Java

Introduction

“We want to get engineers to think about something else.”
— James Gosling, Creator of Java Programming language

Java is a general-purpose class-based, object-oriented programming language. The syntax of java is similar to C and C++ . Java was originally developed by James Gosling at Sun Microsystem. It was released in 1995 as a core component of Sun Microsystems 'Java platform.

It was designed to fulfil the five main purpose.

1. It must be simple, object-oriented, and familiar.
2. It must be robust and secure.
3. It must be architecture-neutral and portable.
4. It must execute with high performance.
5. It must be interpreted, threaded, and dynamic.

Benefits of using Java:

- Java is easy to learn.

  Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

- Java is object-oriented.

  This allows you to create modular programs and reusable code.

- Java is platform-independent.

  One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

Here is the link of IDE which I personally suggest to use

https://www.jetbrains.com/help/idea/installation-guide.html

Popularity of Java Worldwide:

Highest Position (since 2001): #1 in Jan 2020

Lowest Position (since 2001): #2 in Mar 2015

Language of the Year: 2005, 2015

There we have TIOBE Index for January 2020

Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	^	C#	5.349%	+2.07%
6	5	v	Visual Basic .NET	5.287%	-1.17%
7	6	v	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%
9	15	^^	Swift	1.795%	+0.61%
10	9	v	SQL	1.504%	-0.77%
11	18	^^	Ruby	1.063%	-0.03%
12	17	^^	Delphi/Object Pascal	0.997%	-0.10%
13	10	v	Objective-C	0.929%	-0.85%
14	16	^	Go	0.900%	-0.22%
15	14	v	Assembly language	0.877%	-0.32%
16	**20**	^^	**Visual Basic**	**0.831%**	**-0.20%**
17	25	^^	D	0.825%	+0.25%
18	12	vv	R	0.808%	-0.52%
19	13	vv	Perl	0.746%	-0.48%
20	11	vv	MATLAB	0.737%	-0.76%

Basic Structure

```
public class Hello {
    public static void main(String[] args){
        System.out.println("Hello World");
//This is given by Learn Programming
```

```
        int myFirstNumber = (5+9) +(10*7);
        int mySecondNumber = 16;
        int myThirdNumber = myFirstNumber*2;
        int myTotal = myFirstNumber + mySecondNumber + myThirdNumber;
        int myLastNumber = 1000- myTotal;
        System.out.println( myLastNumber );

    }
}
```

Many of the things we have already explained in the Basic portion. So for better understanding first read it.

1.What is **public class** mean?

public : It is an access modifier which means that it defines the scope or how other part of code or even other's code can access your code.

class : A class in Java is a template that is used to create and define objects, object data types, and methods. A class declaration constitutes of the following parts:

- ★ Modifier :

Eg. here it is *public*.

- ★ Class name

Eg. Here it is *Hello.*

- ★ The left and right curly braces **{}** is used to define the class body so anything we write in it is the part of this class.

2.What is **public static void main (String[] args)** mean?

public : It is an access modifier same as used in class.

static : A static member is a member of a class that isn't associated with an instance of a class. Instead, the member belongs to the class itself. As a result, you can access the static member without first creating a class instance.

Now for this don't take too much on your brain as we talk this latter in this course.

void : It is nothing more than a java keyword which shows that our method is not returning anything.

There are more java keywords we can use here as *int* which return integer number or other **data types**(already discuss in basic knowledge)

main : main is a ***method*** we usually call it a special method as Java looks for when we running a program. It is the entry point of any Java code.

3.What is ***method***?

It is a collection of statement(one or more) that perform an operation.

4.What we have to write inside **()** i.e. what is **String[] args**?

Now **String[] args** in Java is an array of strings which stores arguments passed by command line while starting a program. All the command line arguments are stored in that array.

() is used to store the type of variable we want to store in method.

5.What is **statements** in Java?

The complete command which we want to execute is called **statements**.

For eg. **System.out.println** is a statement which is calling a Java method to print our statement. Here we are using a Java built in functionality to print out our statement.

6.What are **Comments**?

Comments are ignored by the computer. It is basically for humans only . It is used to desccribe something in your code.

For writing comments we use **//** in front of code. Anything after this is ignored by the computer. It is also used to disable the code temporarily.

I personally suggest to write the whole code given in your IDE and then see the outcomes.

Now I am giving a simple code try to run it in your IDE and see what the outcomes:

```
package com.company;

public class Main {

    public static void main(String[] args) {
        int numberA = 25;
        int numberB = 20;
        int sumOfNumbers = numberA + numberB;

        String LearnProg= "Now you see the difference";
```

```
    String numberC="25";
     String numberD="20";
     String SumOfCandD=numberC+numberD;

    System.out.println(sumOfNumbers);
     System.out.println(SumOfCandD);
     System.out.println(LearnProg+" between " +SumOfCandD+" "+sumOfNumbers);

  }
}
```

Now I hope you make out this.

So what happen here is when you print sumOfNumbers, Basically you are adding two integers so the result is 45. But when you try out to print SumOfCandD, You are adding two strings that is why the result 2520. Now a question arises what do mean by adding a string.

Adding a string mean to add the text representation of number(String data type converts number into text). Which in case appends the new one in the old one. It is not recommended to appended value like this as it is inefficient. Latter on in this we do a thing StringBuffer to do this stuff.

In Java *Strings are immutable* which means you cannot change a string after it created instead of it a new string is created.

The basic things are already explained in the basic portion. Now I provided you the link where you get the list of Java Keywords which keep on increasing day by day.

https://en.wikipedia.org/wiki/List_of_Java_keywords

Now there are some questions given to give you hand on experience in java

I personally suggest to first solve the question on yourself and try less and less to see the solution.

and I if you have any doubt ask on our mail ID.

When people cheat in any arena, they diminish themselves—they threaten their own self-esteem and their relationships with others by undermining the trust they have in their ability to succeed and in their ability to be true.

—Cheryl Hughes

1.Write a Java program to print 'Hello' on screen and then print your name on a separate line.

Expected Output :

Hello
Anmol Varshney

```
package com.company;

public class Main {

    public static void main(String[] args) {
        System.out.println("Hello\nAnmol Varshney");

    }
}
```

2. Write a Java program to print the result of the following operations. Go to the editor
Test Data:
a. -5 + 8 * 6
b. (55+9) % 9
c. 20 + -3*5 / 8
d. 5 + 15 / 3 * 2 - 8 % 3
Expected Output :
43
1
19
13

```
package com.company;

public class Main {

    public static void main(String[] args) {
        System.out.println(-5 + 8 * 6);
        System.out.println((55+9) % 9);
        System.out.println(20 + -3*5 / 8);
        System.out.println(5 + 15 / 3 * 2 - 8 % 3);
    }
}
```

3. Anshul and Khushi just started to learn the java. So they decided to made a Speed converter which converts the speed from kilometersPerHour to MilesPerHour. They found notes on it so help them to made the speed converter.

Write a method that has 1 parameter of type double. This method needs to return the rounded value of the calculation of type long. If the parameter is less than 0, the method needs to return -1 to indicate an invalid value. Otherwise, if it is positive, calculate the value of miles per hour, round it and return it.

Examples of input/output:

* (1.5); → should return value 1

* (10.25); → should return value 6

*(-5.6); → should return value –1

Write another method called printConversion with 1 parameter of type double with the name kilometersPerHour.This method should not return anything (void) and it needs to calculate milesPerHour from the kilometersPerHour parameter.

Then it needs to print a message in the format "XX km/h = YY mi/h".

XX represents the original value kilometersPerHour.
YY represents the rounded milesPerHour from the kilometersPerHour parameter.

If the parameter kilometersPerHour is < 0 then print the text "Invalid Value".
Use method Math.round to round the number of calculated miles per hour(double). The method round returns long.
By the way Math.round() is a built-in math method which returns the closest long to the argument.
NOTE: All methods should be defined as public static like we have been doing .
NOTE: 1 mile per hour is 1.609 kilometers per hour

```
public class SpeedConverter {

   public static long toMilesPerHour(double kilometersPerHour) {
      if(kilometersPerHour<0){
         return -1;
      }
      return Math.round(kilometersPerHour/1.609);
   }

   public static void  printConversion(double kilometersPerHour){
      if(kilometersPerHour>0){
         long milesPerHour = toMilesPerHour(kilometersPerHour);
         System.out.println(kilometersPerHour + "km/hr = " + milesPerHour + "mi/hr");
      }
      else{
         System.out.println("Invalid");
      }
```

```
    }
}
```

4. Amar and Shikhar are the handosme guy in there group. Once they argue about the actual number of megabytes and kilobytes in a given number of kilobytes. So to find out who is right they went to Shubham. He provide them a java code which sort out there problem guess out the java code with the help of information found in Shubham notebook.

Write a method that has 1 parameter of type int.
The method should not return anything (void) and it needs to calculate the megabytes and remaining kilobytes from the kilobytes parameter.
Then it needs to print a message in the format "XX KB = YY MB and ZZ KB".
XX represents the original value kiloBytes.
YY represents the calculated megabytes.
ZZ represents the calculated remaining kilobytes.
For example, when the parameter kiloBytes is 2500 it needs to print "2500 KB = 2 MB and 452 KB"
If the parameter kiloBytes is less than 0 then print the text "Invalid Value".
TIP: Use the remainder operator
1 MB = 1024 KB

```
public class MegaBytesConvereter {
   public static void printMegaBytesAndKiloBytes(int kiloBytes){
      if(kiloBytes<0)
         System.out.println("Invalid Value");
      else {

            int KiloBytes = kiloBytes % 1024;
            int MegaBytes = kiloBytes / 1024;
            System.out.println(kiloBytes + " KB " + "=" +MegaBytes + "MB" + " And " + KiloBytes
+ "KB");
      }
   }
}
```

5.Anmol has a dog named Jacky. He is very cute and lovely. But Jacky has a very weird habit of barking at night which make the whole colony(named Factory Colony)disturbed. So help Anmol to write a program in which every time Jacky bark at night Anmol wakeup to keep him quit.
Write a method shouldWakeUp that has 2 parameters.
1st parameter should be of type boolean and be named barking it represents if Jacky is currently barking.

2nd parameter represents the hour of the day and is of type int with the name hourOfDay and has a valid range of 0-23.
Anmol have to wake up if the dog is barking before 8 or after 22 hours so in that case return true.
In all other cases return false.
If the hourOfDay parameter is less than 0 or greater than 23 return false.

Examples of input/output:
* shouldWakeUp (true, 1); → should return true
* shouldWakeUp (false, 2); → should return false since the Jacky is not barking.
* shouldWakeUp (true, 8); → should return false, since it's not before 8.
* shouldWakeUp (true, -1); → should return false since the hourOfDay parameter needs to be in a range 0-23.
TIP: Use the if else statement with multiple conditions.

```
public class BarkingDog {
  public static boolean shouldWakeUP(boolean barking, int hourOfDay) {

    if ((barking) && (hourOfDay < 8 || hourOfDay > 22)){
      return true;}
    if (hourOfDay < 0 || hourOfDay < 23){
      return false;
  }
    return false;
  }
}
```

6. After doing so many problems Anshul ask Khushi that, "Sister why not be make a program which tells us that a given year is leap year or not".
Then they both started it and completed very fastly. Their record time to make program is 7 minutes. They challenge you to break their record.
Write a method isLeapYear with a parameter of type int named year.
The parameter needs to be greater than or equal to 1 and less than or equal to 9999.
If the parameter is not in that range return false.
Otherwise, if it is in the valid range, calculate if the year is a leap year and return true if it is a leap year, otherwise return false.

To determine whether a year is a leap year, follow these steps:
1. If the year is evenly divisible by 4, Otherwise go to step 5
2. If the year is evenly divisible by 100, go to step3. Otherwise go to step 4.
3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days). The method isLeapYear needs to return true.
5. The year is not a leap year (it has 365 days). The method isLeapYear needs to return false.

The following years are not leap years:
1700, 1800, 1900, 2100, 2200, 2300, 2500, 2600
This is because they are evenly divisible by 100 but not by 400.
The following years are leap years:
1600, 2000, 2400
This is because they are evenly divisible by both 100 and 400.
Examples of input/output:
* isLeapYear(-1600); → should return false since the parameter is not in range (1-9999)
* isLeapYear(1600); → should return true since 1600 is a leap year
* isLeapYear(2017); → should return false since 2017 is not a leap year
* isLeapYear(2000); → should return true because 2000 is a leap year

```
public class LeapYear {
   public static boolean isLeapYear(int year) {
      if (year >= 1 && year <= 9999) {
         if (((year % 4 == 0) && (year % 100 != 0) ) || (year % 400 == 0))
            return  true;

      }
      return false;
   }

}
```

7. After writing C++ content for you Shubham fell drowsy so he decided to play a game with Ansh(Python Champion). They write a number on paper. If their number is same upto three decimal places they give party to Anmol and if not then Anmol took party from them(>< party is fix!!). So help Anmol to write a program which decided their number is same or not. Write a method areEqualByThreeDecimalPlaces with two parameters of type double.The method should return boolean and it needs to return true if two double numbers are the same up to three decimal places. Otherwise, return false.

EXAMPLES OF INPUT/OUTPUT:
* areEqualByThreeDecimalPlaces(-3.1756, -3.175); → should return true since numbers are equal up to 3 decimal places.
* areEqualByThreeDecimalPlaces(3.175, 3.176); → should return false since numbers are not equal up to 3 decimal places
* areEqualByThreeDecimalPlaces(3.0, 3.0); → should return true since numbers are equal up to 3 decimal places.
* areEqualByThreeDecimalPlaces(-3.123, 3.123); → should return false since numbers are not equal up to 3 decimal places.
TIP: Use paper and pencil.
TIP: Use casting.

```
public class DecimalComparator {
  public static boolean areEqualByThreeDecimalPlaces (double a, double b){
    int aa = (int) (1000*a);
    int bb = (int)(1000*b);
    if(aa == bb)
      return true;
    return false;
  }
}
```

8. Mr. Sanskar wants to become rich. So he started to buy the lottery tickets. There are two numbers written on the ticket. If the sum of these two numbers are equal to the number written on the lottery board then he won the lottery. So help him to find out that he win the lottery or lost it.

Write a method hasEqualSum with 3 parameters of type int.
The method should return boolean and it needs to return true if the sum of the first and second parameter is equal to the third parameter. Otherwise, return false.
EXAMPLES OF INPUT/OUTPUT:
* hasEqualSum(1, 1, 1); should return false since 1 + 1 is not equal to 1
* hasEqualSum(1, 1, 2); should return true since 1 + 1 is equal to 2
* hasEqualSum(1, -1, 0); should return true since 1 + (-1) is 1 - 1 and is equal to 0

```
public class EqualSumChecker {
  public static boolean hasEqualSum(int x, int y, int z) {
    if(x + y == z) {
      return true;
    } return false;
  }
}
```

9. Being a teenager is an amazing time and a hard time. It's when you make your best friends - I have girls who will never leave my heart and I still talk to. You get the best and the worst as a teen. You have the best friendships and the worst heartbreaks. So actually what we have to do is write a method named hasTeen with 3 parameters of type int. The method should return boolean and it needs to return true if one of the parameters is in range 13(inclusive) - 19 (inclusive). Otherwise return false.
We'll say that a number is "teen" if it is in the range 13 -19 (inclusive).

EXAMPLES OF INPUT/OUTPUT:
* hasTeen(9, 99, 19); should return true since 19 is in range 13 - 19
* hasTeen(23, 15, 42); should return true since 15 is in range 13 - 19
* hasTeen(22, 23, 34); should return false since numbers 22, 23, 34 are not in range 13-19

Drama, lies, tears...teenage years.

Write another method named isTeen with 1 parameter of type int.
The method should return boolean and it needs to return true if the parameter is in range 13(inclusive) - 19 (inclusive). Otherwise return false.
EXAMPLES OF INPUT/OUTPUT:
* isTeen(9); should return false since 9 is in not range 13 - 19
* isTeen(13); should return true since 13 is in range 13 – 19

```
public class TeenNumberChecker {
   public static boolean hasTeen(int a, int b, int c){
      if((a <20 && a>12)  || (b<20 && b>12) || (c<20&& c>12)){
         return true;
      }
      return false;
   }
   public static boolean isTeen(int x){
      if(x<20 && x>12){
         return true;
      }
    return false;
   }
}
```

10. Anshul feel boring by doing such simple questions so he asked Khushi to make it a little bit more interesting. She asked Anmol. Anmol gave them this problem and say Lets enjoy!!. Create a method called getDurationString with two parameters, first parameter minutes and 2nd parameter seconds.

You should validate that the first parameter minutes is >= 0.
You should validate that the 2nd parameter seconds is >= 0 and <= 59.

The method should return “Invalid value” in the method if either of the above are not true.
If the parameters are valid then calculate how many hours minutes and seconds equal the minutes and seconds passed to this method and return that value as string in format XXh YYm ZZs where XX represents a number of hours, YY the minutes and ZZ the seconds.

Create a 2nd method of the same name but with only one parameter seconds.
Validate that it is >= 0, and return “Invalid value” if it is not true.
If it is valid, then calculate how many minutes are in the seconds value and then call the other overloaded method passing the correct minutes and seconds calculated so that it can calculate correctly.

Call both methods to print values to the console.
Tips:
  Use int or long for your number data types is probably a good idea.

1 minute = 60 seconds and 1 hour = 60 minutes or 3600 seconds.
Methods should be static as we have used previously.
For the input 61 minutes output should be 01h 01m 00s, but it is ok if it is 1h 1m 0s (Tip: use if-else)

```
public class Main {

    public static void main(String[] args) {
        getDurationString(90, 40);
        getDurationString(900);
    }

    public static int getDurationString(int minutes, int seconds) {
        if ((minutes < 0) && (seconds < 0 || seconds > 60)) {
            System.out.println("Invalid value ");
            return -1;
        } else {
            int hours = (int) minutes / 60;
            hours += (int) seconds / 3600;
            int min = minutes % 60;
            min += (int) seconds / 60;
            int sec = seconds % 60;
            String hoursString = hours + "h";
            if (hours < 10) {
                hoursString = "0" + hoursString;
            }
            String secondString = sec + "s";
            if (sec < 10) {
                secondString = "0" + secondString;
            }
            String minuteString = min + "m";
            if (min < 10) {
                minuteString = "0" + minuteString;
            }
            System.out.println(minutes + "minutes and " + seconds + " Seconds = " + hoursString
+ "" + minuteString + "" + secondString + "");
            return 0;
        }
    }
    public static int getDurationString(int seconds) {
        if (seconds < 0) {
            System.out.println("Invalid Value");
            return -1;
        } else {
```

```
        int min = (int) seconds / 60;
        int sec = seconds % 60;
        System.out.println(seconds + "second = " + min + "minutes " + sec + "seconds");
        return 0;
    }
  }
}
```

11. Monika is a good teacher. She can inspire hope, ignite the imagination, and instil a love of learning. So to make the maths easy she ask you to make a program in which you have to write a method named area with one double parameter named radius.
The method needs to return a double value that represents the area of a circle.
If the parameter radius is negative then return -1.0 to represent an invalid value.
Write another overloaded method with 2 parameters x and y (both doubles), where x and y represent the sides of a rectangle.
The method needs to return an area of a rectangle.
If either or both parameters is/are a negative return -1.0 to indicate an invalid value.
For formulas and PI value please check the tips below.
Examples of input/output:
* area(5.0); should return 78.53975
* area(-1); should return -1 since the parameter is negative
* area(5.0, 4.0); should return 20.0 (5 * 4 = 20)
* area(-1.0, 4.0); should return -1 since first the parameter is negative
TIP: The formula for calculating the area of a rectangle is x * y.
TIP: The formula for calculating a circle area is radius * radius * PI.
TIP: For PI use a constant from Math class e.g. Math.PI

```
package com.company;

public class Main {

  public static void main(String[] args) {
    area(4.5);
    area(5, 4);
  }
  public static double area(double radius) {
    if (radius < 0) {
      System.out.println("Invalid");
      return -1;
    } else {
      double Area = Math.PI * (radius * radius);
      System.out.println(Area);
      return Area;
    }
```

```
    }
    public static double area(double x, double y) {
        if (x < 0 || y < 0) {
            return -1;
        } else {
            double areaOfRectangle = x * y;
            System.out.println(areaOfRectangle);
            return areaOfRectangle;
        }
    }
}
```

12. After doing question no. 10 Anshul and Khushi told Anmol that they need little bit more interesting. So Anmol said,"Baccha abhi toh party suru huai he".Enjoy this one
Write a method printYearsAndDays with parameter of type long named minutes.
The method should not return anything (void) and it needs to calculate the years and days from the minutes parameter.
If the parameter is less than 0, print text "Invalid Value".
Otherwise, if the parameter is valid then it needs to print a message in the format "XX min = YY y and ZZ d".
XX represents the original value minutes.
YY represents the calculated years.
ZZ represents the calculated days.
EXAMPLES OF INPUT/OUTPUT:
* printYearsAndDays(525600); → should print "525600 min = 1 y and 0 d"
* printYearsAndDays(1051200); → should print "1051200 min = 2 y and 0 d"
* printYearsAndDays(561600); → should print "561600 min = 1 y and 25 d"
TIPS:
* Use the remainder operator
* 1 hour = 60 minutes
* 1 day = 24 hours
* 1 year = 365 days

```
package com.company;

public class Main {

    public static void main(String[] args) {
        printYearsAndDays(561600);
    }

    public static void printYearsAndDays(long minutes) {
        if (minutes < 0) {
```

```
            System.out.println("Invalid Value");
        } else {
            long years = (long) minutes / (60 * 24 * 365);
            long days = (long) ((minutes / (60 * 24)) - years * 365);
            System.out.println(minutes + " min = " + years + " y and " + days + " d");
        }
    }
}
```

13. There is a cat in the Aryabhatta hostel. She is very cute. She spends most of the time playing with the students. In particular, they play if the temperature is between 25 and 35 (inclusive). Unless it is summer, then the upper limit is 45 (inclusive) instead of 35.
Write a method isCatPlaying that has 2 parameters. Method needs to return true if the cat is playing, otherwise return false
1st parameter should be of type boolean and be named summer it represents if it is summer.
2nd parameter represents the temperature and is of type int with the name temperature.
EXAMPLES OF INPUT/OUTPUT:
* isCatPlaying(true, 10); should return false since temperature is not in range 25 - 45
* isCatPlaying(false, 36); should return false since temperature is not in range 25 - 35 (summer parameter is false)
* isCatPlaying(false, 35); should return true since temperature is in range 25 – 35

```
package com.company;

public class Main {

    public static void main(String[] args) {
        isCatPlaying(true, 45);
    }

    public static boolean isCatPlaying(boolean summer, int temperature) {
        if ((((!summer) && (temperature < 25 || temperature > 35))) || ((summer) &&
(temperature < 25 || temperature > 45))) {
            System.out.println("false");
            return false;
        } else {

            System.out.println("true");
            return true;
        }
    }
}
```

14. Simple one just try it:
Write a method with the name sumDigits that has one int parameter called number.
If parameter is >= 10 then the method should process the number and return sum of all digits, otherwise return -1 to
indicate an invalid value.
The numbers from 0-9 have 1 digit so we don't want to process them, also we don't want to process negative numbers,
so also return -1 for negative numbers.
For example calling the method sumDigits(125) should return 8 since 1 + 2 + 5 = 8.
Calling the method sumDigits(1) should return -1 as per requirements described above.
Add some code to the main method to test out the sumDigits method to determine that it is working correctly for valid
and invalid values passed as arguments.
Hint:
Use n % 10 to extract the least-significant digit.
Use n = n / 10 to discard the least-significant digit.
The method needs to be static like other methods so far in the course.

```
package com.company;

public class Main {

    public static void main(String[] args) {
        System.out.println("sum of digits in 125 is " + sumDigit(125));
        System.out.println("sum of digits in -125 is " + sumDigit(-125));
        System.out.println("sum of digits in 4 is " + sumDigit(4));
        System.out.println("sum of digits in 3125 is " + sumDigit(3125));
public  static int sumDigit(int number){
        if(number<10){
            return -1;
        }
int sum = 0;
    while (number > 0) {
        int digit=number%10;
        number /=10;
     sum+=digit;
    }
return sum;
    }
}
```

15. Intermediate one just try it:
Write a method named sumFirstAndLastDigit with one parameter of type int called number.
The method needs to find the first and the last digit of the parameter number passed to the method, using a loop and return the sum of the first and the last digit of that number.
If the number is negative then the method needs to return -1 to indicate an invalid value.

Example input/output
* sumFirstAndLastDigit(252); → should return 4, the first digit is 2 and the last is 2 which gives us 2+2 and the sum is 4.
* sumFirstAndLastDigit(257); → should return 9, the first digit is 2 and the last is 7 which gives us 2+7 and the sum is 9.
* sumFirstAndLastDigit(0); → should return 0, the first digit and the last digit is 0 since we only have 1 digit, which gives us 0+0 and the sum is 0.
* sumFirstAndLastDigit(5); → should return 10, the first digit and the last digit is 5 since we only have 1 digit, which gives us 5+5 and the sum is 10.
* sumFirstAndLastDigit(-10); → should return -1, since the parameter is negative and needs to be positive.

```
package com.company;

public class Main {

    public static void main(String[] args) {
        sumFirstAndLastDigit(212121450);
    }
    public static int sumFirstAndLastDigit(int number) {
        if (number < 0) {
            return -1;
        }
        int num = number;
        int lastDigit = num % 10;
        while (num > 9) {
            num = num / 10;
        }
        System.out.println(lastDigit + num);
        return lastDigit + num;
    }
}
```

16. One more step just try it:
Write a method named getEvenDigitSum with one parameter of type int called number.
The method should return the sum of the even digits within the number.
If the number is negative, the method should return -1 to indicate an invalid value.

EXAMPLE INPUT/OUTPUT:
* getEvenDigitSum(123456789); → should return 20 since 2 + 4 + 6 + 8 = 20
* getEvenDigitSum(252); → should return 4 since 2 + 2 = 4
* getEvenDigitSum(-22); → should return -1 since the number is negative

```
package com.company;
public class Main {
    public static void main(String[] args) {
        getEvenDigitSum(123456789);
    }
    public static int getEvenDigitSum(int number) {
        if (number < 0) {
            return -1;
        }
        int sum = 0;
        while (number != 0) {
            int num = number % 10;
            if (num % 2 == 0) {
                sum = sum + num;
            }
            number = number / 10;
        }
        System.out.println(sum);
        return sum;
    }
}
```

17. Anshul get the task from his school Silver Stone Senior Secondary School to study about the Palindrome. He has to find out the given set of number is palindrome or not. Help him to solve his problem by making a java program.
Write a method called isPalindrome with one int parameter called number.
The method needs to return a boolean.
It should return true if the number is a palindrome number otherwise it should return false.
Check the tips below for more info about palindromes.
Example Input/Output
isPalindrome(-1221); → should return true
isPalindrome(707); → should return true
isPalindrome(11212); → should return false because reverse is 21211 and that is not equal to 11212.
Tip: What is a Palindrome number? A palindrome number is a number which when reversed is equal to the original number. For example: 121, 12321, 1001 etc.
Tip: Be careful with negative numbers. They can also be palindrome numbers.

Tip: Be careful with reversing a number, you will need a parameter for comparing a reversed number with the
starting number (parameter).

```
package com.company;
public class Main {
   public static void main(String[] args) {
      isPalindrome(355555550);
   }
   public static boolean isPalindrome(int number) {
      int reverse = 0;
      int num = number;
      while (num != 0) {
         int lastDigit = num % 10;
         reverse = reverse * 10 + lastDigit;
         num = num / 10;
         if (number == reverse) {
            System.out.println("true");
            return true;
         }
}
      System.out.println("false");
      return false;
   }
}
```

18. The sharing of joy, whether physical, emotional, psychic, or intellectual, forms a bridge between the sharers which can be the basis for understanding much of what is not shared between them, and lessens the threat of their difference.
But here is the sharing of digits between number. We have to find the number who have any digit common.
Write a method named hasSharedDigit with two parameters of type int.
Each number should be within the range of 10 (inclusive) - 99 (inclusive). If one of the numbers is not within the range, the method should return false.
The method should return true if there is a digit that appears in both numbers, such as 2 in 12 and 23; otherwise, the method should return false.

EXAMPLE INPUT/OUTPUT:
* hasSharedDigit(12, 23); → should return true since the digit 2 appears in both numbers
* hasSharedDigit(9, 99); → should return false since 9 is not within the range of 10-99
* hasSharedDigit(15, 55); → should return true since the digit 5 appears in both numbers.

```java
package com.company;
public class Main {
  public static void main(String[] args) {
      hasSharedDigit(238, 543);
    }
    public static boolean hasSharedDigit(int A, int B) {
      if ((A < 10 || A > 99) || (B < 10 || B > 99)) {
        return false;
      }
      int b = B;
      while (A != 0) {
        while (B != 0) {
          if (A % 10 == B % 10) {
            System.out.println("true");
            return true;
          }
          B /= 10;
        }
        B = b;
        A /= 10;
      }
      return false;
    }
}
```

19. Actually the thing is like that Anmol is felling happy after writing this Java content but still wants to add some more questions in this as practice makes a man or women perfect. So he add a little problem to revise you what you have done so far. So he also not provided you the solution of this problem.
Write a method named getGreatestCommonDivisor with two parameters of type int named first and second.
If one of the parameters is < 10, the method should return -1 to indicate an invalid value.
The method should return the greatest common divisor of the two numbers (int).
The greatest common divisor is the largest positive integer that can fully divide each of the integers (i.e. without leaving a remainder).

For example 12 and 30:
12 can be divided by 1, 2, 3, 4, 6, 12
30 can be divided by 1, 2, 3, 5, 6, 10, 15, 30
The greatest common divisor is 6 since both 12 and 30 can be divided by 6, and there is no resulting remainder.
EXAMPLE INPUT/OUTPUT:

* getGreatestCommonDivisor(25, 15); should return 5 since both can be divided by 5 without a remainder
* getGreatestCommonDivisor(12, 30); should return 6 since both can be divided by 6 without a remainder
* getGreatestCommonDivisor(9, 18); should return -1 since the first parameter is < 10
* getGreatestCommonDivisor(81, 153); should return 9 since both can be divided by 9 without a remainder
HINT: Use a while or a for loop and check if both numbers can be divided without a remainder.
HINT: Find the minimum of the two numbers.

20. "Oho yarro question to khatam ni ho rahe",that is I fell when I am writing the question and I suppose you fell the same when you doing the questions. So one more please.(No solution dosto!!)
Write a method named printFactors with one parameter of type int named number.
If number is < 1, the method should print "Invalid Value".
The method should print all factors of the number. A factor of a number is an integer which divides that number wholly (i.e. without leaving a remainder).
For example, 3 is a factor of 6 because 3 fully divides 6 without leaving a remainder. In other words 6 / 3 = 2

EXAMPLE INPUT/OUTPUT:
* printFactors(6); → should print 1 2 3 6
* printFactors(32); → should print 1 2 4 8 16 32
* printFactors(10); → should print 1 2 5 10
* printFactors(-1); → should print "Invalid Value" since number is < 1
HINT: Use a while or for loop.
NOTE: When printing numbers, each number can be in its own line. They don't have to be separated by a space.
For example, the printout for printFactors(10); can be:
1
2
5
10

21. Everyone wants to be perfect in her or his life. Then why the maths stay behind they proposed certain criteria to become perfect for a number.
What is the perfect number?
A perfect number is a positive integer which is equal to the sum of its proper positive divisors.
Proper positive divisors are positive integers that fully divide the perfect number without leaving a remainder and exclude the perfect number itself.

For example, take the number 6:
Its proper divisors are 1, 2, and 3 (since 6 is the value of the perfect number, it is excluded), and the sum of its proper divisors is 1 + 2 + 3 = 6.
Therefore, 6 is a perfect number (as well as the first perfect number).

Write a method named isPerfectNumber with one parameter of type int named number.
If number is < 1, the method should return false.
The method must calculate if the number is perfect. If the number is perfect, the method should return true; otherwise, it should return false.

EXAMPLE INPUT/OUTPUT:
* isPerfectNumber(6); should return true since its proper divisors are 1, 2, 3 and the sum is 1 + 2 + 3 = 6
* isPerfectNumber(28); should return true since its proper divisors are 1, 2, 4, 7, 14 and the sum is 1 + 2 + 4 + 7 + 14 = 28
* isPerfectNumber(5); should return false since its only proper divisor is 1 and the sum is 1 not 5
* isPerfectNumber(-1); should return false since the number is < 1
HINT: Use a while or for loop.
HINT: Use the remainder operator.

22. "Night gathers, and now my watch begins. It shall not end until my death. I shall take no wife, hold no lands, father no children. I shall wear no crowns and win no glory. I shall live and die at my post. I am the sword in the darkness. I am the watcher on the walls. I am the shield that guards the realms of men. I pledge my life and honour to the Night's Watch, for this night and all the nights to come." This is all what the life of coder is. Try this question and fell the emotions.

Write a method called numberToWords with one int parameter named number.
The method should print out the passed number using words for the digits.
If the number is negative, print "Invalid Value".
To print the number as words, follow these steps:
1. Extract the last digit of the given number using the remainder operator.
2. Convert the value of the digit found in Step 1 into a word. There are 10 possible values for that digit, those being 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Print the corresponding word for each digit, e.g. print "Zero" if the digit is 0, "One" if the digit is 1, and so on.
3. Remove the last digit from the number.
4. Repeat Steps 2 through 4 until the number is 0.
The logic above is correct, but in its current state, the words will be printed in reverse order. For example, if the number is 234, the logic above will produce the output "Four Three Two" instead of "Two Three Four". To overcome this problem, write a second method called reverse.

The method reverse should have one int parameter and return the reversed number (int). For example, if the number passed is 234, then the reversed number would be 432. The method reverse should also reverse negative numbers.

Use the method reverse within the method numberToWords in order to print the words in the correct order.

Another thing to keep in mind is any reversed number with leading zeroes (e.g. the reversed number for 100 is 001). The logic above for the method numberToWords will print "One", but that is incorrect. It should print "One Zero Zero". To solve this problem, write a third method called getDigitCount.

The method getDigitCount should have one int parameter called number and return the count of the digits in that number. If the number is negative, return -1 to indicate an invalid value.

For example, if the number has a value of 100, the method getDigitCount should return 3 since the number 100 has 3 digits (1, 0, 0).

Example Input/Output - getDigitCount method

* getDigitCount(0); should return 1 since there is only 1 digit
* getDigitCount(123); should return 3
* getDigitCount(-12); should return -1 since the parameter is negative
* getDigitCount(5200); should return 4 since there are 4 digits in the number

Example Input/Output - reverse method

* reverse(-121); should return -121
* reverse(1212); should return 2121
* reverse(1234); should return 4321
* reverse(100); should return 1

Example Input/Output - numberToWords method

* numberToWords(123); should print "One Two Three".
* numberToWords(1010); should print "One Zero One Zero".
* numberToWords(1000); should print "One Zero Zero Zero".
* numberToWords(-12); should print "Invalid Value" since the parameter is negative.

HINT: Use a for loop to print zeroes after reversing the number. As seen in a previous example, 100 reversed becomes 1, but the method numberToWords should print "One Zero Zero". To get the number of zeroes, check the difference between the digit count from the original number and the reversed number.

NOTE: When printing words, each word can be in its own line. For example, numberToWords(123); can be:

One
Two
Three

They don't have to be separated by a space.

NOTE: In total, you have to write 3 methods.

```
package com.company;

public class NumberToWords {

    public static void numberToWords(int number) {
        if (number < 0) {
            System.out.println("Invalid Value");
        }
        if (number == 0) {
            System.out.println("Zero");
        }

        int numDigit = getDigitCount(number); //calculates number of digits in number
        int revDigit = getDigitCount(reverse(number)); //calculates number of digits in reversed
number.
        // Send number to reverse, then result of reverse to getDigitCount
        int zerosToPrint = numDigit - revDigit; //calculate difference between original number
and reversed
        // number, identifies how many leading zero's to print (100 reversed to 001, or 1)
        number = reverse((number)); //reassigns number as the reversed number from reverse
method
        int printNumber = 0;

        while (number > 0) {
            printNumber = number % 10; //extracts last digit (technically the first digit as number
is now reversed number)
            switch (printNumber) { //prints the corresponding word for extracted digit
                case 0:
                    System.out.println("Zero");
                    break;
                case 1:
                    System.out.println("One");
                    break;
                case 2:
                    System.out.println("Two");
                    break;
                case 3:
                    System.out.println("Three");
                    break;
                case 4:
                    System.out.println("Four");
                    break;
                case 5:
                    System.out.println("Five");
```

```
                break;
            case 6:
                System.out.println("Six");
                break;
            case 7:
                System.out.println("Seven");
                break;
            case 8:
                System.out.println("Eight");
                break;
            case 9:
                System.out.println("Nine");
                break;
        }
        number /= 10;
    }
    for (int i = 0; i < zerosToPrint; i++) { //prints the number of leading zero's required
        System.out.println("Zero");
    }

}

public static int reverse(int number) {
    int reversed = 0; //initialises reversed as 0
    int workNumber = 0;  //initialises workNumber as 0. This variable will be hold the
individual digits to pass to reversed

    while (number != 0) {
        workNumber = number % 10; //extracts final digit
        number /= 10; //removes final digit for next round
        reversed += workNumber; //adds extracted digit to reversed
        if (number != 0) { //if more digits to extract and add
            reversed *= 10; //*10 to add zero to end of reversed (otherwise will add numbers
together,
            // 2+3 = 5 rather than 2+3 = 23)
        }
    }
    return reversed; //returns the new reversed number to numberToWords
}

public static int getDigitCount(int number) {
    if (number < 0) {
        return -1;
    }
```

```
        if (number == 0) {
            return 1;
        }

        int digitCount = 0; //initialise digitCount as 0
        for (int i = 0; number > 0; i++) {
            number /= 10; //remove last digit
            digitCount++; //+1 to digitCount for every digit removed
        }
        return digitCount;
    }
}
```

23. Anshul was stubborn from his childhood. He wept out for very little little things(ya you may be the anshul). Once he started stubborn. Then his unlcle(Sandeep) asked his father(Anees) what happen. He said the he wanted the largest prime number among all. Then his sister (Khushi) enters and made a program to sort out this problem. Guess out the program.

Write a method named getLargestPrime with one parameter of type int named number.
If the number is negative or does not have any prime numbers, the method should return -1 to indicate an invalid value.
The method should calculate the largest prime factor of a given number and return it.
EXAMPLE INPUT/OUTPUT:
* getLargestPrime (21); should return 7 since 7 is the largest prime (3 * 7 = 21)
* getLargestPrime (217); should return 31 since 31 is the largest prime (7 * 31 = 217)
* getLargestPrime (0); should return -1 since 0 does not have any prime numbers
* getLargestPrime (45); should return 5 since 5 is the largest prime (3 * 3 * 5 = 45)
* getLargestPrime (-1); should return -1 since the parameter is negative
HINT: Since the numbers 0 and 1 are not considered prime numbers, they cannot contain prime numbers.

24. “There are only patterns, patterns on top of patterns, patterns that affect other patterns. Patterns hidden by patterns. Patterns within patterns.
If you watch close, history does nothing but repeat itself.
What we call chaos is just patterns we haven't recognized. What we call random is just patterns we can't decipher. what we can't understand we call nonsense. What we can't read we call gibberish.
There is no free will.
There are no variables.”
So there is importance of pattern try to solve out this pattern.
Write a method named printSquareStar with one parameter of type int named number.

If number is < 5, the method should print "Invalid Value".

The method should print diagonals to generate a rectangular pattern composed of stars (*). This should be accomplished by using loops (see examples below).

EXAMPLE INPUT/OUTPUT:

EXAMPLE 1

printSquareStar(5); should print the following:

```
*****

** **

* * *

** **

*****
```

Explanation:

***** 5 stars

** ** 2 stars space 2 stars

* * * 1 star space 1 star space 1 star

** ** 2 stars space 2 stars

***** 5 stars

EXAMPLE 2

printSquareStar(8); should print the following:

```
********

**  **

* * * *

* ** *

* ** *
```

* * * *

** **

****

The patterns above consist of a number of rows and columns (where number is the number of rows to print). For each row or column, stars are printed based on four conditions (Read them carefully):

* In the first or last row

* In the first or last column

* When the row number equals the column number

* When the column number equals rowCount - currentRow + 1 (where currentRow is current row number)

HINT: Use a nested loop (a loop inside of a loop).

HINT: To print on the same line, use the print method instead of println, e.g. System.out.print(" "); prints a space and does not "move" to another line.

HINT: To "move" to another line, you can use an empty println call, e.g. System.out.println(); .

Now I believe that by doing so many problems you get a better understanding of basics in java and I am also believing that most of the question you done by yourself. If you have any doubt in any question just mail us. Your reviews are always welcomed. Most importantly If you want to practice more good questions go to C++ section and practice them in Java.

How to take input from user

```
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
```

```
System.out.println("Enter your year of birth:");


  int yearOfBirth = scanner.nextInt();
      scanner.nextLine();
      System.out.println("Enter your name: ");
      String name = scanner.nextLine();


  int age = 2020 - yearOfBirth;

      System.out.println("Your name is " + name + ", and you are " + age + " years old.");
      scanner.close();
   }
}
```

This the basic syntax to take input from user.
Scanner here is a class in java.util package used for obtaining the input of the primitive types like int, double, etc.

When you write `Scanner scanner = new Scanner(System.in);` also write `scanner.close();` because your program is run but it takes a lot of memory with out it as it is not close. `System.in` is an InputStream which is typically connected to keyboard input of console programs. Its function is basically to take input from user. `scanner.nextInt()` is used to take Int number. For taking ***double*** write **.Double** in place of **.Int**. Similarly done for other data types. For strings we use use `scanner.nextLine();`. It is also used for handle next line character (enter key). Now to understand this try to run the program without it.
This all about the How to take input from user.

25. After learning to take input Khushi is very excited to solve problems. She really love this app and rate it 5 star and recommended to many of her friends. To boost her confidence Anmol gave a very simple problem, You must also try it.
-Read 10 numbers from the console entered by the user and print the sum of those numbers.
-Create a Scanner like we did in the previous video.
-Use the hasNextInt() method from the scanner to check if the user has entered an int value.
-If hasNextInt() returns false, print the message Invalid Number.
-Continue reading until you have read 10 numbers.
-Use the nextInt() method to get the number and add it to the sum.
-Before the user enters each number, print the message Enter number #x: where x represents the count, i.e. 1, 2, 3, 4, etc.
-For example, the first message printed to the user would be Enter number #1:, the next Enter number #2: , and so on.

Hint:
-Use a while loop.
-Use a counter variable for counting valid numbers.
-Close the scanner after you don't need it anymore.

```
package com.company;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int counter = 0;
      int sum = 0;
      while (true) {
         int order = counter + 1;
         System.out.println("Enter number #" + order + ":");
         boolean isAnInt = scanner.hasNextInt();
         if (isAnInt) {
            int number = scanner.nextInt();
            counter++;
            sum += number;
            if (counter == 10) {
               break;
            }
         } else {
            System.out.println("Invalid value");
         }
         scanner.nextLine();
      }
      System.out.println("sum = " + sum);
      scanner.close();
   }
}
```

26. Megha is the most intelligent girl in her group. She was challenged by her father that if she solved this problem she got married within a month(which she wants ><) other wise not. Help her to solve this problem.
-Read the numbers from the console entered by the user and print the minimum and maximum number the user has entered.

-Before the user enters the number, print the message Enter number:

-If the user enters an invalid number, break out of the loop and print the minimum and maximum number.

Hint:

-Use an endless while loop.

```
package com.timbuchalka;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      int min = Integer.MAX_VALUE;
      int max = Integer.MIN_VALUE;
      while(true) {
         System.out.println("Enter number:");
         boolean isAnInt = scanner.hasNextInt();
         if(isAnInt) {
            int number = scanner.nextInt();
            if(number > max) {
               max = number;
            }
            if(number < min) {
               min = number;
            }
         } else {
            break;
         }
         scanner.nextLine(); // handle input
      }
      System.out.println("min= "+ min + ", max= " + max);
      scanner.close();
   }
}
```

27. It was the time when Mahabharat was going on and Arjun was felling very sad. Lord Krishna came and said, “it does not matter how you starts, it always matter that how you end”. Arjun ask him I want to know the sum of the number of peoples I killed and their average. Krishna said that it was very simple I made a program for you in which every time you write the number of people you killed and at the end you get the result. As a Arjun try it to make it yourself.

Write a method called inputThenPrintSumAndAverage that does not have any parameters.

The method should not return anything (void) and it needs to keep reading int numbers from the keyboard.
When the user enters something that is not an int then it needs to print a message in the format "SUM = XX AVG = YY".
XX represents the sum of all entered numbers of type int.
YY represents the calculated average of all numbers of type long.
EXAMPLES OF INPUT/OUTPUT:
EXAMPLE 1:
INPUT:
1
2
3
4
5
a
OUTPUT
SUM = 15 AVG = 3
EXAMPLE 2:
INPUT:
hello
OUTPUT:
SUM = 0 AVG = 0
TIP: Use Scanner to read an input from the user.
TIP: Use casting when calling the round method since it needs double as a parameter.
NOTE: Use the method Math.round to round the calculated average (double). The method round returns long.

At the end I want to tell you some thing about programming which I follow and suggest you also to follow it.

- ☺ **Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.**

- ☺ **Most good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program.**

- ☺ **Any fool can write code that a computer can understand. Good programmers write code that humans can understand.**

- ☺ **The only way to learn a new programming language is by writing programs in it.**

- ☺ **Programming is like a game of golf. The point is not getting the ball in the hole but how many strokes it takes.**

Best of luck to everyone.