Java programming video summary
(Abdusami)

What is a computer ?
Computer is a machine that is capable of processing data.

What is a program?
Program is a set of instructions that a computer follows to perform a task.

What is algorithm?
Algorithm is a high level step by step guide to complete a task.

Package name should be all lowercase.
Class name should start with uppercase.
Java does not allow to have spaces in the name of class.

What is Convention in java?
Convention in java is what has been adapted and accepted by java programmers worldwide. Following these conventions makes it easy for programmers to read and edit java code.

In order to run any code from within a class , it has to be inside of a Method.
Main method is a starting point of the program.
Semicolon is used to end a statement in java.

What is variable?
Variable is a memory location that stores the data.variables can store all sorts of data. But we have to declare what kind of data that it should store. Because different data types use different amounts of memory location.
Numbers cannot be very first character for the variable name.

Demonstration of example:

```java
package PracticeS2;

import java.util.Scanner;

public class GrossPayCalculator {

  public static void main(String[] args) {
     //1. Get the number of hours

     System.out.println("Please enter the number of hours which employee worked");
     Scanner scanner = new Scanner(System.in);
     int hours = scanner.nextInt();

     // 2. Get the hourly pay rate

     System.out.println("Please Enter the hourly pay rate of employee");
     double rate = scanner.nextDouble();

     // 3. Multiply hours and pay rate

     double grossPay = hours*rate;

     // Display result

     System.out.println("Gross pay is: " + grossPay);
  }
}
```

```
Please enter the number of hours which employee worked
45
Please Enter the hourly pay rate of employee
15
Gross pay is: 675.0

Process finished with exit code 0
```

Assignment from Video:

```
package PracticeS2;

import java.util.Scanner;

public class CoffeeOfWholeYear {

  public static void main(String[] args) {
     //1. Get which season we are
    System.out.println("Please enter which season we are on");
     Scanner scanner = new Scanner(System.in);
     String season = scanner.next();
     // 2. Ask for whole number of coffee
     System.out.println("Please enter whole number of coffee you drink");
     int numberOfCoffee = scanner.nextInt();
     //3. Ask for adjective about weather
     System.out.println("how was the weather that day?");
     String adjective = scanner.next();
     // 4. Display result

     System.out.print("On a " +adjective +" " + season + " day, " +"I drink of minimum of " +
numberOfCoffee + " cups of coffee." );


  }
}
```

Output:

```
Please enter which season we are on
summer
Please enter whole number of coffee you drink
45
how was the weather that day?
hot
On a hot summer day, I drink of minimum of 45 cups of coffee.
Process finished with exit code 0
```

If statement is ‘’if certain situation occurs do <something> then come back to the main flow’’.

```
package chapter3;
import java.util.Scanner;
public class SalaryCalculator {
   public static void main(String[] args) {
       /* every salesperson can receive a paycheck of 1000$ a week no matter what, if
salesperson who exceeds 10 sales
       get an additional 250$
        */

       // Initialize known value
       int weeklySalary = 1000;
       int bonus = 250;
       int quota = 10;

       //Get values for unknown
       System.out.println("Please enter the number of sales which employee made this week
");
       Scanner scanner = new Scanner(System.in);
       int sales = scanner.nextInt();
       scanner.close();

       // quick detour for the bonus earner
       if(sales>quota)
         weeklySalary = weeklySalary +bonus
        // result
 System.out.print("this week this salesperson salary is : " + weeklySalary + "$");

   }
}
```

```
Output:
 Please enter the number of sales which employee made this week
12
this week this salesperson salary is : 1250$
Process finished with exit code 0
Please enter the number of sales which employee made this week
5
this week this salesperson salary is : 1000$
```

```
Process finished with exit code 0
```

If else statement

If a certain situation occurs , <do something> , otherwise <do something else>

```
package chapter3;
/* If else
all salespeople are expected to make at least 10 sales a week.
for  those who do, they receive a congratulatory message.
for those who don't, they are informed of how many sales they were short.
*/

import java.util.Scanner;
public class QuotaCalculator {
  public static void main(String[] args) {
     //Initialize known value
     int quota = 10;
     //Get unknown value
     System.out.println("Please enter how many sales employee made this week :");
     Scanner scanner = new Scanner(System.in);
     int realSales = scanner.nextInt();
     scanner.close();
     //Make a decision on the path to take- output
     if (realSales >= quota) {
        System.out.println("Congrats, you have met your quota");
     } else {
        int salesShort = quota - realSales;
        System.out.println("you did not make your quota,You were " + salesShort + " short");
     }
  }
}
```

```
Please enter how many sales employee made this week :
11
Congrats, you have met your quota
Process finished with exit code 0
Second run:
Please enter how many sales employee made this week :
7
you did not make your quota,You were 3 short
Process finished with exit code 0
```

Nested if statements are path inside of the path,if a certain condition is met, it goes inside of the if block and is immediately faced with another if statement.

```
package chapter3;
import java.util.Scanner;

/*
NASTED IFS
to qualify a loan ,a person must make at least 30000$ a year.
and have been working at their current job for at least 2 years.
*/
public class LoanQualifier {
  public static void main(String[] args) {

      //Initialize known value
      int requiredSalary = 30000;
      int requiredYears = 2;

      //Get Unknown value
      System.out.println("Enter your salary");
      Scanner scanner = new Scanner(System.in);
      double realSalary = scanner.nextDouble();
      System.out.println("How many years you worked");
      double workedYears = scanner.nextDouble();

      // Make decision
      if (realSalary >= requiredSalary) {
         if (workedYears >= requiredYears) {
            System.out.println("Congrats,You've met the requirements");
         } else {
            System.out.println("Sorry,you have to work at least " + requiredYears + " years");
         }
      } else {
         System.out.println("Sorry, you must earn at least  " + realSalary + "$ to qualify for a
loan");
      }

  }
}
```

Outputs:

```
Enter your salary
35000
How many years you worked
4
Congrats,You've met the requirements
```

Second run:

```
Enter your salary
25000
How many years you worked
3
Sorry, you must earn at least  25000.0$ to qualify for a loan
```

Third run:

```
Enter your salary
45000
How many years you worked
1
Sorry,you have to work at least 2 years
```

If-else-if statement

If situation A occurs <do something>,Else if situation B occurs <do something else>,Else if situation C occurs <do something else>.

```
package chapter3;
//Determine the letter grade based on scores
import java.util.Scanner;
public class TestResults {
  public static void main(String[] args) {
       // Get the test score
       System.out.println("Enter your test score");
       Scanner scanner = new Scanner(System.in);
       double score = scanner.nextDouble();
       // Determine the letter grade
       char grade;
       if (score < 60) {
          grade = 'F';
       } else if (score <= 70) {
          grade = 'D';
       } else if (score <= 80) {
          grade = 'C';
       } else if (score <= 90) {
          grade = 'B';
       } else {
          grade = 'A';
       }
       System.out.println("Your grade is : " + grade);
  }
}
```

First scenario:

```
Enter your test score
96
Your grade is : A
```

Second scenario:

```
Enter your test score
54
Your grade is : F
```

Third scenario:

```
Enter your test score
```

```
76
Your grade is : C
```

The Switch statement

If situation A occurs <do something>,Else if situation B occurs <do something else>,Else if situation C occurs <do something else>.
The difference between **Switch statement** and **If-Else-If statement** is If-Else-If statement checks the condition to be true,whereas Switch statement checks for equality.
Switch statement is case sensitive.

```java
package chapter3;
import java.util.Scanner;
public class GradeMessage {
  public static void main(String[] args) {
     System.out.println("Enter your letter grade:");
     Scanner scanner = new Scanner(System.in);
     String grade = scanner.next();
     String message;
     switch (grade) {
        case "A":
           message = "Excellent job!";
           break;
        case "B":
           message = "Great job!";
           break;
        case "C":
           message = "Good job";
           break;
        case "D":
           message = "You need to work a bit harder";
           break;
        case "F":
           message = " Oh !";
           break;
        default:
           message = "Error, invalid grade";
           break;
     }

     System.out.println(message);
  }
}
```

First scenario:

```
Enter your letter grade:
R
Error, invalid grade
```

Second scenario:

```
Enter your letter grade:
C
Good job
```

Third scenario:

```
Enter your letter grade:
F
 Oh !
Process finished with exit code 0
```

Relational Operator and Logical Operator

Java has relational operators to compare two numbers and return a boolean value. The relational operators are:
 > greater than, < less than, >= greater than or equal to, <= less than or equal to,== equal to, != not equal to.
When java compare two Strings , it compares memory locations of Strings, not value.
Compare to two Strings we use the equals method of String like below example.

```
String name1 = "Tom";
String name2 = "tom";
if(name1.equals(name2))
If we don't care about cases method like this.
if(name1.equalsIgnoreCase(name2))
```

Logical Operator

Java has logical operators to get boolean result either true or false.
And operator ----> && to be true both of conditions must be true
Or operator   ----> || to be true at least one condition must be true
Not operator -----> !   to be true condition must be false

Example which is used Logical operator:

```java
/*
Logical Operator
to qualify a loan ,a person must make at least 30000$ a year.
and have been working at their current job for at least 2 years.
*/
public class LogicalOperatorLoanQualifier {
   public static void main(String[] args) {

       //Initialize known value
       int requiredSalary = 30000;
       int requiredYears = 2;

       //Get Unknown value
       System.out.println("Enter your salary");
       Scanner scanner = new Scanner(System.in);
       double realSalary = scanner.nextDouble();
       System.out.println("How many years you worked");
       double workedYears = scanner.nextDouble();

       // Make decision
       if (realSalary >= requiredSalary && workedYears >= requiredYears)
           System.out.println("Congrats,You've met the requirements");

       else
           System.out.println("Sorry, you must earn at least $" + requiredSalary+ " and at least
work 2 years" + " to qualify for a loan");
   }
}
```

Output:

```
Enter your salary
50000
How many years you worked
1
Sorry, you must earn at least $30000 and at least work 2 years to qualify for a loan

Process finished with exit code 0
```

loops
Java has three different kind of loop which are while loop, do while loop, for loop.
What is loop in java?
Loops are structures that cause a block of code to repeat.
Advantage of using loops is to eliminate redundant act of copying the same statement again over again.
While loop continues running while the specified condition remains true.

```
package chapter4;
import java.util.Scanner;
public class WhileLoopGrossPayValidation {
  public static void main(String[] args) {
      /* While Loop
      Each store employee makes 15$ per hour, and They are not allowed to work over 40
hours. Check no one did overtime by writing a code  */
      int hourlysalary = 15;
      int maximumHoursAllowed = 40;
      // Unknown values
      System.out.println("how many hours did you work this week");
      Scanner scanner = new Scanner(System.in);
      int realWorkedHours = scanner.nextInt();

      while (realWorkedHours > maximumHoursAllowed) {
         System.out.println("you exceeded work hours you are allowed!! ");
         realWorkedHours = scanner.nextInt();
      }
      scanner.close();
      //Calculate the gross
      int grossPay = hourlysalary * realWorkedHours;
      System.out.println(grossPay);
  }
}
```

Output :

```
how many hours did you work this week
87
you exceeded work hours you are allowed!!
65
you exceeded work hours you are allowed!!
23
```

```
Gross pay is : 345
Process finished with exit code 0
```

do while loop

Do while loop is similar to the while loop except that the condition is checked after the statements are checked. do while loop guarantees the loop execution at least once.

```java
package chapter4;
import java.util.Scanner;
/* write a program that user allows a user can input two number and add two number.the
user should be able to repeat this action until they indicate they are done.
*/
public class AddNumbersDoWhileLoop {
  public static void main(String[] args) {
     Scanner scanner = new Scanner(System.in);
     boolean yes;
     do {
        System.out.println("Enter the First number");
        double firstNum = scanner.nextDouble();
        System.out.println("Enter the Second number");
        double secondNum = scanner.nextDouble();
        double sum = firstNum + secondNum;
        System.out.println("sum of two numbers is : " + sum);
        System.out.println("would you like to start over? ");
        yes = scanner.nextBoolean();
     }
     while (yes);
     scanner.close();
  }
}
```

Output:

```
Enter the First number
65
Enter the Second number
33
sum of two numbers is : 98.0
would you like to start over?
true
Enter the First number
43
Enter the Second number
12
```

```
sum of two numbers is : 55.0
would you like to start over?
False           Process finished with exit code 0
```

For loop

For loop has three parts which are initialization, condition and updater. It will be running until the condition becomes false.

```
package chapter4;
import java.util.Scanner;
public class Cashier {
   public static void main(String[] args) {
       //write a program that a cashier will scan a given number of items, and tell the total
price
      // get the number of items to scan
       System.out.println("Please enter the number of items would you like to scan");
       Scanner scanner = new Scanner(System.in);
       int quantity = scanner.nextInt();
       double total = 0;
       //Create a loop to iterate through all of the items and accumulate the price
       for(int i = 0; i<quantity; i++) {
           System.out.println("Please enter the price of item");
           double price = scanner.nextDouble();
           total=total+price;
       }
       scanner.close();
       System.out.println("total is :" + total);
   }
}
```

Output:

```
Please enter the number of items would you like to scan
6
Please enter the price of item
5.78
Please enter the price of item
5
Please enter the price of item
23
Please enter the price of item
34
Please enter the price of item
45
Please enter the price of item
```

```
23
total is :135.78
Process finished with exit code 0
```

For loop runs specified number of time.

```java
package chapter4;
import java.util.Scanner;
// check String contains letter A
public class LetterSearch {
  public static void main(String[] args) {
      //get text
      System.out.println("Please Enter a letter");
      Scanner scanner = new Scanner(System.in);
      String text = scanner.next();
      scanner.close();
      boolean letterFound = false;
      // search letter A
      for (int i = 0; i < text.length(); i++) {
         char currentChar = text.charAt(i);
         if (currentChar == 'A' || currentChar == 'a') {
            letterFound = true;
            break;
         }
      }

      if (letterFound)
         System.out.println("Text contains letter 'A' ");
      else
         System.out.println("NOT FOUND LETTER A");
  }
}
```

First run:

```
Please Enter a letter
yfhfsjg
NOT FOUND LETTER A
Process finished with exit code 0
```

Second run:

```
Please Enter a letter
gytguau
```

```
Text contains letter 'A'
Process finished with exit code 0
```

Nested Loop

Nested loop is basically loop inside of loop.

```java
package chapter4;

import java.util.Scanner;

// Find the average of each students test scores

public class AverageTestScores {
  public static void main(String[] args) {
      //Initialize what we know
      int numberOfStudents = 24;
      int numberOfTests = 4;
      Scanner scanner = new Scanner(System.in);
      // process for all students
      for (int i = 0; i < numberOfStudents; i++) {

         double total = 0;
         for (int j = 0; j < numberOfTests; j++) {
            System.out.println("Please enter test score for test # " + (j + 1));
            double score = scanner.nextDouble();
            total = total + score;
         }
         double average = total / numberOfTests;
         System.out.println("the test average for student #" + (i + 1) + " is: " + average);
      }
      scanner.close();
  }
}
```

```
Please enter test score for test # 1
89
Please enter test score for test # 2
38
Please enter test score for test # 3
97
```

```
Please enter test score for test # 4
94
the test average for student #1 is: 79.5
```

Method

Method is subtask in a class, it can eliminate redundancy of code.
Method has access modifier.
Access modifier indicates who can access to use this method. We have public,private,protected and default access modifier.
Having access modifier is not required if one is not specified. If we don't mention any access modifier, that method can be used by any classes within the same package.
Next part of header is a non access modifier this includes static,final,abstract and synchronized.the next is return type this indicates data type of value that this method will be returned. Every method is required to specify return type.
Return type of main method is void that means return nothing but perform it .
Main method drives the flow.
After return type there is method name. Every method is required to have a name.by convention method name will begin with a verb and lowercase.if return type is boolean , we have to name it as a question. After the method name comes a set of parentheses.
Inside of parentheses can be empty. Parentheses are used to hold a list of data(parameters) that should be supplied to this method.
 After method header comes method body within a set of curly braces. The body consists of zero or more statements to be executed in this method. If the body specified a return type anything other than void,it will return that specified return type.the method has to include a return statement.
Example:

```
package chapterFive;
import java.util.Scanner;
// write a method that asks a user for their name and then greets by name
public class Greeting {
  public static void main(String[] args) {
     System.out.println("Please enter your name");
     Scanner scanner = new Scanner(System.in);
     String name = scanner.next();
     greetUse(name);
  }
  public static void greetUse(String name) {
     System.out.println("Hi there, " + name);
  }
}
```

```
Please enter your name
sami
Hi there, sami
Process finished with exit code 0
```

Scope of variable

A variable is only available within the curly braces that it was declared within. In order to use these variables outside of curly braces, we have to either pass them to the methods that they need or declare them at a higher scope(class level). Then every single method can access these variables.

Variables defined within a method are called local variables. Variables that are not defined within a local scope are called global variables.

Example:

```
package chapterFive;
import java.util.Scanner;
/* write a program that approves anyone who makes more than 25000$ and credit score is
greater than 700,reject all others.
*/
public class InstantCreditCheck {
  static double requiredSalary = 25000;
  static int requiredCreditScore = 700;
  static Scanner scanner = new Scanner(System.in);
  public static void main(String[] args) {
     double salary = getSalary();
     int creditScore = getCreditScore();
     scanner.close();
     boolean qualified = isQualified(creditScore,salary);
     notifyUser(qualified);
  }
  public static double getSalary() {
     System.out.println("Please enter your salary");
     double salary = scanner.nextDouble();
     return salary;  }
  public static int getCreditScore() {
  System.out.println("please enter your credit score");
     int creditScore = scanner.nextInt();
     return creditScore; }
  public static boolean isQualified(int creditScore, double salary) {
     if (creditScore >= requiredCreditScore && salary >= requiredSalary) {
        return true;
}
     else {
        return false;
}
  }
 public static void notifyUser(boolean isQualified){
     if(isQualified) {
        System.out.println("Congrats,you have been qualified!!");
     }
     else {
        System.out.println("Sorry, you have been declined..");
     }
  }
}
```

```
Output:
First Scenario:
Please enter your salary
67000
please enter your credit score
740
Congrats,you have been qualified!!

Process finished with exit code 0
Second Scenario:
Please enter your salary
22000
please enter your credit score
670
Sorry, you have been declined..

Process finished with exit code 0
```

Classes and Objects

Java is an object oriented programming language.objects are structures which contain data in the form of fields and methods. These fields and methods can be utilized in other classes by creating an object. Class is a blueprint is meant to be generalized shell.all objects have a constructor.Constructor name must be exactly the same name as class. we have default constructor that can be used to assign values to the fields. We can have as many constructor we want. Classes should practice encapsulation which is a key principle of object oriented programming. When we create a class that is going to represent an object and field should be private. Private means is that other code outside of this class can’t access.

```
package chapter6;
public class Rectangle {
   private double length;
   private double width;
   public Rectangle(double length, double width) {
       this.length = length;
       this.width = width;
   }
   public double getLength() {
       return length;  }
   public void setLength(double length) {
       this.length = length;  }
   public double getWidth() {
       return width;  }
   public void setWidth(double width) {
       this.width = width; }
  public double calculatePerimeter() {
       return (2*length)+(2*width);
  }
   public double calculateArea() {
       return length*width;
   }
}
package chapter6;
write a class that creates instances of the Rectangle class to find
the total area of two rooms in a house.
public class HomeAreaCalculator {
   public static void main(String[] args) {
       //Create an instance of Rectangle class
       Rectangle room1 = new Rectangle(25, 40);
       double areaOfRoom1 = room1.calculateArea();
       // instance for Second room
       Rectangle room2 = new Rectangle(24, 50);
       double areaOfRoom2 = room2.calculateArea();
       double totalArea = areaOfRoom1 + areaOfRoom2;
       System.out.println("Total area of two rooms is : " + totalArea);
```

```
 }
}
```

Output : Total area of two rooms is : 2200.0

```
package chapter6;

import java.util.Scanner;

public class HomeAreaCalculatorRedo {
  Scanner scanner = new Scanner(System.in);

  public static void main(String[] args) {

     HomeAreaCalculatorRedo areaCalculator = new HomeAreaCalculatorRedo();
     Rectangle kitchen = areaCalculator.getRoom();
     Rectangle bathRoom = areaCalculator.getRoom();
     double area = areaCalculator.calculateTotalArea(kitchen, bathRoom);
     System.out.println("the total area is " + area);
     areaCalculator.scanner.close();
  }
  public Rectangle getRoom() {

     System.out.println("Please enter length of room");
     double length = scanner.nextDouble();
     System.out.println("Please enter width of room ");
     double width = scanner.nextDouble();
     return new Rectangle(length, width);
  }
  public double calculateTotalArea(Rectangle rectangle1, Rectangle rectangle2) {
     return rectangle1.calculateArea() + rectangle2.calculateArea();
  }
}
```

```
Please enter length of room
25
Please enter width of room
40
Please enter length of room
34
Please enter width of room
```

```
20
the total area is 1680.0
Process finished with exit code 0
```

Overloading methods is within a class we can have multiple methods that have the same name but different signatures.

```java
package chapter6;
public class Month {
   public static String getMonth(int month) {
       switch (month){
           case 1: return "January";
           case 2: return "February";
           case 3: return "March";
           case 4: return "April";
           case 5: return "May";
           case 6: return "June";
           case 7: return "July";
           case 8: return "August";
           case 9: return "September";
           case 10: return "October";
           case 11: return "November";
           case 12: return "December";
           default:return "invalid input, please enter number between 1 and 12";
       }
   }
   public static int getMonth(String month){
       switch (month){
           case "January": return 1;
           case "February": return 2;
           case "March": return 3;
           case "April": return 4;
           case "May" : return 5;
           case "June":return 6;
           case "July": return 7;
           case "August" : return 8;
           case  "September": return 9;
           case "October": return 10;
           case "November": return 11;
           case  "December": return 12;
           default:return 0;
```

```java
        }
    }
}
```

```java
package chapter6;

public class MonthConverter {
    public static void main(String[] args) {
        System.out.println(Month.getMonth(7));
        System.out.println(Month.getMonth("November"));
    }
}
```

```
July
11
Process finished with exit code 0
```

Arrays

Arrays are special objects or are containers which can hold multiple values.
Syntax of declaration of an array :
data type +[ ] +arrayname = new +data type [length of an array];
Index starts from zero.index zero represents the first element of the array.
If we have the value of array we can initialize like
Data type + [ ] +arrayname = {all values of array contains};

```java
package Chapter7;
import java.util.Random;
public class LotteryTicket {
   public static final int LENGTH = 6;
   public static final int MAX_TICKET_NUMBER = 69;

   public static void main(String[] args) {
       int [] ticket = generateNumbers();
       printTheTicket(ticket);
   }
   public static int[] generateNumbers(){
       int[] ticket = new int[LENGTH];
       Random random = new Random();
       for(int i =0; i<LENGTH;i++) {
           ticket[i]=random.nextInt(MAX_TICKET_NUMBER);
       }
       return ticket;
   }
   public static void printTheTicket(int ticket[]) {
       for(int i =0; i<LENGTH;i++) {
           System.out.print(ticket[i] + " | ");
       }

   }
}
```

```
14 | 16 | 12 | 24 | 9 | 32 |
Process finished with exit code 0
```

Array search has two types which are Sequential search and Binary search.
Sequential search look for in order and if we search within several thousands element it is not efficient at all. Binary search works like eliminating half of array for what needs to be searched until we find the value we are looking for. Binary search is much quicker than sequential search. Using **Arrays.binarySearch** method we can search the element quicker.

```java
package Chapter7;
import java.util.Arrays;
import java.util.Random;
public class SearchLotteryTicketNumber {
   public static final int LENGTH = 6;
   public static final int MAX_TICKET_NUMBER = 69;
   public static void main(String[] args) {
      int[] ticket = generateNumbers();
      Arrays.sort(ticket);
      printTheTicket(ticket);  }
   public static int[] generateNumbers() {
      int[] ticket = new int[LENGTH];
      Random random = new Random();
      for (int i = 0; i < LENGTH; i++) {
         int randomNumber;
         do {
            randomNumber = random.nextInt(MAX_TICKET_NUMBER) + 1;
         } while (search(ticket, randomNumber));
         ticket[i] = randomNumber;  }
      return ticket;  }
   public static void printTheTicket(int ticket[]) {
      for (int i = 0; i < LENGTH; i++) {
         System.out.print(ticket[i] + " | ");
      }
   }

   /**
    * this does sequential search on the array to find a value
    *
    * @param array            array to search through
    * @param numberToSearchFor value to search for
```

```
     * @return true if found false if not
     */


 public static boolean search(int[] array, int numberToSearchFor) {
        for (int value : array) {
            if (value == numberToSearchFor) {
                return true;
            }

        }
        return false;
    }
    public static boolean binarySearch(int[] array, int numberToSearchFor) {
        Arrays.sort(array);
        int index = Arrays.binarySearch(array,numberToSearchFor);
        if(index>=0) {
            return true;
        }
        else
            return false;
    }
}
4 | 21 | 45 | 53 | 61 | 67 |
Process finished with exit code 0
```

Get sum, average,maximum and minimum number examples :

```
package Chapter7;
import java.util.Scanner;
public class Grades {
  private static int grades[];
  private static Scanner scanner = new Scanner(System.in);
  public static void main(String[] args) {
     System.out.println("How many grades do you want to enter");
     grades = new int[scanner.nextInt()];
     getGrades();
     System.out.println("SUM is: " + calculateSum());
     System.out.println("Average is " + String.format("%.2f", average()));
     System.out.println("Maximum number is :" + getHighest());
     System.out.println("Lowest number is :" + getlowest());
  }

  public static void getGrades() {
     for (int i = 0; i < grades.length; i++) {
        System.out.println(" Enter grade# " + (i + 1));
        grades[i] = scanner.nextInt();
     }
  }

  public static int calculateSum() {
     int sum = 0;
     for (int grade : grades) {
        sum = sum + grade;
     }
     return sum;
  }
  public static double average() {
     return calculateSum() / grades.length;
  }
  public static int getHighest() {
     int highest = 0;
     for (int grade : grades) {
        if (grade > highest) {
           highest = grade;
```

```
            }
        }
        return highest;
    }
    public static int getlowest() {
        int lowest = 0;
        for (int grade : grades) {
            if (grade < lowest) {
                lowest = grade;
            }
        }
        return lowest;
    }
}
```

```
How many grades do you want to enter
4
 Enter grade# 1
77
 Enter grade# 2
45
 Enter grade# 3
34
 Enter grade# 4
99
SUM is: 255
Average is 63.00
Maximum number is :99
Lowest number is :0

Process finished with exit code 0
```

Declaration of multidimensional array:

Data type+[ ] [ ] arrayname =new arrayname[row number][column number];

Data types

Programming languages are either dynamically typed or statically typed.dynamically typed programming language determine the data type of a variable at runtime.
Java is statically typed programming language which means it expects its variables to be declared before they can be assigned values. Because in statically typed programming language data types are checked at compile time. We can not use var variable as a global variable. Var variable is used only as local variable.
Primitive data types are provided by java. There are byte,short,int,double,long,float,char and boolean.
We have also wrapper class like Integer,Double,Short Character, etc.... There are a lot of methods available in wrapper class.

String

String is an object. It is a sequence of characters.
In String class , there are some methods we use quite often. Example:
charAt(int index), contains,equals(), equalsIgnoreCase(),indexOf(),isEmpty(),replaceAll(), split(),toUpperCase(),Trim().

```
package chapter8;
/**
* Split an string into an array by tokenizing it.
* counts words and print them
*/
public class TextProcessor {
  public static void main(String[] args) {
     countWords("I love Test Automation University ");   }
  public static void countWords(String text){
     var words = text.split(" ");
     int numberOfWords = words.length;
     String message = String.format("your text contains %d words: ",numberOfWords);
     System.out.println(message);
     for(int i = 0; i< numberOfWords; i++) {
        System.out.println(words[i]);
     }
  }
}
```

```
your text contains 5 words:
I
love
Test
Automation
```

```
University
Process finished with exit code 0
```

```java
package chapter8;

/**
* Split an string into an array by tokenizing it.
* counts words and print them
*/
public class TextProcessor {
  public static void main(String[] args) {
     reverseString("camel");
  }
  public static void reverseString(String text) {
     for (int i = text.length() - 1; i >= 0; i--) {
        System.out.print(text.charAt(i));
     }
  }
}
```

```
lemac
Process finished with exit code
```

Example:

```java
package chapter8;
/**add space before uppercase letter
*/
public class addSpaces {
  public static void main(String[] args) {
     addSpaces("HeyIt'sMeSami");
  }
  public static void addSpaces(String text) {
     var modifiedText = new StringBuilder(text);
     for (int i = 0; i < modifiedText.length(); i++) {
        if (i != 0 && Character.isUpperCase(modifiedText.charAt(i))) {
           modifiedText.insert(i, " ");
           i++;   }
     }
     System.out.print(modifiedText);
  }
}
```

```
Hey It's Me Sami
```

Process finished with exit code 0

Inheritance
There are four major principles of Object oriented programming, they are Encapsulation,Inheritance, Abstraction and Polymorphism.
Inheritance is one class become an extension of another class,therefore inheriting the members of that class. In inheritance there are two parts which are parent class and child class.parent class is known as super class sometimes called as base class.. Child class is known as subclass.sometimes refer to as derived class. Child class inheritances members of parent class. This allows classes reuse data that already exist within another class.
Super class constructor must be in first line in subclass constructors.

Inheritance example:
Parent class:

```
package chapter9;
public class Person {
  private String name;
  private String gender;
  private int age;
  public Person() {
     System.out.println("In Person default constructor");
  }
  public Person(String name) {
     System.out.println("Second constructor name is sami");
  }
  public String getName() {
     return name;  }
  public void setName(String name) {
     this.name = name;}
  public String getGender() {
     return gender; }
  public void setGender(String gender) {
     this.gender = gender;  }
  public int getAge() {
     return age;
  }
  public void setAge(int age) {
     this.age = age;
  }
}
```

Child class:

```
package chapter9;

public class Employee extends Person {
   String employeeID;
   String title;

 public Employee() {
       super("Sami");
       System.out.println("In employee default constructor");
   }
   public String getEmployeeID() {
       return employeeID;
   }
   public void setEmployeeID(String employeeID) {
       this.employeeID = employeeID;
   }
   public String getTitle() {
       return title;
   }
   public void setTitle(String title) {
       this.title = title;
   }
}
```

Testing class:

```
package chapter9;
public class InheritanceTester {
   public static void main(String[] args) {
       Employee employee = new Employee();
   }
}
```

```
Second constructor name is sami
In employee default constructor
Process finished with exit code 0
```

Superclass constructor is run before the subclass constructor.
Explicit call can be made to superclass's constructors from one of the subclass's constructor by using super.
Super calls must be first.

If the superclass does not have a default constructor,the subclass must explicitly call one of its other constructor.

Overriding and overloading method

If subclass may want to change the functionality of a method that it inherited,this is allowed by overriding the inherited method.

Overriding coding example:

```
package chapter9;
public class Rectangle {
   protected double length;
   protected double width;
   protected double sides=4;

 public double getLength() {
       return length;
   }
   public void setLength(double length) {
       this.length = length;
   }
   public double getWidth() {
       return width;
   }
   public void setWidth(double width) {
       this.width = width;
   }
   public double getSides() {
       return sides;
   }
   public void setSides(double sides) {
       this.sides = sides;
   }
   public double calculatePerimeter(){
       return (2*length)+(2*width);
   }
}

package chapter9;
public class Square extends Rectangle {
   @Override
   public double calculatePerimeter() {
       return sides * length;
```

```
  }
}
```

Overloading methods is when we have two exactly the same but with different parameter lists
When dealing with a subclass we can overload a method that we have inherited from a superclass,even though that method lives in another class.

Limiting access in Inheritance

When subclass inherits from a superclass not everything inherited. For example the constructors are not inherited. All the public and protected methods and fields and a superclass those are indeed inherited. But private methods and fields are not inherited.
If any final method, it can be inherited but not can be overridden.
Chain of inheritance: superclass can also inherit from another class.

Polymorphism

Polymorphism is the ability to take multiple forms specifically in Object oriented programming, polymorphism is where a subclass can define their own unique behaviors and share some of the same behaviors of their super class.

Example:

```
package chapter10;
public class Animal {
  public void makeSound() {
     System.out.println("Unknown Animal sound");
  }
}


package chapter10;
public class Dog extends Animal {
  @Override
  public void makeSound() {
     System.out.println("woow!!!");
  }

  public void fetch() {
     System.out.println("fetch is fun");
  }
}
```

```java
package chapter10;

public class Cat extends Animal {
   public void makeSound() {
      System.out.println("meow!!!");
   }

   public void scratch() {
      System.out.println("I am a cat , I scratch things");
   }
}


package chapter10;

public class Zoo {
   public static void main(String[] args) {
      Dog rocky = new Dog();
      rocky.fetch();
      feed(rocky);
      rocky.makeSound();
      Animal sasha = new Dog();
      sasha.makeSound();
      sasha = new Cat();
      feed(sasha);
      sasha.makeSound();
      ((Cat) sasha).scratch();
   }

   public static void feed(Animal animal) {
      if (animal instanceof Dog) {
         System.out.println("THIS IS A DOG FOOD");
      } else if (animal instanceof Cat) {
         System.out.println("THIS is a cat food");
      }

   }
}
```

Output:

```
fetch is fun
THIS IS A DOG FOOD
woow!!!
woow!!!
THIS is a cat food
meow!!!
I am a cat , I scratch things

Process finished with exit code 0
```

An object can have a superclass type, and subclass instance.
Polymorphic object can only access the methods of their type,not their instance.
Casting is required in order to access the methods of their instance.
If a method overridden by the subclass, the polymorphic object will execute the overridden method at runtime.
The instance operator is used to determine if an object is an instance of a certain class.

Abstraction
Abstraction is defined as something that exists in theory,but does not have a concrete existence. Abstract method has no body only the signature of the method is defined and it is not designed to be run as is designed to be overridden by a subclass. Abstract class is not designed to be instantiated ,it is designed to be extended.

Example:

```
package chapter11;

public abstract class Shape {
  abstract double calculateArea();

  public void print() {
     System.out.println("I am a shape");
  }
}
```

```java
package chapter11;

public class Rectangle extends Shape {
  private double length;
  private double width;

  public double getLength() {
      return length;
  }

  public void setLength(double length) {
      this.length = length;
  }

  public double getWidth() {
      return width;
  }

  public void setWidth(double width) {
      this.width = width;
  }

  public Rectangle(double length, double width) {
      setLength(length);
      setWidth(width);

  }

  @Override
  double calculateArea() {
      return length*width;
  }
}
```

```
package chapter11;

public class ShapeTester {
  public static void main(String[] args) {

    Shape rectangle = new Rectangle(5,4);
    rectangle.print();
    System.out.println(rectangle.calculateArea());
  }
}
I am a shape
20.0
Process finished with exit code 0
```

Abstract classes and methods are templates that are meant to be implemented by their subclasses.
Classes and methods are declared abstract by using the abstract keyword.
If a subclass extends from an abstract class, it must implement its abstract methods or be declared abstract itself.
Abstract class cannot be instantiated. It is only to be used as a superclass. If even one method in class is abstract ,the class must be declared as abstract as well.

Interface
All of the methods of an Interface are abstract by nature.there is no need to declare the methods as abstract. Interfaces are implemented not extended. Any class that implements an Interface must implement all of its methods or it must declare itself as an abstract class.
Interfaces can not be instantiated.
One class can implement multiple Interfaces.
Example:

```
package chapter11;
public interface Product {
  double getPrice();
  void setPrice(double price);
  String getName();
  void setName(String name);
  String getColor();
  void setColor(String color);
  default String gerBarcode() {
      return "no barcode"; }
}

package chapter11;

public class Book implements Product {
  private String name;
  private String color;
  private String author;
  private double price;
  private int pages;
  @Override
  public String getName() {
      return name; }
  @Override
  public void setName(String name) {
      this.name = name; }
  @Override
  public String getColor() {
      return color;   }
  @Override
  public void setColor(String color) {
      this.color = color; }
  public String getAuthor() {
      return author;  }
  public void setAuthor(String author) {
      this.author = author; }
  @Override
  public double getPrice() {
      return price; }
  @Override
  public void setPrice(double price) {
      this.price = price; }
  public int getPages() {
      return pages;  }
  public void setPages(int pages) {
      this.pages = pages;
  }
}
```

```
package chapter11;

public class Customer {
  public static void main(String[] args) {
      Product book = new Book();
      book.setPrice(99.99);
      book.setName("New life and new challenge");
      book.setColor("White");
      System.out.println(book.getName());
      System.out.println(book.getPrice() + "$");
      System.out.println("COLOR OF THE BOOK IS: " + book.getColor());

  }
}
```

```
New life and new challenge
99.99$
COLOR OF THE BOOK IS: White

Process finished with exit code 0
```

Collections

A collection is an object that holds references to other objects these are known as data structures. Java provides a collections framework which consists of Interfaces, Classes,and Methods to store and manipulate,aggregate data.
Set is a collection can not contain duplicate elements.
List is a collection where the elements are ordered. List can contain duplicate elements.
Queue is a collection of ordered elements which is typically used to hold items that are going to be processed in someway.
Map is not inherited from collection Interface. However it contains collection like operations which enable them to be used as collections.maps are used to hold key-value pairs.
Set has hashSet, LinkedhashSet, treeSet.
List has ArrayList, LinkedList,Stack and vector.
Most common implementation of queue is linkedList and PriorityQueue.
Queue follows the first in first out algorithm which means elements are processed in the order in which they are entered.

We can access an element in a map by its key. Popular implementations of map has HashMap,TreeMap and LinkedHashMap.

Map has a put Method which is used to add the elements. A put Method takes two arguments a key and a value.
List.of Method uses to add elements.

Looping through collections and maps

There are several ways to loop through collections.
Examples:

```
package chapter12;
import java.util.HashSet;
import java.util.Set;
public class LoopCollectionDemo {
  public static void main(String[] args) {
     setDemo();
  }
  public static void setDemo() {
     Set<String> fruit = new HashSet();
     fruit.add("Apple");
     fruit.add("lemon");
     fruit.add("banana");
     fruit.add("orange");
     fruit.add("lemon");
     // while loop
     var i = fruit.iterator();
     while (i.hasNext()) {
        System.out.println(i.next());
        // foreach loop
        for (String item : fruit) {
           System.out.print(item + " ,");
           // lambda expression
           fruit.forEach(x -> System.out.println(x));
           fruit.forEach(System.out::println);  }
     }
  }
}
```

```
banana
orange
Apple
```

lemon
Process finished with exit code 0

Example for map:

```
package chapter12;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class LoopCollectionDemo {
   public static void main(String[] args) {
       mapDemo();
   }
     public static void mapDemo() {
       Map <String,Integer>fruitCalories = new HashMap();
       fruitCalories.put("apple",90);
       fruitCalories.put("banana",120);
       fruitCalories.put("lemon",80);
       fruitCalories.put("orange",105);
       fruitCalories.put("lemon",95);
       for (var items:fruitCalories.entrySet()) {
           System.out.println(items.getValue());
fruitCalories.forEach((K,V)-> System.out.println("Fruit : " + K + "\nCalories: " + V));
       }
   }
}
```

Output for foreach loop:

```
120
105
90
95
```

Output for lambda expression:

```
Fruit : banana
Calories: 120
Fruit : orange
Calories: 105
Fruit : apple
Calories: 90
Fruit : lemon
Calories: 95
```

Process finished with exit code 0

Exceptions
An exception is an unexpected error that occurs at runtime,exceptions disrupt the normal flow of program.
Exceptions : ArrayIndexOutOfBoundException,IOException,FileNotFoundException,
To handle exceptions we will use try catch clause. When an exception is caught that exception has a stack trace which will provide information about the error and path that the code took before getting there.
Example:

```
package chapter13;

import java.io.File;
public class ExceptionHandling {
   public static void main(String[] args) {
      CreateNewFile();
   }
   public static void CreateNewFile(){
      File file = new File("resources/Unexist.txt");
      try{
         file.createNewFile();
      } catch (Exception e) {
         System.out.println("file does not exist");
         e.printStackTrace();
      }
   }
}
```

```
file does not exist
java.io.IOException: No such file or directory
	at java.base/java.io.UnixFileSystem.createFileExclusively(Native Method)
	at java.base/java.io.File.createNewFile(File.java:1024)
	at chapter13.ExceptionHandling.CreateNewFile(ExceptionHandling.java:13)
	at chapter13.ExceptionHandling.main(ExceptionHandling.java:7)

Process finished with exit code 0
```

There can be multiple catch clauses to handle different kinds of exceptions. If multiple catch clauses contain related exceptions the subclasses catch clause must appear first, otherwise it would never have the possibility of reaching other exception clauses.

The Finally clause is executed after the try and catch clauses, even if the catch clauses don't execute.finally clause will execute no matter what if exceptions are thrown or not thrown.

Example for rethrowing exception

```
public class ExceptionHandling {
  public static void main(String[] args) throws IOException {
     rethrowHandling();
public static void rethrowHandling() throws IOException {
  File file = new File("resources/Unexist.txt");
     file.createNewFile();
  }
```