sroll.nextInt();  
 StudArray[i].setRollno(RollnoOfStudent);*//Here we have used Setter function of Students object which is.setRollno()*  
 *// to set the name of student.*  
  
 System.out.println(**"Enter the Class of "** + (i + 1) + **" student: "**);  
 **int** ClassOfStudent = Sclass.nextInt();  
 StudArray[i].setClassno(ClassOfStudent);*//Here we have used Setter function of Students object which is.setClassno()*  
 *// to set the name of student.*  
 }  
  
 System.out.println(**"Records Entered by You are: \n"**);  
  
 **int** i = 1;  
 **for** (Students S : StudArray) {  
 System.out.println(**"Name of "** + i + **" Student: "** + S.getName());  
 System.out.println(**"Roll no of "** + i + **" Student: "** + S.getRollno());  
 System.out.println(**"Class of "** + i + **" Student: "** + S.getClassno());  
 i += 1;  
  
 */\*Here above we have used three Getter functions of Students object which is .getName(), .getRollno(), .getClassno()  
 to get the name, roll number, Class of the students. \*/*  
  
 }  
  
 }  
}

**Text

Description automatically generatedOutput:**

**Practical 15**

**Aim:** WAP to create 100 random numbers( range 1 to 100) and store then in array, sort the array and display the sorted array using enhanced for loop.

**Program:**

**import** java.util.Random;  
  
**public** **class** RandomNumbers {  
 **public** **static** **void** main(String[] args) {  
 *//Generating random numbers*  
 Random random = **new** Random();  
 **int** starting = 1;  
 **int** ending = 101;  
 **int**[] randomArray = **new** **int**[100];  
 **for** (**int** i = 0; i < 100; i++) {  
 randomArray[i] = random.nextInt(ending - starting) + starting;  
 }  
 System.out.println(**"\nArray before sorting is:"**);  
 **for** (**int** randomNumber : randomArray) {  
 System.out.println(randomNumber + **", "**);  
 }  
  
 Arrays.sort(randomArray);  
  
 System.out.println(**"\nArray after sorting is:"**);  
 **for** (**int** sortedNumber : randomArray) {  
 System.out.println(sortedNumber + **", "**);  
 }  
 }  
}

Text

Description automatically generated**Output:**

Text

Description automatically generated with low confidence

A picture containing text, electronics, screenshot, display

Description automatically generated Text

Description automatically generated with low confidence

A picture containing text, electronics, screenshot, computer

Description automatically generated

Text

Description automatically generated with low confidence

Text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

A computer screen capture

Description automatically generated with low confidence

**Practical 16**

**Aim:** Demonstrate the Matrix addition, subtraction, multiplication using 2D arrays

**Program:**

**import** java.util.Scanner;  
  
**public** **class** MatrixMultiplication {  
 **int** scaicol;  
 **int** scairow;  
 **int**[][] one\_array;  
 **int**[][] another\_array;  
  
 **void** matrixMake() {  
 System.out.println(**"\nHow many columns you want to have in your both 2D array/matrix:"**);  
 Scanner inpc = **new** Scanner(System.in);  
 scaicol = inpc.nextInt();  
  
 System.out.println(**"Enter how many rows you want to have in your 2D array:"**);  
 Scanner inpr = **new** Scanner(System.in);  
 scairow = inpr.nextInt();  
  
 one\_array = **new** **int**[scairow][scaicol];  
 another\_array = **new** **int**[scairow][scaicol];  
  
 System.out.println(**"\nEnter values for (1st) 2D array/matrix"**);  
 System.out.println(**"Enter "** + scaicol \* scairow + **" values one by one and press enter after each entry:"**);  
 **for** (**int** r = 0; r < scairow; r++) {  
 **for** (**int** c = 0; c < scaicol; c++) {  
 System.out.println(**"Enter the value you want to put at ("** + r + **','** + c + **") :"**);  
 Scanner inparrys = **new** Scanner(System.in);  
 one\_array[r][c] = inparrys.nextInt();  
 }  
 }  
 System.out.println(**"Entered (1st) array is:"**);  
 **for** (**int** i = 0; i < 3; i++) {  
 **for** (**int** j = 0; j < 3; j++) {  
 System.out.print(one\_array[i][j] + **" "**);  
 }  
 System.out.println();  
 }  
  
 System.out.println(**"\nEnter values for (2nd) 2D array/matrix"**);  
 System.out.println(**"Enter "** + scaicol \* scairow + **" values one by one and press enter after each entry:"**);  
 **for** (**int** r = 0; r < scairow; r++) {  
 **for** (**int** c = 0; c < scaicol; c++) {  
 System.out.println(**"Enter the value you want to put at ("** + r + **','** + c + **") :"**);  
 Scanner inparrys = **new** Scanner(System.in);  
 another\_array[r][c] = inparrys.nextInt();  
 }  
 }  
 System.out.println(**"Entered (2nd) array is:"**);  
 **for** (**int** i = 0; i < 3; i++) {  
 **for** (**int** j = 0; j < 3; j++) {  
 System.out.print(another\_array[i][j] + **" "**);  
 }  
 System.out.println();  
 }  
  
 }  
  
 *// This function specifies how to multiply two matrix*  
 **public** **void** multiply() {  
 **int**[][] temp\_array;  
 temp\_array = **new** **int**[scairow][scaicol];  
 System.out.println(**"Multiplication of two entered matrix gives us following matrix:"**);  
 **for** (**int** i = 0; i < scairow; i++) {  
 **for** (**int** j = 0; j < scaicol; j++) {  
 temp\_array[i][j] = 0;  
 **for** (**int** k = 0; k < 3; k++) {  
 temp\_array[i][j] += one\_array[i][k] \* another\_array[k][j];  
 }  
 System.out.print(temp\_array[i][j] + **" "**);  
 }  
 System.out.println();  
 }  
 }  
  
 *// This function specifies how to do addition of matrix*  
 **public** **void** addition() {  
 **int**[][] temp\_array;  
 temp\_array = **new** **int**[scairow][scaicol];  
 System.out.println(**"\nAddition of two entered matrix gives us following matrix:"**);  
 **for** (**int** i = 0; i < scairow; i++) {  
 **for** (**int** j = 0; j < scaicol; j++) {  
 temp\_array[i][j] = 0;  
 **for** (**int** k = 0; k < 3; k++) {  
 temp\_array[i][j] += one\_array[i][k] + another\_array[k][j];  
 }  
 System.out.print(temp\_array[i][j] + **" "**);  
 }  
 System.out.println();  
 }  
 }  
  
 *// Main driver function of the program*  
 **public** **static** **void** main(String[] args) {  
 MatrixMultiplication matrixMultiplication = **new** MatrixMultiplication();  
 matrixMultiplication.matrixMake();  
 matrixMultiplication.multiply();  
 matrixMultiplication.addition();  
  
 }  
}

Text

Description automatically generated**Output:**

Text

Description automatically generated Text

Description automatically generated

**Practical 17**

**Aim:** WAP to solve MagicBoard problem(CodeChef) using 2D arrays.

**Program:**

**import** java.util.\*;  
**import** java.io.\*;  
**import** java.math.BigInteger;  
**import** java.text.DecimalFormat;  
  
**public** **class** MagicBoard {  
 **final** **long** mod = (**int**) 1e9 + 7, IINF = (**long**) 1e19;  
 **final** **int** MAX = (**int**) 1e6 + 1, MX = (**int**) 1e7 + 1, INF = (**int**) 1e9, root = 3;  
 DecimalFormat df = **new** DecimalFormat(**"0.0000000000000"**);  
 **double** eps = 1e-9, PI = 3.141592653589793238462643383279502884197169399375105820974944;  
 **static** **boolean** multipleTC = **true**, memory = **false**;  
 FastReader in;  
 PrintWriter out;  
  
 **public** **static** **void** main(String[] args) **throws** Exception {  
 **if** (memory) **new** Thread(**null**, **new** Runnable() {  
 **public** **void** run() {  
 **try** {  
 **new** MagicBoard().run();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }, **"1"**, 1 << 28).start();  
 **else** **new** MagicBoard().run();  
 }  
  
 **void** run() **throws** Exception {  
 System.out.println(**"Enter the input values one by one and press enter after every input:"**);  
 in = **new** FastReader();  
 out = **new** PrintWriter(System.out);  
 **for** (**int** i = 1, T = (multipleTC) ? ni() : 1; i <= T; i++) solve(i);  
 out.flush();  
 out.close();  
 }  
  
 **void** solve(**int** TC) **throws** Exception {  
 **int** n = ni(), m = ni();  
 Queue<Integer>[] q = **new** Queue[MAX];  
 **for** (**int** i = 1; i < MAX; i++) q[i] = **new** LinkedList<>();  
 **int**[][] g = **new** **int**[n][m];  
 **for** (**int** i = 0; i < n; i++) {  
 String s = n();  
 **for** (**int** j = 0; j < m; j++) {  
 g[i][j] = s.charAt(j) - **'0'**;  
 **if** (g[i][j] > 0) q[g[i][j]].add(i \* m + j);  
 }  
 }  
 **for** (**int** i = 1; i < 10; i++) {  
 **if** (q[i].isEmpty()) {  
 pn(i);  
 **return**;  
 }  
 }  
 **for** (**int** num = 10; num < MAX; num++) {  
 **int** x = num / 10;  
 **while** (!q[x].isEmpty()) {  
 **int** p = q[x].poll();  
 **int** i = p / m, j = p % m;  
 **if** (i < n - 1) q[x \* 10 + g[i + 1][j]].add((i + 1) \* m + j);  
 **if** (i > 0) q[x \* 10 + g[i - 1][j]].add((i - 1) \* m + j);  
 **if** (j > 0) q[x \* 10 + g[i][j - 1]].add(i \* m + j - 1);  
 **if** (j < m - 1) q[x \* 10 + g[i][j + 1]].add(i \* m + j + 1);  
 }  
 **if** (q[num].isEmpty()) {  
 pn(num);  
 **return**;  
 }  
 }  
 }  
  
 **void** push(Queue<Integer>[] q, **int**[][] g, **int** x) {  
 **int** n = g.length, m = g[0].length;  
  
 }  
  
 **int** lis(**long**[] a) {  
 TreeMap<Long, Integer> map = **new** TreeMap<>();  
 **int** ans = 0;  
 **for** (**int** i = 0; i < a.length; i++) {  
 **int** l = 0;  
 **if** (map.isEmpty() || map.floorEntry(a[i]) == **null**) l = 1;  
 **else** l = map.floorEntry(a[i]).getValue() + 1;  
 Map.Entry<Long, Integer> e = **null**;  
 **while** ((e = map.ceilingEntry(a[i])) != **null** && Integer.compare(e.getValue(), l) <= 0)  
 map.remove(e.getKey());  
 map.put(a[i], l);  
 ans = Math.max(ans, l);  
 }  
 **return** ans;  
 }  
  
 **int**[] sort(**int**[] a) {  
 **if** (a.length == 1) **return** a;  
 **int** mid = a.length / 2;  
 **int**[] b = sort(Arrays.copyOfRange(a, 0, mid)), c = sort(Arrays.copyOfRange(a, mid, a.length));  
 **for** (**int** i = 0, j = 0, k = 0; i < a.length; i++) {  
 **if** (j < b.length && k < c.length) {  
 **if** (b[j] < c[k]) a[i] = b[j++];  
 **else** a[i] = c[k++];  
 } **else** **if** (j < b.length) a[i] = b[j++];  
 **else** a[i] = c[k++];  
 }  
 **return** a;  
 }  
  
 **long**[] sort(**long**[] a) {  
 **if** (a.length == 1) **return** a;  
 **int** mid = a.length / 2;  
 **long**[] b = sort(Arrays.copyOfRange(a, 0, mid)), c = sort(Arrays.copyOfRange(a, mid, a.length));  
 **for** (**int** i = 0, j = 0, k = 0; i < a.length; i++) {  
 **if** (j < b.length && k < c.length) {  
 **if** (b[j] < c[k]) a[i] = b[j++];  
 **else** a[i] = c[k++];  
 } **else** **if** (j < b.length) a[i] = b[j++];  
 **else** a[i] = c[k++];  
 }  
 **return** a;  
 }  
  
 **long** gcd(**long** a, **long** b) {  
 **return** (b == 0) ? a : gcd(b, a % b);  
 }  
  
 **int** gcd(**int** a, **int** b) {  
 **return** (b == 0) ? a : gcd(b, a % b);  
 }  
  
 **int** bitcount(**long** n) {  
 **return** (n == 0) ? 0 : (1 + bitcount(n & (n - 1)));  
 }  
  
 **void** p(Object o) {  
 out.print(o);  
 }  
  
 **void** pn(Object o) {  
 out.println(o);  
 }  
  
 **void** pni(Object o) {  
 out.println(o);  
 out.flush();  
 }  
  
 String n() {  
 **return** in.next();  
 }  
  
 String nln() {  
 **return** in.nextLine();  
 }  
  
 **int** ni() {  
 **return** Integer.parseInt(in.next());  
 }  
  
 **long** nl() {  
 **return** Long.parseLong(in.next());  
 }  
  
 **double** nd() {  
 **return** Double.parseDouble(in.next());  
 }  
  
 **class** FastReader {  
 BufferedReader br;  
 StringTokenizer st;  
  
 **public** FastReader() {  
 br = **new** BufferedReader(**new** InputStreamReader(System.in));  
 }  
  
 **public** FastReader(String s) **throws** Exception {  
 br = **new** BufferedReader(**new** FileReader(s));  
 }  
  
 String next() {  
 **while** (st == **null** || !st.hasMoreElements()) {  
 **try** {  
 st = **new** StringTokenizer(br.readLine());  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **return** st.nextToken();  
 }  
  
 String nextLine() {  
 String str = **""**;  
 **try** {  
 str = br.readLine();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **return** str;  
 }  
 }  
}

Text

Description automatically generated**Output:**

**Practical 18**

**Aim:** WAP to create Tic-Tac-Toe game (2 players version)

**Program:**

**import** java.util.Scanner;  
  
**public** **class** TicTacToeVsPlayer {  
 **private** **char**[][] board;  
 **private** **char** currentPlayer;  
  
 **public** **static** **void** main(String[] args) {  
 Scanner scan = **new** Scanner(System.in);  
  
 TicTacToeVsPlayer ticTacToeVsPlayer = **new** TicTacToeVsPlayer();  
 ticTacToeVsPlayer.board = **new** **char**[3][3];  
 ticTacToeVsPlayer.currentPlayer = **'x'**;  
 ticTacToeVsPlayer.initializeBoard();  
 System.out.println(**"Tic-Tac-Toe!"**);  
 **do** {  
 System.out.println(**"Current board layout:"**);  
 ticTacToeVsPlayer.printBoard();  
 **int** row;  
 **int** col;  
 **do** {  
 System.out.println(**"Player "** + ticTacToeVsPlayer.getCurrentPlayer() + **", enter an empty row and column to make your move"**);  
 row = scan.nextInt() - 1;  
 col = scan.nextInt() - 1;  
 }  
 **while** (!ticTacToeVsPlayer.placeMark(row, col));  
 ticTacToeVsPlayer.changePlayer();  
 }  
 **while** (!ticTacToeVsPlayer.isWinner() && !ticTacToeVsPlayer.isBoardFull());  
 **if** (ticTacToeVsPlayer.isBoardFull() && !ticTacToeVsPlayer.isWinner()) {  
 System.out.println(**"The game was a tie!"**);  
 } **else** {  
 System.out.println(**"Current board layout:"**);  
 ticTacToeVsPlayer.printBoard();  
 ticTacToeVsPlayer.changePlayer();  
 System.out.println(Character.toUpperCase(ticTacToeVsPlayer.getCurrentPlayer()) + **" Wins!"**);  
 }  
  
 }  
  
 *//Gives us access to currentPlayerMark*  
 **public** **char** getCurrentPlayer() {  
 **return** currentPlayer;  
 }  
   
 *// Set/Reset the board back to all empty values.*  
 **public** **void** initializeBoard() {  
  
 *// Loop through rows*  
 **for** (**int** i = 0; i < 3; i++) {  
  
 *// Loop through columns*  
 **for** (**int** j = 0; j < 3; j++) {  
 board[i][j] = **'-'**;  
 }  
 }  
 }  
  
 *// Print the current board (may be replaced by GUI implementation later)*  
 **public** **void** printBoard() {  
 System.out.println(**"-------------"**);  
  
 **for** (**int** i = 0; i < 3; i++) {  
 System.out.print(**"| "**);  
 **for** (**int** j = 0; j < 3; j++) {  
 System.out.print(board[i][j] + **" | "**);  
 }  
 System.out.println();  
 System.out.println(**"-------------"**);  
 }  
 }  
  
 *// Loop through all cells of the board and if one is found to be empty (contains char '-') then return false.*  
 *// Otherwise the board is full.*  
 **public** **boolean** isBoardFull() {  
 **boolean** isFull = **true**;  
  
 **for** (**int** i = 0; i < 3; i++) {  
 **for** (**int** j = 0; j < 3; j++) {  
 **if** (board[i][j] == **'-'**) {  
 isFull = **false**;  
 }  
 }  
 }  
  
 **return** isFull;  
 }  
  
 *// Returns true if there is a win, false otherwise.*  
 *// This calls our other win check functions to check the entire board.*  
 **public** **boolean** isWinner() {  
 **return** (checkRows() || checkColumns() || checkDiagonals());  
 }  
  
 *// Loop through rows and see if any are winners.*  
 **private** **boolean** checkRows() {  
 **for** (**int** i = 0; i < 3; i++) {  
 **if** (checkRowCol(board[i][0], board[i][1], board[i][2]) == **true**) {  
 **return** **true**;  
 }  
 }  
 **return** **false**;  
 }  
  
  
 *// Loop through columns and see if any are winners.*  
 **private** **boolean** checkColumns() {  
 **for** (**int** i = 0; i < 3; i++) {  
 **if** (checkRowCol(board[0][i], board[1][i], board[2][i]) == **true**) {  
 **return** **true**;  
 }  
 }  
 **return** **false**;  
 }  
  
 *// Check the two diagonals to see if either is a win. Return true if either wins.*  
 **private** **boolean** checkDiagonals() {  
 **return** ((checkRowCol(board[0][0], board[1][1], board[2][2]) == **true**) || (checkRowCol(board[0][2], board[1][1], board[2][0]) == **true**));  
 }  
  
 *// Check to see if all three values are the same (and not empty) indicating a win.*  
 **private** **boolean** checkRowCol(**char** c1, **char** c2, **char** c3) {  
 **return** ((c1 != **'-'**) && (c1 == c2) && (c2 == c3));  
 }  
  
 *// Change player marks back and forth.*  
 **public** **void** changePlayer() {  
 **if** (currentPlayer == **'x'**) {  
 currentPlayer = **'o'**;  
 } **else** {  
 currentPlayer = **'x'**;  
 }  
 }  
  
 *// Places a mark at the cell specified by row and col with the mark of the current player.*  
 **public** **boolean** placeMark(**int** row, **int** col) {  
  
 *// Make sure that row and column are in bounds of the board.*  
 **if** ((row >= 0) && (row < 3)) {  
 **if** ((col >= 0) && (col < 3)) {  
 **if** (board[row][col] == **'-'**) {  
 board[row][col] = currentPlayer;  
 **return** **true**;  
 }  
 }  
 }  
  
 **return** **false**;  
 }  
}

**Output:**

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generatedText

Description automatically generated

**Practical 19**

**Aim:** WAP to create Tic-Tac-Toe game ( VS computer version)

**Program:**

**import** java.util.\*;

**public** **class** TicTacToeVsComputer {  
  
 **static** ArrayList<Integer> playerPositions = **new** ArrayList<Integer>();  
 **static** ArrayList<Integer> cpuPositions = **new** ArrayList<Integer>();  
  
 **public** **static** **void** main(String[] args) {  
 **char**[][] gameBoard = {{**' '**, **'|'**, **' '**, **'|'**, **' '**},  
 {**'-'**, **'-'**, **'-'**, **'-'**, **'-'**},  
 {**' '**, **'|'**, **' '**, **'|'**, **' '**},  
 {**'-'**, **'-'**, **'-'**, **'-'**, **'-'**},  
 {**' '**, **'|'**, **' '**, **'|'**, **' '**}};  
 System.out.println(**"Hello! player 1 First turn is yours.\nand your mark is \"X\""**);  
 printGameBoard(gameBoard);  
 Scanner scan = **new** Scanner(System.in);  
 **while** (**true**) {  
 System.out.println(**"Enter your position (1-9):"**);  
 **int** playerPos = scan.nextInt();  
 **while** (playerPositions.contains(playerPos) || cpuPositions.contains(playerPos)) {  
 System.out.println(**"position taken ! enter correct position"**);  
 playerPos = scan.nextInt();  
 }  
 placeMove(gameBoard, playerPos, **"player"**);  
 String result = checkWinner();  
 **if** (result.length() > 0) {  
 System.out.println(result);  
 **break**;  
 }  
 Random rand = **new** Random();  
 **int** cpuPos = rand.nextInt(9) + 1;  
 **while** (playerPositions.contains(cpuPos) || cpuPositions.contains(cpuPos)) {  
 System.out.println(**"position taken ! enter correct position"**);  
 cpuPos = rand.nextInt(9) + 1;  
 }  
 placeMove(gameBoard, cpuPos, **"cpu"**);  
 System.out.println(**"\n"**);  
 printGameBoard(gameBoard);  
 result = checkWinner();  
 **if** (result.length() > 0) {  
 System.out.println(result);  
 **break**;  
 }  
 }  
  
 }  
  
 **public** **static** **void** printGameBoard(**char**[][] gameBoard) {  
 **for** (**char**[] row : gameBoard) {  
 **for** (**char** c : row) {  
 System.out.print(c);  
 }  
 System.out.println();  
 }  
 }  
  
 **public** **static** **void** placeMove(**char**[][] gameBoard, **int** pos, String user) {  
 **char** symbol = **' '**;  
 **if** (user.equals(**"player"**)) {  
 symbol = **'X'**;  
 playerPositions.add(pos);  
 } **else** **if** (user.equals(**"cpu"**)) {  
 symbol = **'0'**;  
 cpuPositions.add(pos);  
 }  
 **switch** (pos) {  
 **case** 1:  
 gameBoard[0][0] = symbol;  
 **break**;  
 **case** 2:  
 gameBoard[0][2] = symbol;  
 **break**;  
 **case** 3:  
 gameBoard[0][4] = symbol;  
 **break**;  
 **case** 4:  
 gameBoard[2][0] = symbol;  
 **break**;  
 **case** 5:  
 gameBoard[2][2] = symbol;  
 **break**;  
 **case** 6:  
 gameBoard[2][4] = symbol;  
 **break**;  
 **case** 7:  
 gameBoard[4][0] = symbol;  
 **break**;  
 **case** 8:  
 gameBoard[4][2] = symbol;  
 **break**;  
 **case** 9:  
 gameBoard[4][4] = symbol;  
 **break**;  
 **default**:  
 **break**;  
 }  
 }  
  
 **public** **static** String checkWinner() {  
 List topRow = Arrays.asList(1, 2, 3);  
 List midRow = Arrays.asList(4, 5, 6);  
 List botRow = Arrays.asList(7, 8, 9);  
 List leftCol = Arrays.asList(1, 4, 7);  
 List midCol = Arrays.asList(2, 5, 8);  
 List rightCol = Arrays.asList(3, 6, 9);  
 List cross1 = Arrays.asList(1, 5, 9);  
 List cross2 = Arrays.asList(7, 5, 3);  
 List<List> winning = **new** ArrayList<List>();  
 winning.add(topRow);  
 winning.add(midRow);  
 winning.add(botRow);  
 winning.add(leftCol);  
 winning.add(midCol);  
 winning.add(rightCol);  
 winning.add(cross1);  
 winning.add(cross2);  
 **for** (List l : winning) {  
 **if** (playerPositions.containsAll(l)) {  
 **return** **" you won !"**;  
 } **else** **if** (cpuPositions.containsAll(l)) {  
 **return** **" Cpu wins !"**;  
 } **else** **if** (playerPositions.size() + cpuPositions.size() == 9) {  
 **return** **" Tie !"**;  
 }  
 }  
 **return** **""**;  
 }  
}

**Output:**

**Text

Description automatically generatedText

Description automatically generated**

**Practical 20**

**Aim:** WAP to create Rock-Pape Scissors game

**Program:**

**import** java.util.Random;  
**import** java.util.Scanner;  
  
**public** **class** RockPaperScissorsGame {  
  
 **public** **static** **final** String ROCK = **"ROCK"**;  
 **public** **static** **final** String PAPER = **"PAPER"**;  
 **public** **static** **final** String SCISSORS = **"SCISSORS"**;  
  
 **public** **static** **void** main(String args[]) {  
 System.out.println(**"Enter any one of the following inputs: "**);  
 System.out.println(**"ROCK"**);  
 System.out.println(**"PAPER"**);  
 System.out.println(**"SCISSORS"**);  
 System.out.println();  
  
 String playerMove = getPlayerMove();  
 String computerMove = getComputerMove();  
  
 *//Below function specifies the Rules of the Game.*  
 */\*if both playerMove and computerMove  
 produces the same formation, then  
 Game is a tie\*/*  
 **if** (playerMove.equals(computerMove))  
 System.out.println(**"Game is Tie !!"**);  
 *// if playerMove is ROCK*  
 **else** **if** (playerMove.equals(RockPaperScissorsGame.ROCK))  
 System.out.println(computerMove.equals(RockPaperScissorsGame.PAPER) ? **"Computer Wins"** : **"Player wins"**);  
 *// if playerMove is PAPER*  
 **else** **if** (playerMove.equals(RockPaperScissorsGame.PAPER))  
 System.out.println(computerMove.equals(RockPaperScissorsGame.SCISSORS) ? **"Computer Wins"** : **"Player wins"**);  
 *// if playerMove is SCISSORS*  
 **else**  
 System.out.println(computerMove.equals(RockPaperScissorsGame.ROCK) ? **"Computer Wins"** : **"Player wins"**);  
 }  
  
 */\* Get Computer's move using Random  
 class nextInt() method \*/*  
 **public** **static** String getComputerMove() {  
 String computermove;  
 Random random = **new** Random();  
 **int** input = random.nextInt(3) + 1;  
 **if** (input == 1)  
 computermove = RockPaperScissorsGame.ROCK;  
 **else** **if** (input == 2)  
 computermove = RockPaperScissorsGame.PAPER;  
 **else**  
 computermove = RockPaperScissorsGame.SCISSORS;  
  
 System.out.println(**"Computer move is: "** + computermove);  
 System.out.println();  
 **return** computermove;  
 }  
  
 */\* Get Player's move using Scanner  
 class \*/*  
 **public** **static** String getPlayerMove() {  
 Scanner in = **new** Scanner(System.in);  
 String input = in.next();  
 String playermove = input.toUpperCase();  
 System.out.println(**"Player move is: "** + playermove);  
 **return** playermove;  
 }  
}

Text

Description automatically generated**Output:**

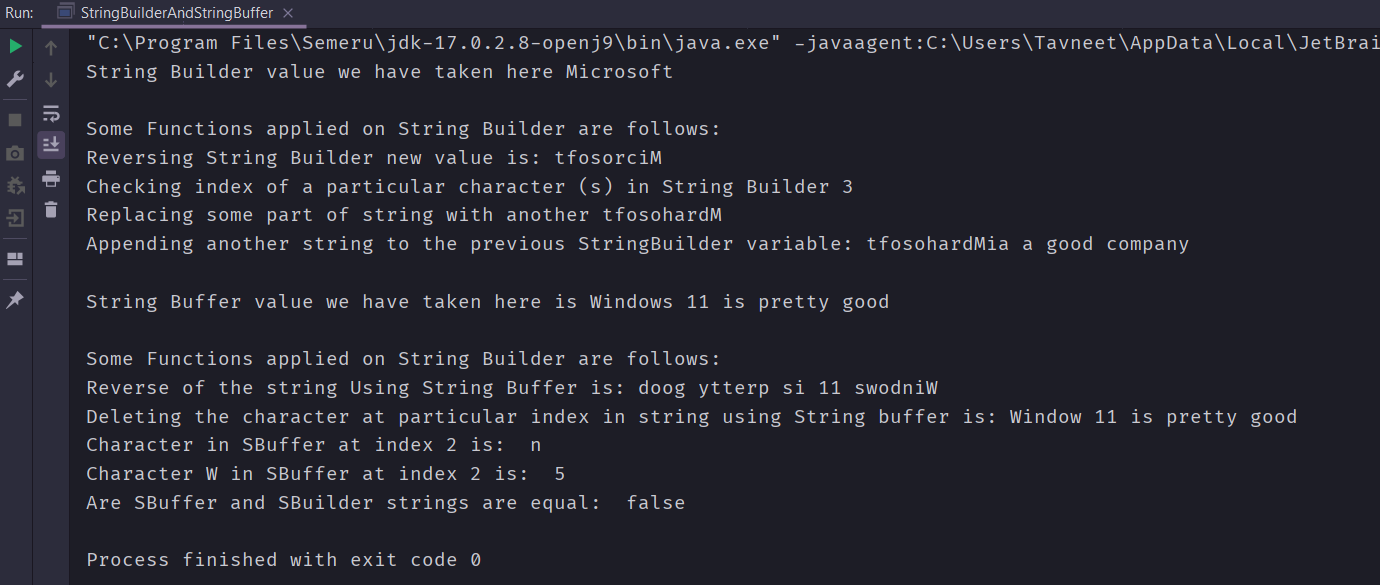
**Practical 21**

**Aim:** Demonstrate the functions available in StringBuilder/StringBuffer classes.

**Program:**

**public** **class** StringBuilderAndStringBuffer {  
 **public** **static** **void** main(String[] args) {  
  
 String str = **"Microsoft"**;  
 StringBuilder SBuild1 = **new** StringBuilder(str);  
  
 System.out.println(**"String Builder value we have taken here "** + SBuild1);  
 System.out.println(**"\nSome Functions applied on String Builder are follows:"**);  
 System.out.println(**"Reversing String Builder new value is: "** + SBuild1.reverse());  
 System.out.println(**"Checking index of a particular character (s) in String Builder "** + SBuild1.indexOf(**"s"**));  
 System.out.println(**"Replacing some part of string with another "** + SBuild1.replace(5, 8, **"hard"**));  
 System.out.println(**"Appending another string to the previous StringBuilder variable: "** + SBuild1.append(**"ia a good company"**));  
  
  
 StringBuffer SBuffer = **new** StringBuffer(**"Windows 11"**);  
 SBuffer.append(**"is pretty good"**);  
 System.out.println(**"\nString Buffer value we have taken here is "** + SBuffer);  
 System.out.println(**"\nSome Functions applied on String Builder are follows:"**);  
 System.out.println(**"Reverse of the string Using String Buffer is: "** + SBuffer.reverse());  
 System.out.println(**"Capitalization of all the elements in string using String buffer is: "** + SBuffer.deleteCharAt(6));  
 System.out.println(**"Character in SBuffer at index 2 is: "** + SBuffer.charAt(2));  
 System.out.println(**"Character W in SBuffer at index 2 is: "** + SBuffer.indexOf(**"w"**));  
 System.out.println(**" Are a SBuffer and SBuilder strings are equal: "** + SBuffer.equals(SBuild1));  
  
 }  
}

**Output:**

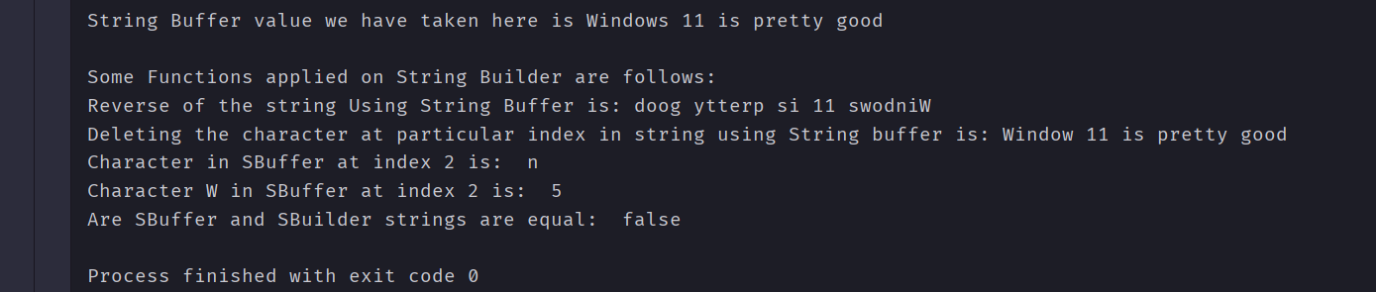


**Practical 22**

**Aim:** Write program to demonstrate various String/String Buffer/String Builder functions for: Reversing a string( and check Palindrome), Conversion to Upper Case/Lower Case, Equality check of given String, Inserting/Deleting a character in string, Fetching the index of a particular character/substring, Replacing the occurrence of a substring/character in a String, comparing strings Lexicographically, splitting a string based on a given token etc.

**Program:**

**public** **class** NormalStringStringBuilderAndStringBufferFunctions {  
 **public** **static** **void** main(String[] args) {  
 String str1 = **"IBM research"**;  
 String str2 = **"Microsoft"**;  
 StringBuilder SBuild1 = **new** StringBuilder(str2);  
  
 System.out.println(**"\nNormal String value that we have taken is: "**+ str1);  
 System.out.println(**"Applying some function on normal string:"**);  
 System.out.println(**"After Capitalization of every character in string we get: "**+ str1.toUpperCase());  
 System.out.println(**"After lower casing of every character in string we get: "**+ str1.toLowerCase());  
 System.out.println(**"Getting length of the entered string which is: "**+ str1.length());  
 String[] words = str1.split(**""**);  
 System.out.println(**"After splitting words present in string based on space we get: "**);  
 **for**(String w: words){  
 System.out.println(w);  
 }  
  
 System.out.println(**"String Builder value we have taken here "** + SBuild1);  
 System.out.println(**"\nSome Functions applied on String Builder are follows:"**);  
 System.out.println(**"Reversing String Builder new value is: "** + SBuild1.reverse());  
 System.out.println(**"Checking index of a particular character (s) in String Builder "** + SBuild1.reverse().indexOf(**"s"**));  
 System.out.println(**"Replacing some part of string with another "** + SBuild1.replace(5, 8, **"hard"**));  
 System.out.println(**"Appending another string to the previous StringBuilder variable: "** + SBuild1.append(**"ia a good company"**));  
  
  
 StringBuffer SBuffer = **new** StringBuffer(**"Windows 11"**);  
 SBuffer.append(**" is pretty good"**);  
 System.out.println(**"\nString Buffer value we have taken here is "** + SBuffer);  
 System.out.println(**"\nSome Functions applied on String Builder are follows:"**);  
 System.out.println(**"Reverse of the string Using String Buffer is: "** + SBuffer.reverse());  
 System.out.println(**"Deleting the character at particular index in string using String buffer is: "** + (SBuffer.reverse()).deleteCharAt(6));  
 System.out.println(**"Character in SBuffer at index 2 is: "** + SBuffer.charAt(2));  
 System.out.println(**"Character W in SBuffer at index 2 is: "** + SBuffer.indexOf(**"w"**));  
 System.out.println(**"Are SBuffer and SBuilder strings are equal: "** + SBuffer.equals(SBuild1));  
  
 }  
}

Text

Description automatically generated**Output:**

**Practical 23**

**Aim:** You are given a string at run time You are required to capitalize the first letter of every word in that string.

**Program:**

**import** java.util.Scanner;  
  
**public** **class** CapitalizeFirstLetter {  
 **public** **static** **void** main(String[] args) {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the String"**);  
  
 String str = take.nextLine();  
  
  
 *// hello my name is tavneet*  
 String[] words = str.split(**" "**);  
  
 *// words[] = [hello, my, name, is, tavneet]*  
  
 String newString = **""**;  
  
 **for** (String w : words) {  
 String oneWord = w.substring(0, 1);  
 String rest = w.substring(1);  
 newString = newString + oneWord.toUpperCase() + rest + **" "**;  
 }  
  
  
 System.out.println(**"After Capitalization of First character of every word we have: "** + newString.trim());  
 }  
}

Text

Description automatically generated**Output:**

**Practical 24**

**Aim:** You are given a string at run time. Capitalize all the characters at all the odd indexes in the string e.g., at index 1, 3, 5, etc.

**Program:**

**import** java.util.Scanner;  
  
**public** **class** CapitalizeAtOddIndexes {  
 **public** **static** **void** main(String[] args) {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the String"**);  
  
 String str = take.nextLine();  
  
 **char**[] charArray = str.toCharArray();  
  
 StringBuffer newString = **new** StringBuffer();  
  
  
 **for** (**int** i = 0; i < str.length(); i++) {  
  
 **char** c = charArray[i];  
 **if** (i % 2 != 0) {  
 c = Character.toUpperCase(c);  
 }  
 newString.append(c);  
 }  
  
  
 System.out.println(**"\nAfter Capitalization of character at odd indexes in a string we get: "** + newString.toString());  
 }  
}

Text

Description automatically generated**Output:**

**Practical 25**

**Aim:** You are given a string at run time. You are required to fetch the remaining String after the occurrence of a certain sequence/character. E.g., fetch the remaining String after the occurrence of “and” in the String.

**Program:**

**import** java.util.Scanner;  
  
**public** **class** SplitOnCharacterOrWord {  
 **public** **static** **void** main(String[] args) {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the String"**);  
  
 String str = take.nextLine();  
  
 System.out.println(**"\nEnter any character or word in basi of which you want to split the main String:"**);  
 String strSplit = take.nextLine();  
 String[] words = str.split(strSplit);  
  
 System.out.println(**"\nAfter doing splitting based on split word or string("**+strSplit+**") we get:"**);  
 **for** (String w : words) {  
 System.out.println(w);  
 }  
 }  
}

Text

Description automatically generated**Output:**

**Practical 26**

**Aim:** You are given a string at run time. Count the occurrences of a given word/sequence in that string.

**Program:**

**import** java.util.Scanner;  
  
**public** **class** CountOccurences {  
 **public** String str;  
 **public** String newString;  
  
 **public** **void** CountOccurencesOfword() {  
 String countWord;  
 **int** count = 0;  
 System.out.println(**"\nEnter the word which you want to count in string:"**);  
 Scanner sc = **new** Scanner(System.in);  
 countWord = sc.nextLine();  
 String[] words = str.split(**" "**);  
 **for** (String w : words) {  
 **if** (w.equalsIgnoreCase(countWord)) {  
 count++;  
 }  
 }  
 System.out.println(**"The entered word \""** + countWord + **"\" occures \""** + count + **"\" times in our main string."**);  
  
 }  
  
 **public** **void** CountOccurencesOfAlphbet() {  
 String countChar;  
 **int** count = 0;  
 System.out.println(**"\nEnter the Alphabet or character which you want to count in string:"**);  
 Scanner sc = **new** Scanner(System.in);  
 countChar = sc.nextLine();  
 **char**[] charArray = str.toCharArray();  
  
 **for** (**int** i = 0; i < str.length(); i++) {  
  
 **if** (countChar.equalsIgnoreCase(String.valueOf(charArray[i]))) {  
 count++;  
 }  
 }  
 System.out.println(**"The entered word \""** + countChar + **"\" occures \""** + count + **"\" times in our main string."**);  
  
 }  
  
 **public** **static** **void** main(String[] args) {  
 CountOccurences countOccurences = **new** CountOccurences();  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the String"**);  
 countOccurences.str = take.nextLine();  
  
 System.out.println(**"Enter 1 if you want to search for words and 2 if you want to count character:"**);  
 **int** option = take.nextInt();  
 **if** (option == 1) {  
 countOccurences.CountOccurencesOfword();  
 } **else** **if** (option == 2) {  
 countOccurences.CountOccurencesOfAlphbet();  
 } **else** {  
 System.out.println(**" Wrong option chosen. Please try again.."**);  
 }  
  
 }  
}

**Output:**

Text

Description automatically generated**1) For counting character/alphabet in a string:**

Text

Description automatically generated**2) For counting words in a string:**

**Practical 27**

**Aim:** You are given a string at run time. Delete all the duplicate words from the String(ignore case) and return modified String.

**Program:**

**import** java.util.Arrays;  
**import** java.util.LinkedHashSet;  
**import** java.util.Scanner;  
  
**public** **class** RemoveDuplicateWords {  
 **public** **static** **void** main(String[] args) {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the String"**);  
  
 String str = take.nextLine();  
  
 String[] words = str.split(**" "**);  
  
 LinkedHashSet<String> hashSetWords = **new** LinkedHashSet<String>(Arrays.asList(words));  
  
 StringBuilder newString = **new** StringBuilder();  
 **int** i = 0;  
 **for** (String w : hashSetWords) {  
 **if** (i > 0) {  
 newString.append(**" "**);  
 }  
  
 newString.append(w);  
 i++;  
 }  
  
  
 System.out.println(**"After removal of duplicate words, new string formed is: "** + newString);  
 }  
}

Text

Description automatically generated**Output:**

**Practical 28**

**Aim:** You are Given a String at run time. Find total vowels in the given String and also display the count of each vowel in the string.

**Program:**

**import** java.util.Scanner;  
  
**public** **class** CountVowels {  
 **public** **static** **void** main(String args[]) {  
 **int** vCount = 0;  
 Scanner sc = **new** Scanner(System.in);  
 System.out.println(**"Enter the string: "**);  
 String str = sc.nextLine();  
 **int**[] count = **new** **int**[5];  
  
 **for** (**int** i = 0; i < str.length(); i++) {  
 **char** ch = str.charAt(i);  
 **if** (ch == **'a'** || ch == **'A'**) {  
 count[0]++;  
 } **else** **if** (ch == **'e'** || ch == **'E'**) {  
 count[1]++;  
 } **else** **if** (ch == **'i'** || ch == **'I'**) {  
 count[2]++;  
 } **else** **if** (ch == **'o'** || ch == **'O'**) {  
 count[3]++;  
 } **else** **if** (ch == **'u'** || ch == **'U'**) {  
 count[4]++;  
 }  
 }  
 System.out.println(**"Count of 'a' vowel in entered string is = "** + count[0]);  
 System.out.println(**"Count of 'e' vowel in entered string is = "** + count[1]);  
 System.out.println(**"Count of 'i' vowel in entered string is = "** + count[2]);  
 System.out.println(**"Count of 'o' vowel in entered string is = "** + count[3]);  
 System.out.println(**"Count of 'u' vowel in entered string is = "** + count[4]);  
  
 }  
}

**Output:**

Text

Description automatically generated

**Practical 29**

**Aim:** You are given a string at run time. Replace all the characters at the even index by the character present at the next even index, Replace the character at the last even index by “1” e.g., “HELLO” shall return LEOL1.

**Program:**

**import** java.util.Scanner;  
  
**public** **class** EvenIndexes {  
 **public** **static** **void** main(String[] args) {  
 Scanner take = **new** Scanner(System.in);  
 System.out.println(**"Enter the String:"**);  
  
 String str = take.nextLine();  
 StringBuilder SBuilder = **new** StringBuilder();  
 **char**[] stringCharacters = str.toCharArray();  
  
 **for** (**int** i = 0; i < stringCharacters.length; i++) {  
 **char** c = stringCharacters[i];  
  
 **if** (i % 2 == 0) {  
 **if** (i >= stringCharacters.length) {  
 c = 1;  
 } **else** **if** (i < stringCharacters.length) {  
 **if** ((i + 2) < stringCharacters.length) {  
 c = stringCharacters[i + 2];  
 }  
 }  
  
 }  
 SBuilder.append(c);  
  
  
 }  
 System.out.println(**"Modified string is: "** + SBuilder);  
  
 }  
  
}

Text

Description automatically generated**Output:**

**Practical 30**

**Aim:** Demonstrate parent child relationship among classes add call the method available by virtue of inheritance in child class

**Program:**

**interface** dev {  
 **public** **void** devnews();  
}  
  
**class** windows {  
 **void** news() {  
 System.out.println(**"Windows 11 is launched"**);  
 }  
}  
  
**class** microsoft **extends** windows **implements** dev {  
 **public** **void** devnews() {  
 System.out.println(**"DotNet 6.0 hits General availability"**);  
 }  
  
}  
  
**public** **class** InheritenceProgram {  
 **public** **static** **void** main(String[] args) {  
  
 microsoft M = **new** microsoft(); *// M is the obect of Microsoft class*  
  
 M.devnews(); *// calls devnews() function*  
  
 M.news(); *// calls news method*  
  
 }  
}

Text

Description automatically generatedOutput:

**Practical 31**

**Aim:** Demonstrate method overloading add overriding add the use of super keyword in assessing parent class constructor add overridden methods

**Program:**

**import** java.util.Scanner;  
  
**class** Python {  
 **void** hello() {  
 System.out.println(**"Hello From Python Class"**);  
 }  
}  
  
**class** DotNet **extends** Python {  
 **void** hello() {  
 System.out.println(**"Hello From Dotnet Class"**);  
 }  
}  
  
**class** AirthmeticOverloading {  
 **public** **void** multiply(**int** a, **int** b) {  
 System.out.println(**"Multiplication of two entered integer numbers gives us: "** + (a \* b));  
 }  
  
 **public** **void** multiply(**double** a, **double** b) {  
 System.out.println(**"Multiplication of two entered decimal numbers gives us: "** + (a \* b));  
 }  
}  
  
  
**public** **class** MethodOverloadingAndOverriding {  
 **public** **static** **void** main(String[] args) {  
 AirthmeticOverloading airthmeticOverloading = **new** AirthmeticOverloading();  
  
 Python p = **new** Python();  
  
 DotNet m = **new** DotNet();  
  
 Python ref; *// refrence object of python class*  
  
 ref = p; *//refrence points to the python object*  
  
 ref.hello(); *// calling the hello function present in python class*  
  
 ref = m; *// refrence points to the dotnet object*  
  
 ref.hello(); *// calling the hello function present in dotnet calss*  
  
 Scanner sc = **new** Scanner(System.in);  
  
 System.out.println(**"\nEnter any two integer values for multiplication: "**);  
 **int** a = sc.nextInt();  
 **int** b = sc.nextInt();  
 airthmeticOverloading.multiply(a, b);  
  
 System.out.println(**"\nEnter any two decimal values for multiplication: "**);  
 **double** da = sc.nextDouble();  
 **double** db = sc.nextDouble();  
 airthmeticOverloading.multiply(da, db);  
 }  
}

**Output:**

A screenshot of a computer

Description automatically generated

**Practical 32**

**Aim:** Demonstrate that a parent class reference variable can be used to point to the subclass object but using a parent class reference variable we can call only those methods which are available in the parent class.

**Program:**

*// Dynamic method dispatch in java*  
**class** OpenPython {  
 **void** hello() {  
 System.out.println(**"Hello From Python Class"**);  
 }  
}  
  
**class** OpenDotNet **extends** OpenPython {  
 **void** hello() {  
 System.out.println(**"Hello From Dotnet Class"**);  
 }  
}  
  
**public** **class** ReferenceVar {  
 **public** **static** **void** main(String[] args) {  
 OpenPython p = **new** OpenPython();  
  
 OpenDotNet m = **new** OpenDotNet();  
  
 OpenPython ref; *// refrence object of python class*  
  
 ref = p; *//refrence points to the python object*  
  
 ref.hello(); *// calling the hello function present in python class*  
  
 ref = m; *// refrence points to the dotnet object*  
  
 ref.hello(); *// calling the hello function present in dotnet calss*  
 }  
}

**Output:**

Text

Description automatically generated

**Practical 33**

**Aim:** Demonstrate that a parent class reference variable can be used to point to the subclass object but using a parent class reference variable we can call only those methods which are available in the parent class.

**Program:**

**class** Ms {  
 **private** **void** dev() { *// only available to the class Microsoft*  
 System.out.println(**"Visual studio 2022 new preview has been released."**);  
 }  
  
 **public** **void** xbox() {  
 System.out.println(**"Xbox series X and S has been launched"**);  
 }  
  
 **protected** **void** security() { *// cn be acced within same package or in the subclasses of same package*  
 System.out.println(**"windows defender ATP service version: "**);  
 }  
  
 **void** azure() { *// when no access specifier is mentioned it is default specifier.*  
 System.out.println(**"The main business of microsoft is MS Azure Cloud"**);  
 }  
}  
  
  
**public** **class** AccessModifiers {  
 **public** **static** **void** main(String[] args) {  
 Ms microsoft = **new** Ms();  
 *//microsoft.dev(); //not possible here because dev() function has private access specifier on it.*  
 microsoft.xbox();  
 microsoft.security();  
 microsoft.azure();  
  
 }  
}

Text

Description automatically generated**Output:**

**Practical 34**

**Aim:** (a) Demonstrate the various methods available in ArayList class.

(b) Create an ArrayList of objects(say Student/Employee through the ArrayList to display all the information of objects.

**Program: (a) Demonstrate the various methods available in ArayList class**

**import** java.util.ArrayList;  
**public** **class** RawArrayList {  
 **public** **static** **void** main(String[] args) {  
 *// Declaration of raw array list*  
 ArrayList Arrl1 = **new** ArrayList();  
  
 *// Assigning values to this raw array list;*  
 Arrl1.add(5); *// Here we are providing it with value 5 which is of primitive data type Int But Arraylist only*  
 *// takes array of objects. so, here the .add() function automatically converts this primitive data type to an*  
 *// object by using an Integer wrapper function Internally.*  
 *// Here the integer wrapper function is used internally because here the primitive data type passed to the*  
 *// .add() function is of type Integer here.*  
 *// This whole phenomena of automatically converting primitive data type to an object is called Autoboxing.*  
  
 *// Creating Integer object and Float Object*  
 Integer InOb1 = **new** Integer(6);  
 Float FlOb1 = **new** Float(0.7F);  
  
 *// Adding objects to the ArrayList Arrl1*  
 Arrl1.add(InOb1);  
 Arrl1.add(FlOb1);  
  
 *// This gives us all the elements present in Raw Array List*  
 System.out.println(**"Values Contained in Raw Array List are: "**+Arrl1+**"\n"**);  
  
 *// Now if we want to get particular element from Raw Array list we need to know its data type*  
 *// at what location which type of data is stored in Raw Array List.*  
 *// so that we can do its Typecasting and extract Dta from our Raw Array List.*  
 *// here as we know that at first 2 location/Indexes integer type of objects(data) are stored and at*  
 *// Third and Last the object (data) of type float is stored.*  
 Integer Ind1 = (Integer) Arrl1.get(0);  
 Integer Ind2 = (Integer) Arrl1.get(1);  
 Float Fld1 = (Float) Arrl1.get(2);  
 *// Here in above lines Ind1 , Ind2 contains the integer values contained at index 0 and 1 respectively*  
 *// of the Raw ArrayList Arrl1 and Fld1 contains the Float type data Contained at index 2 of the*  
 *// Raw ArrayList Arrl1.*  
  
 System.out.println(**"Value at Index 0 of Raw ArrayList Arrl1 is: "** + Ind1);  
 System.out.println(**"Value at Index 1 of Raw ArrayList Arrl1 is: "** + Ind2);  
 System.out.println(**"Value at Index 2 of Raw ArrayList Arrl1 is: "** + Fld1);  
  
 }  
  
}

**Output:**

Text

Description automatically generated

**Program: (b) Create an ArrayList of objects(say Student/Employee through the ArrayList to display all the information of objects.**

**Students.java File:**

**public** **class** Students {  
 String Name;  
 **int** Rollno;  
 **int** Classno;  
  
 **public** Students() {  
 }  
  
 Students(String Name, **int** Rollno, **int** Class) {  
 **this**.Name = Name;  
 **this**.Rollno = Rollno;  
 **this**.Classno = Class;  
  
 }  
  
 **public** **void** setName(String Name) {  
 **this**.Name = Name;  
 }  
  
  
 **public** **void** setRollno(**int** Rollno) {  
 **this**.Rollno = Rollno;  
 }  
  
 **public** **void** setClassno(**int** Classno) {  
 **this**.Classno = Classno;  
 }  
  
 **public** String getName() {  
 **return** **this**.Name;  
 }  
  
 **public** **int** getRollno() {  
 **return** **this**.Rollno;  
 }  
  
 **public** **int** getClassno() {  
 **return** **this**.Classno;  
 }  
}

**ArrayListObjects.java File:**

**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public** **class** ArrayListObjects {  
 **public** **static** **void** main(String[] args) {  
  
 ArrayList<Students> studentsArrayList = **new** ArrayList<Students>();  
  
 Students student1 = **new** Students(**"Tavneet singh"**, 10, 13);  
 Students student2 = **new** Students(**"Satya"**, 01, 13);  
 Students student3 = **new** Students(**"Sundar"**, 02, 13);  
  
 studentsArrayList.add(student1);  
 studentsArrayList.add(student2);  
 studentsArrayList.add(student3);  
  
 System.out.println(**"Entered Records are:"**);  
 **int** i = 1;  
 **for** (Students student : studentsArrayList) {  
 System.out.println(i + **") Student details are: "** + **"Name:"** + student.Name + **" Rollno.:"** + student.Rollno + **" Class:"** + student.Classno);  
 i++;  
 }  
  
  
 }  
}

**Output:**

Text

Description automatically generated

**Practical 35**

**Aim: The University Problem**

You are requested to input the details of the entire University using the concept of Arrays/ArrayList and composition( an object of another object(). At the lowest level three shall be class Student. The field of student classes are rollNo, name & age. These fields have corresponding getteres(for fetching value) and setters (for setting value functions. There is a displayDetails() function which shall print all details of a student by using getters to fetch the value from the fields

**Program:** Here we have placed our program in university package, and we have different Java class files that makes our program work. These Java class files are used to divide the components/ functionality of our university program into different files. Now let us examine each one of them one by one.

**UniversityStudents.java File:**  This file defines what details are we going to ask to user each student.

**package** university;  
  
**public** **class** UniversityStudent{  
 String rollNo = **""**;  
 String name = **""**;  
 **int** age = 0;  
  
  
 **public** **void** setRoll(String rollNo) {  
 **this**.rollNo = rollNo;  
 }  
 **public** String getRoll() {  
 **return** rollNo;  
 }  
 **public** **void** setName(String name) {  
 **this**.name = name;  
 }  
 **public** String getName() {  
 **return** name;  
 }  
 **public** **void** setAge(**int** age) {  
 **this**.age = age;  
 }  
 **public** **int** getAge() {  
 **return** age;  
 }  
}

**Batch.java File:**

**package** university;  
  
**import** java.util.Scanner;  
  
**public** **class** Batch {  
 String name;  
 **int** size;  
 UniversityStudent[] stuarr;  
  
 **public** Batch(String name, **int** size) {  
 **this**.name = name;  
 **this**.size = size;  
 stuarr = **new** UniversityStudent[size];  
 }  
  
 **public** **void** fillBatch() {  
 Scanner sc = **new** Scanner(System.in);  
 Scanner sc1 = **new** Scanner(System.in);  
 Scanner sc2 = **new** Scanner(System.in);  
 **int** k =1;  
 **for** (**int** i = 0; i < size; i++) {  
 stuarr[i] = **new** UniversityStudent();  
 System.out.println(**"\nEnter Roll NO of ("**+ k +**") student"**);  
 String rollNo = sc1.nextLine();  
 System.out.println(**"Enter name of ("**+ k +**") student"**);  
 String name = sc1.nextLine();  
 System.out.println(**"Enter Age of("**+ k +**") of student"**);  
 **int** age = sc2.nextInt();  
 stuarr[i].setRoll(rollNo);  
 stuarr[i].setName(name);  
 stuarr[i].setAge(age);  
 k++;  
 }  
 }  
  
 **public** **void** displayBatch() {  
 System.out.println(**"\nName of Batch "** + **this**.name);  
 System.out.println(**"Number of students in the batch "** + **this**.size);  
 System.out.println(**"\nStudents Details in Batch "**);  
 **for** (UniversityStudent s : stuarr) {  
 System.out.println(**" Roll No of student is "** + s.getRoll());  
 System.out.println(**" Name of student is "** + s.getName());  
 System.out.println(**" Age of student is"** + s.getAge());  
 }  
 }  
}

**Department.java File:**

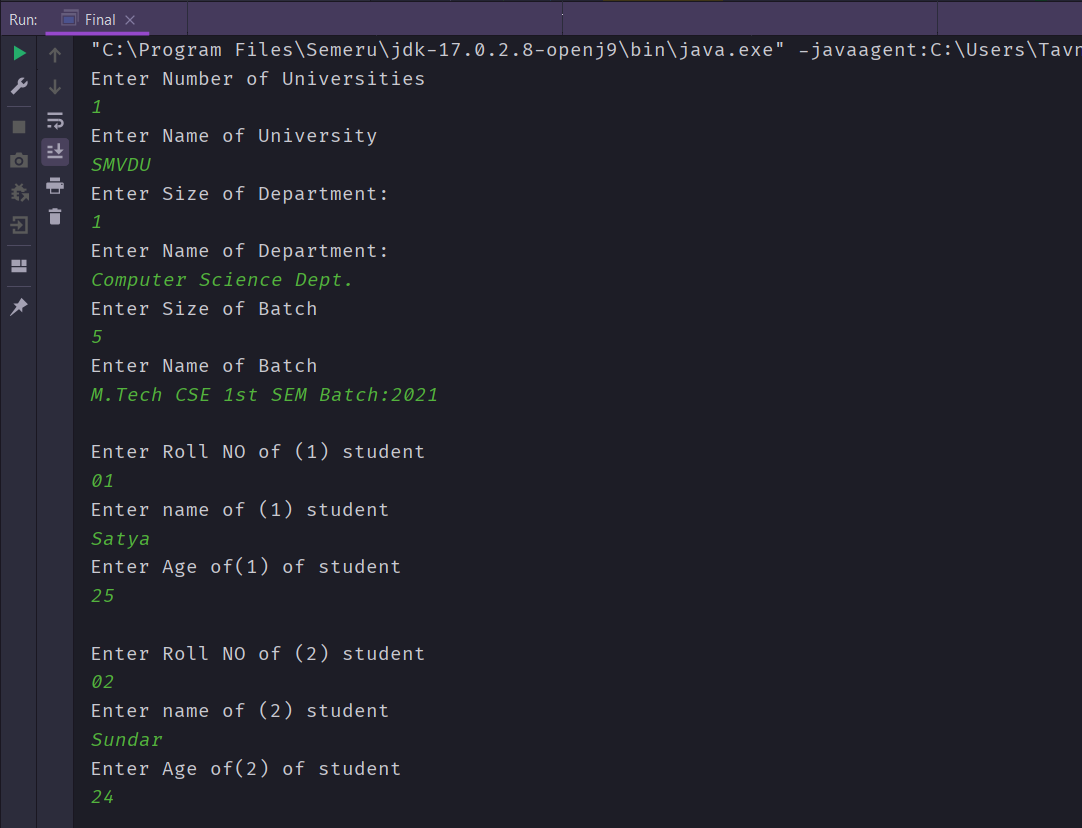
**package** university;  
  
**import** java.util.Scanner;  
  
**public** **class** Department {  
 String name;  
 **int** size;  
 Batch[] batcharr;  
  
 **public** Department(String name, **int** size) {  
 **this**.name = name;  
 **this**.size = size;  
 batcharr = **new** Batch[size];  
 }  
  
 **public** **void** fillDepartment() {  
 **for** (**int** i = 0; i < size; i++) {  
 Scanner sc = **new** Scanner(System.in);  
 Scanner sc1 = **new** Scanner(System.in);  
 System.out.println(**"Enter Size of Batch"**);  
 **int** size = sc.nextInt();  
 System.out.println(**"Enter Name of Batch"**);  
 String name = sc1.nextLine();  
  
 Batch b = **new** Batch(name, size);  
 batcharr[i] = b;  
 batcharr[i].fillBatch();  
 }  
 }  
  
 **public** **void** displayDepartment() {  
 System.out.println(**"Name of Department: "** + **this**.name);  
 System.out.println(**"Number of Batches in the Department: "** + **this**.size);  
 System.out.println(**"Batches Details in Department: "**);  
 **for** (**int** i = 0; i < size; i++) {  
 batcharr[i].displayBatch();  
 }  
 }  
}

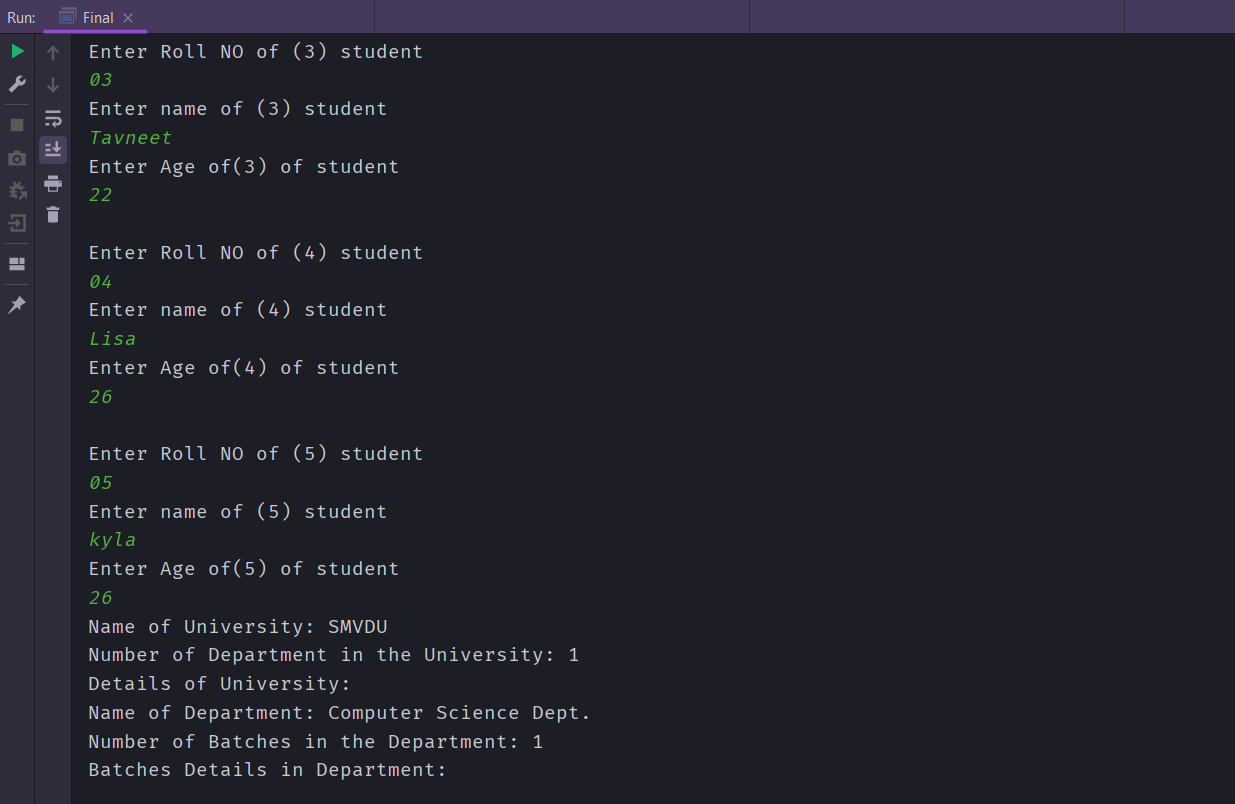
**University.java File:**

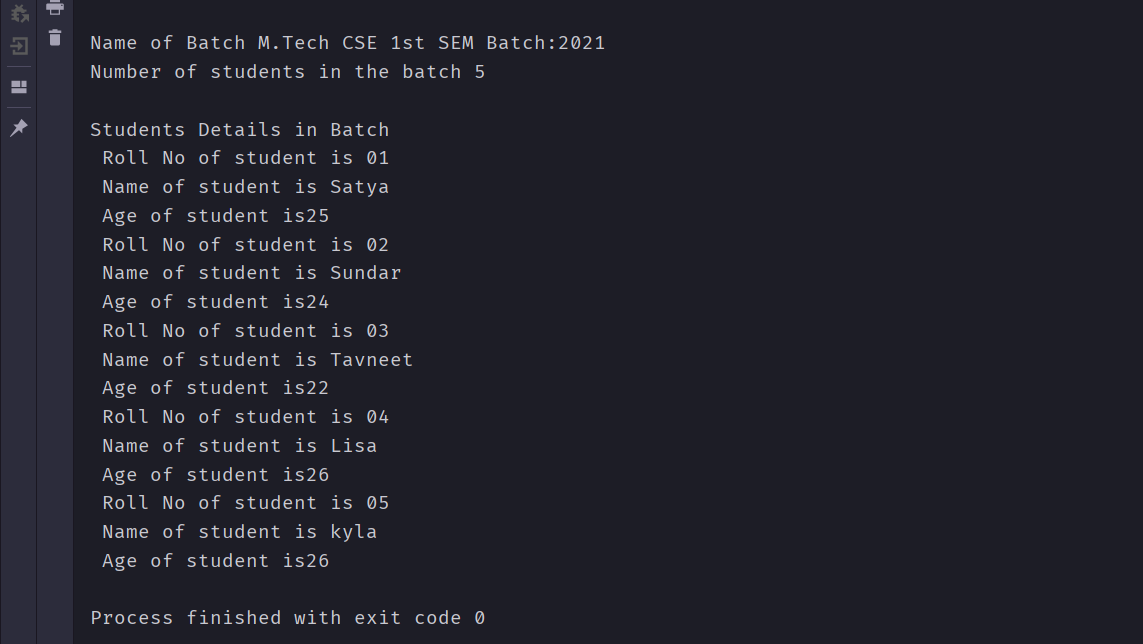
**package** university;  
  
**import** java.util.Scanner;  
  
**public** **class** University {  
 String name;  
 **int** size;  
 Department[] darr;  
 **public** University(String name, **int** size) {  
 **this**.name = name;  
 **this**.size = size;  
 darr = **new** Department[size];  
 }  
 **public** **void** fillUniversity() {  
 **for** (**int** i = 0; i < size; i++) {  
 Scanner sc = **new** Scanner(System.in);  
 Scanner sc1 = **new** Scanner(System.in);  
 System.out.println(**"Enter Size of Department:"**);  
 **int** size = sc.nextInt();  
 System.out.println(**"Enter Name of Department:"**);  
 String name = sc1.nextLine();  
  
 Department d = **new** Department(name, size);  
 darr[i] = d;  
 darr[i].fillDepartment();  
 }  
 }  
 **public** **void** displayUniversity() {  
 System.out.println(**"Name of University: "** + **this**.name);  
 System.out.println(**"Number of Department in the University: "** + **this**.size);  
 System.out.println(**"Details of University: "**);  
 **for** (**int** i = 0; i < size; i++) {  
 darr[i].displayDepartment();  
 }  
 }  
  
}

**Final.java:** This the main driver java file of our whole program. This file contains the main function which is an entry point to start execution of our program.

**package** university;  
  
**import** java.util.Scanner;  
  
**public** **class** Final {  
 **public** **static** **void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.in);  
 Scanner sc1 = **new** Scanner(System.in);  
 System.out.println(**"Enter Number of Universities"**);  
 **int** size = sc1.nextInt();  
 System.out.println(**"Enter Name of University"**);  
 String name = sc.nextLine();  
 University u = **new** University(name, size);  
 u.fillUniversity();  
 u.displayUniversity();  
  
 }  
}

**Output:**





**Practical 36**

**Aim:** WAP to make Database connectivity and insert records into database.

**Program:**

**import** java.sql.Connection;  
**import** java.sql.DriverManager;  
**import** java.sql.SQLException;  
**import** java.sql.Statement;  
**import** java.util.Scanner;  
  
**public** **class** StudentDatabaseConnection {  
 **public** **static** **void** main(String[] args) **throws** SQLException, ClassNotFoundException {  
  
 String driver = **"com.mysql.cj.jdbc.Driver"**;  
 Class.forName(driver);  
 Connection conn = DriverManager.getConnection(**"jdbc:mysql://localhost:3306/java\_students"**, **"root"**, **""**);  
 System.out.println(**"\nConnection string is"** + conn.toString());  
 System.out.println(**"\nHow many records you want to enter into the table: "**);  
 Scanner sc = **new** Scanner(System.in);  
 **int** r = sc.nextInt();  
 **for** (**int** i = 1; i <= r; i++) {  
 Scanner scname = **new** Scanner(System.in);  
 Scanner scrollno = **new** Scanner(System.in);  
 System.out.println(**"\nEnter the name for ("** + i + **") record: "**);  
 String inpName = scname.nextLine();  
 System.out.println(**"Enter the rollno for ("** + i + **") record: "**);  
 **int** inpRollno = scrollno.nextInt();  
  
 String name = **"\""** + inpName + **"\""**;  
 **int** rollno = inpRollno;  
  
 String query = **"INSERT into students VALUES ("** + name + **","** + rollno + **");"**;  
 System.out.println(i + **" Query is :"** + query);  
 Statement st = conn.createStatement();  
 **int** val = st.executeUpdate(query);  
  
 System.out.println(i + **" record added"**);  
 }  
  
 conn.close();  
  
 }  
}

**Output:**

Text

Description automatically generated

**Graphical user interface, application

Description automatically generatedStudents Table( with 2 columns and 3 rows entered):**

**Practical 37**

**Aim:** WAP to input a string at runtime, if the string contains the keyword “smvdu” throw a user defined named “SMVDUException”, if the string contains the keyword “university” Throw another exception named “UniversityException”. If the given string contains Both the keywords, throw both the exceptions, your program must handle both of these Exceptions. You may ignore the case of the input string.

**Program:**

**import** java.util.Scanner;  
  
**class** SMVDUException **extends** Exception {  
 **public** SMVDUException(String str) {  
  
 **super**(str);  
 }  
}  
  
**public** **class** CustomException {  
 **public** **static** **void** main(String args[]) **throws** SMVDUException {  
 Scanner sc = **new** Scanner(System.in);  
 System.out.println(**"Enter String "**);  
 String a = sc.nextLine();  
 String b = **"SMVDU"**;  
 String c = **"university"**;  
 **if** (a.equals(b)) {  
 **try** {  
 **throw** **new** SMVDUException(**"SMVDU Exception"**);  
  
 } **catch** (SMVDUException ex) {  
 System.out.println(ex.getMessage());  
 }  
  
 System.out.println(**"Out Of Exception"**);  
  
 } **else** **if** (a.equals(c)) {  
 **try** {  
 **throw** **new** SMVDUException(**"Univerrsity Exception"**);  
  
 } **catch** (SMVDUException ex) {  
 System.out.println(ex.getMessage());  
 }  
  
 System.out.println(**"Out Of Exception"**);  
 }  
  
 }  
}

Text

Description automatically generated**Output**(if we enter “SMVDU” as string):

**Text

Description automatically generatedOutput**(if we don’t enter “SMVDU” as string) we don’t get any exception**:**

**Practical 38**

**Aim:** WAP to create three threads(including the main thread) either by extending the Thread class or implementing the Runnable interface. The first thread prints Fibonacci series upto 1000, the second thread prints prime no. series upto 1000, the third thread prints multiples of 5 upto 1000. Each thread sleeps for 100 ms after printing a number. Observe the output of the program after repeated runs. Is it the same?

**Program:**

**public** **class** ThreadsProgram {  
 **public** **static** **void** main(String args[]) {  
  
 Thread1 m1 = **new** Thread1();  
 m1.start();  
 Thread2 m2 = **new** Thread2();  
 m2.start();  
 Thread3 m3 = **new** Thread3();  
 m3.start();  
 }  
}  
  
**class** Thread1 **extends** Thread {  
 **public** **void** run() {  
 System.out.println(**"Thread1"**);  
 **int** n1 = 0, n2 = 1, n3, i, count = 1000;  
 System.out.print(n1 + **" "** + n2);  
 **for** (i = 2; i < count; ++i) {  
 n3 = n1 + n2;  
 System.out.print(**" "** + n3);  
 n1 = n2;  
 n2 = n3;  
 }  
 **try** {  
 **this**.sleep(100);  
 } **catch** (InterruptedException ex) {  
 }  
 }  
}  
  
**class** Thread2 **extends** Thread {  
 **public** **void** run() {  
 System.out.println(**"Thread2"**);  
 **int** n;  
 **int** status = 1;  
 **int** num = 3;  
 n = 1000;  
 **if** (n >= 1) {  
 System.out.println(2);  
 }  
 **for** (**int** i = 2; i <= n; ) {  
 **for** (**int** j = 2; j <= Math.sqrt(num); j++) {  
 **if** (num % j == 0) {  
 status = 0;  
 **break**;  
 }  
 }  
 **if** (status != 0) {  
 System.out.println(num);  
 i++;  
 }  
 status = 1;  
 num++;  
 }  
 **try** {  
 **this**.sleep(100);  
 } **catch** (InterruptedException ex) {  
 }  
 }  
}  
  
**class** Thread3 **extends** Thread {  
 **public** **void** run() {  
 System.out.println(**"Thread3"**);  
 **for** (**int** number = 0; number < 1000; number++) {  
 **if** (number % 5 == 0) {  
 System.out.print(number + **" "**);  
 }  
 }  
 **try** {  
 **this**.sleep(100);  
 } **catch** (InterruptedException ex) {  
 }  
 }  
}

**Output: (Output of this program is very large that’s why we have not shown it here)**

**Practical 39**

**Aim:** WAP to demonstrate the use of try-catch-finally block, use of multiple catch statements, nested try-catch.

**Program:**

**public** **class** NestedTryCatchBlock {  
  
 **public** **static** **void** main(String args[]) {  
 **try** {  
 **try** {  
 System.out.println(**"Going to divide by 0"**);  
 **int** b = 27 / 0;  
 } **catch** (ArithmeticException e) {  
 System.out.println(e);  
 }  
 **try** {  
 **int** a[] = **new** **int**[7];  
 a[7] = 2;  
 } **catch** (ArrayIndexOutOfBoundsException e) {  
 System.out.println(e);  
 }  
 System.out.println(**"other statement"**);  
 } **catch** (Exception e) {  
 System.out.println(**"Handled the exception (outer catch)"**);  
 }  
 System.out.println(**"normal flow.."**);  
 }  
  
}

**Output:**

**Text

Description automatically generated**

**Practical 40**

**Aim:** WAP to implement comparable interface and override .hashCode() and .equals() function.

**Program:**

**import** java.util.Arrays;  
**import** java.util.Objects;  
  
**public** **class** ComparableClassDemo {  
 **public** **static** **void** main(String[] args) {  
 Employee Emp1 = **new** Employee(**"Tavneet"**, 89000.00); *// here as we are providing parameters while making*  
 Employee Emp2 = **new** Employee(**"Amit"**, 85000.00); *// objects of our Employee Class, so we are invoking*  
 Employee Emp3 = **new** Employee(**"Jasmeet"**, 83000.00); *// parameterized constructor of Employee Class*  
  
 System.out.println(Emp1);*// here by default the toString Function gets called on the object to which this*  
 *// reference variable points. Here we have overwritten the default toString() function. so now when we*  
 *// call the reference of an object of class Employee. our implementation of toString() function gets executed.*  
  
 System.out.println(**"Is the salaries of Employee 1 and 2 are equal: "** + Emp1.equals(Emp3) + **"\n"**); *// Compares 2 objects on the basis of condition specified in .equals()*  
 *// function.*  
 *// Here we have compared two objects on the basis of their salaries and checked weather their salaries are equal or not.*  
  
 System.out.println(**"Hash Code of Name of Employee 1 is: "** + Emp1.hashCode() + **"\n"**);  
 System.out.println(**"Hash Code of Name of Employee 2 is: "** + Emp2.hashCode() + **"\n"**);  
  
  
 Employee[] EmpArr = **new** Employee[3]; *// here we have defined Array of Employee references*   
 EmpArr[0] = Emp1; *// that points to object of type Employee.*  
 EmpArr[1] = Emp2;  
 EmpArr[2] = Emp3;  
  
 Arrays.sort(EmpArr); *//Sorts the specified array of objects into ascending order, according to the*  
 *// natural ordering of its elements. All elements in the array must implement the Comparable interface.*  
 *// Furthermore, all elements in the array must be mutually comparable (that is, e1.compareTo(e2) must*  
 *// not throw a ClassCastException for any elements e1 and e2 in the array).*  
 *// This sort is guaranteed to be stable: equal elements will not be reordered as a result of the sort.*  
  
 *//System.out.println("List Of Employees Sorted On the Basis of Salary is: \n" + Arrays.toString(EmpArr));*  
 System.out.println(**"List Of Employees Sorted On the Basis of Salary is:"**);  
 **for** (Employee EmpVal : EmpArr) { *// Here this for loop takes one reference variable from the EmpArr and assign*  
 *// it to another reference variable EmpVal of type Employee, and then we use this Ref Variable*  
 *// to fetch a particular object's name and salary*  
 System.out.println(EmpVal.name + **" : "** + EmpVal.salary);  
 }  
 }  
}  
  
**class** Employee **implements** Comparable<Employee> {  
  
 String name;  
 Double salary;  
  
  
 **public** Employee(String name, Double salary) {  
 **this**.name = name;  
 **this**.salary = salary; *// Here this keyword is used to get the variables defined in main class Employee.*  
 }  
  
 @Override  
 **public** String toString() {  
 **return** **this**.name + **", "** + **this**.salary;  
 }  
  
 @Override  
 **public** **int** hashCode() {  
 **return** name.hashCode();  
 }  
  
 @Override  
 **public** **int** compareTo(Employee OtherObject) {  
 **if** (**this**.salary > OtherObject.salary) {  
 **return** 1;  
 } **else** **if** (**this**.salary < OtherObject.salary) {  
 **return** -1;  
 } **else** {  
 **return** 0;  
 }  
 *// we can write the above lines of code / logic In the below format as well*  
 *//return this.salary.compareTo(OtherObject.salary);*  
 }  
  
 @Override  
 **public** **boolean** equals(Object OtherObject) {  
 Employee OtherEmpObject = (Employee) OtherObject; *// here we are type casting the reference variable to its actual object.*  
 **return** Objects.equals(**this**.salary, OtherEmpObject.salary);  
 }  
}

**Text

Description automatically generatedOutput:**

**Practical 41**

**Aim:** WAP to show usage of hash table, hash map and linked hash map.

**Program:**

*/\* Hash Table, Hash Map and Linked Hash Map \*/*  
**import** java.util.HashMap;  
**import** java.util.Hashtable;  
**import** java.util.LinkedHashMap;  
**import** java.util.Set;  
  
**public** **class** HashPrograms {  
 **public** **static** **void** main(String[] args) {  
 Hashtable<String, String> ht = **new** Hashtable<String, String>();  
 HashMap<String, String> hm = **new** HashMap<String, String>();  
 HashMap<Integer, String> myMap = **new** HashMap<Integer, String>();*// Q*  
 LinkedHashMap<String, String> lhm = **new** LinkedHashMap<String, String>();  
 ht.put(**"Microsoft"**, **"Windows 11"**);  
 ht.put(**"IBM"**, **"IBM Quantum"**);  
 ht.put(**"Google"**, **"Fushia OS"**);  
  
 hm.put(**"1"**, **"Tizen OS"**);  
 hm.put(**"Apple"**, **"Mac OS"**);  
 hm.put(**"1"**, **"z/OS"**);  
  
 lhm.put(**"Microsoft"**, **"Windows 11"**);  
 lhm.put(**"IBM"**, **"IBM Quantum"**);  
 lhm.put(**"Google"**, **"Fushia OS"**);  
  
 System.out.println(**"Hash Table:"**);  
 System.out.println(ht);  
  
 System.out.println(**"- - - - - - - - - - - -"**);  
  
 System.out.println(**"Hash Map:"**);  
 System.out.println(hm);  
  
 System.out.println(**"- - - - - - - - - - - - -"**);  
  
 System.out.println(**"Linked Hash Map:"**);  
 System.out.println(lhm);  
  
 System.out.println(**"- - - - - - - - - - "**);  
 System.out.println(**"Hash Table Values:"**);  
 Set<String> keys\_ht = ht.keySet();  
  
 **for** (String s : keys\_ht)  
 System.out.println(ht.get(s));  
  
 System.out.println(**"- - - - - - - - - - "**);  
 System.out.println(**"Hash Map Values:"**);  
 Set<String> keys\_hm = hm.keySet();  
  
 **for** (String s : keys\_hm)  
 System.out.println(hm.get(s));  
  
 System.out.println(**"- - - - - - - - - - "**);  
 System.out.println(**"Linked Hash Map Values:"**);  
 Set<String> keys\_lhm = lhm.keySet();  
  
 **for** (String s : keys\_lhm)  
 System.out.println(lhm.get(s));  
 }  
}

**Text

Description automatically generatedOutput:**

**Practical 42**

**Aim:** WAP to show usage of TreeSet, LinkedHashSet and HashSet.

**Program:**

**import** java.util.HashSet;  
**import** java.util.Iterator;  
**import** java.util.LinkedHashSet;  
**import** java.util.TreeSet;  
  
**public** **class** SetsDemo {  
 **public** **static** **void** main(String[] args) {  
 *// in set we cannot add duplicate values*  
 *// Tree set is sorted set.*  
 *// In tree set The elements are ordered using their natural ordering,*  
 *// or by a Comparator provided at set creation time, depending on which constructor is used.*  
 TreeSet<String> Ts = **new** TreeSet<String>();  
 Ts.add(**"S"**);  
 Ts.add(**"M"**);  
 Ts.add(**"V"**);  
 Ts.add(**"D"**);  
 Ts.add(**"U"**);  
 Ts.add(**"U"**);  
 Ts.add(**"N"**);  
 Ts.add(**"I"**);  
 Ts.add(**"V"**);  
  
 System.out.println(**"Elements in tree set (accessed via Println) are:"** + Ts + **"\n"**);  
  
 System.out.println(**"Elements in tree set (accessed via Iterator interface) are:"**);  
 Iterator<String> itr3 = Ts.iterator();  
 **while** (itr3.hasNext()) {  
 System.out.println(itr3.next());  
 }  
  
 *// Linked Hash Set and Hash Set Both Maintains the Order in which elements are added to the set and also*  
 *// No duplication of Entries are allowed in all three variants of set;*  
 LinkedHashSet<String> Lhs = **new** LinkedHashSet<String>();  
 Lhs.add(**"S"**);  
 Lhs.add(**"M"**);  
 Lhs.add(**"V"**);  
 Lhs.add(**"D"**);  
 Lhs.add(**"U"**);  
 Lhs.add(**"U"**);  
 Lhs.add(**"N"**);  
 Lhs.add(**"I"**);  
 Lhs.add(**"V"**);  
  
 System.out.println(**"Elements in Linked Hash Set (accessed via Println) are:"** + Lhs + **"\n"**);  
  
 System.out.println(**"Elements in Linked Hash set (accessed via Iterator interface) are:"**);  
 Iterator<String> itr4 = Lhs.iterator();  
 **while** (itr4.hasNext()) {  
 System.out.println(itr4.next());  
 }  
  
 HashSet<String> Hs = **new** HashSet<String>();  
 Hs.add(**"S"**);  
 Hs.add(**"M"**);  
 Hs.add(**"V"**);  
 Hs.add(**"D"**);  
 Hs.add(**"U"**);  
 Hs.add(**"U"**);  
 Hs.add(**"N"**);  
 Hs.add(**"I"**);  
 Hs.add(**"V"**);  
  
 System.out.println(**"Elements in Hash Set (accessed via Println) are:"** + Hs + **"\n"**);  
  
 System.out.println(**"Elements in Hash set (accessed via Iterator interface) are:"**);  
 Iterator<String> itr5 = Hs.iterator();  
 **while** (itr5.hasNext()) {  
 System.out.println(itr5.next());  
 }  
  
 }  
}

**Output:**

Text

Description automatically generatedText

Description automatically generated