ASSIGNMENT No. 1

|  |
| --- |
| **Need of Java Programming:-**   1. java is object oriented ,high level programming language. 2. java is portable. 3. Java machine independent programming language. 4. Java is secure,fast,reliable. 5. It's relatively easy to make programs in java. 6. Also due to another character like encapsulation,polymorphism,inheritance.   7. Java is a platform indepedent language which leads to the easy portation from platform to another without changing the sourse code while other languages does not support to that.some minor changes must have to be done while changing the platform. |

|  |
| --- |
| Java virtual machine (JVM):-  A Java virtual machine (JVM) is mainly use to execute java byte code that is devloped by compiler.A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.  Once a Java virtual machine has been implemented for a given platform, any Java program can run on that platform.A Java virtual machine can interpret the bytecode one instruction at a time. |

|  |
| --- |
| JIT(Just In Time) Compiler   1. A just-in-time (JIT) compiler is a program that turns Java bytecode into instructions that can be sent directly to the processor. 2. Using the Java just-in-time compiler at the particular system platform compiles the bytecode into the particular system code . Once the code has been (re-)compiled by the JIT compiler, it will usually run more quickly in the computer. 3. The just-in-time compiler comes with the virtual machine and is used optionally. It compiles the bytecode into platform-specific executable code that is immediately executed. |

Program 1:-

If statement 1:-

|  |
| --- |
| Syntax:  If(condition)  {  Statement s1;  Statement s2;  .  .  .  Statement sn;  } |

Program:

|  |
| --- |
| import java.lang.Math;  class Squareroot  {  public static void main (String args[])  {  double s=25;  double r;  r=Math.sqrt(s);  System.out.println("squareroot of s is "+r);  if (r==5)  System.out.println("the answer is absolutely correct");  }  } |

Output:

|  |
| --- |
| squareroot of s is 5.0  the answer is absolutely correct |

Explanation:

|  |
| --- |
| we had declared variable double to calculate squareroot (r),of a no.(s).By using the sqrt() fn which was imported from jav object to use in program & calculated the squareroot . Now, by using if control statement and giving condition we confirmed whether the answer is absolutely correct.  As r=5, we got the output  **the answer is absolutely correct**  after exicutting the if statement & condition. |

Program 2:-

If statement 2:

|  |
| --- |
| Syntax:  If(condition)  {  Statement s1;  Statement s2;  .  .  .  Statement sn;  } |

Program:

|  |
| --- |
| public class Controlstatementifdemo2  {  public static void main (String args[])  {  double l=5,b=4,tb=8,h=5;  double a1,a2;  a1=l\*b;  System.out.println("area of rectange is "+a1);  a2=(tb\*h)/2;  System.out.println("area of triangle is"+a2);  if(a1==a2)  System.out.println("both figures occupy same area");  }  } |

Output:

|  |
| --- |
| area of rectange is 20.0  area of triangle is20.0  both figures occupy same area |

Explanation:

|  |
| --- |
| In the above program we have used the simple decision control structure **if**. While using **if** we are giving a condition which is checked by the compiler and if it is true the statements following it will be executed. In the above program the condition i.e**. a1=20** and also **a2=20** so the condition is true that’s why the  **Output: both the figures occupy same area**  is printed otherwise no output would be printed. |

Program 3:-

If-else statement 1:

|  |
| --- |
| Syntax:  If(condition)  {  statement s1;s2;……sn;  }  else  {  s1;s2;……..;sn;  } |

Program:

|  |
| --- |
| public static void main(String args[])  {  int a=21 ,b=43,c,d;  c=a+b;  System.out.println("addition is "+c);  if(c>50)  {  d=c\*2;  System.out.println(value of d is greater than 100");  }  else  {  d=c/2;  System.out.println("the no is less than 25");  } |

Output:

|  |
| --- |
| run:  addition is 64  value of d is greater than 100 |

Explanation:

|  |
| --- |
| In the above particular program we have used the dicision control structure If-Else. Using if-else structure we checked the **condition (C>50)**  and wrote two output for each possibilities. That is if the condition is true the first statement will be carried out and if the condition is not true then the statement of else gets executed. In the above program the condition is true that’s why the statement of if has get executed. Therefore we got the  **output :value of d is greater than 100 (d=c\*2).** |

Program 4:-

If-else statement 2:

|  |
| --- |
| Syntax:  If(condition)  {  statement s1;s2;……sn;  }  else  {  s1;s2;……..;sn;  } |

Program:

|  |
| --- |
| public class Ifelsedemo2  {  public static void main(String args[])  {  int a1=42,a2=45,a3=90;  double tot;  tot=a1+a2+a3;  if(tot==180)  System.out.println("the triangle is valid");  else  System.out.println("the triangle is invalid");  }    } |

Output:

|  |
| --- |
| run:  the triangle is invalid |

Explanation:

|  |
| --- |
| In the above program we assigned the value of angles to the variables and calculated the total. Now, using **condition whether tot=180** we written some statements in if loop & then in else.  As the condition became false the control passes to else & statements of else get executed. Therefore the **output is** :**the triangle is invalid** . |

Program 5:-

Nested If else demo1:

|  |
| --- |
| Syntax:  if(condition)  {  statement s1;s2;……sn;  }  else if(condition)  {  s1;s2;……..;sn;  }  …..  ……  else  {  s1;s2;……..;sn;  } |

Program:

|  |
| --- |
| class Nestedifelsedemo1  {  public static void main(String args[])  {  int x=45,y=65,z=20;  if(x>y&&x>z)  System.out.println("the greatest no is "+x);  else if(y>x&&y>z)  System.out.println("the greatest no is "+y);  else  System.out.println("the greatest no is "+z);  }    } |

Output:

|  |
| --- |
| run:  the greatest no is 65 |

Explanation:

|  |
| --- |
| In the above program again we check series of conditions to determine which of the number is greatest using nesting of if-else statements which is the series of if then continuously if else for conditions and for last condition else is used. |

Program 6:-

Nested If else demo1:

|  |
| --- |
| Syntax:  if(condition)  {  statement s1;s2;……sn;  }  else if(condition)  {  s1;s2;……..;sn;  }  ……  else  {  s1;s2;……..;sn;  } |

Program:

|  |
| --- |
| public class Ifelsedemo1  {  public static void main(String args[])  {  int sal=20000;  if(sal>40000)  System.out.println("CEO");  else if(sal<40000&&sal>25000)  {  System.out.println("Manager");  }  else if(sal<25000&&sal>15000)  {  System.out.println("Accountant");  }  else  System.out.println("Clerk");  }  } |

Output:

|  |
| --- |
| run:  Accountant |

Explanation:

|  |
| --- |
| In the above program again we check series of conditions to determine which of the number is greatest using nesting of if-else statements which is the series of if then continuously if else for conditions and for last condition else is used. |

Program 7:-

While statement:

|  |
| --- |
| Syntax:  Initialization;  while(condition)  {  Statement s1;  s2;s3;s4;……………….sn;  increment/decrement;  } |

Program:

|  |
| --- |
| public class Whiledemo1  {  public static void main(String args[])  {  int n=1;  System.out.println("odd numbers less than 20 are:");  while(n<=20)  {  System.out.println(""+n);  n=n+2;  }  }  } |

Output:

|  |
| --- |
| run:  odd numbers less than 20 are:  1  3  5  7  9  11  13  15  17  19 |

Explanation:

|  |
| --- |
| In the above program we are using while loop to execute same block of statements number of times. The while loop first checks the before entering into the loop, the control enters the loop if and only if the condition becomes true, as the condition is true the statements get executed and at last the variable used is increamented as per given condition. The loop continues till the condition , when condition becomes false the control comes ot of loop and statements below loop executes and the output is printed. |

Program 8:-

While statement:

|  |
| --- |
| Syntax:  Initialization;  while(condition)  {  Statement s1;  s2;s3;s4;……………….sn;  increment/decrement;  } |

Program:

|  |
| --- |
| public class Whiledemo2  {  public static void main(String args[])  {  double fact=1,n=10;  while(n!=0)  {  fact=fact\*n;  n--;  }  System.out.println("factorial is "+fact);  }    } |

Output:

|  |
| --- |
| run:  factorial is 3628800.0 |

Explanation:

|  |
| --- |
| In the above program we are using while loop to execute same block of statements number of times. The while loop first checks the before entering into the loop, the control enters the loop if and only if the condition becomes true, as the condition is true the statements get executed and at last the variable used is increamented as per given condition. The loop continues till the condition , when condition becomes false the control comes ot of loop and statements below loop executes and the output is printed |

Program 9:-

For Statement:

|  |
| --- |
| Syntax:  For(initialization;test condition;increment)  {  Body of loop;  } |

Program:

|  |
| --- |
| public class ForProgram1  {  public static void main(String[] args )  {  int a;  System.out.println(“the numbers divisible by 5 between 1 to 20 are:”)  for(a=1;a<=20;a++)  {  If(a%5==0)  System.out.println(a);  }  }  } |

Output:

|  |
| --- |
| run:  the numbers divisible by 5 between 1 to 20 are:s  5  10  15  20 |

Explanation:

|  |
| --- |
| In the above program we are using for loop to execute same block of statements number of times. As we can in the syntax the loop will keep on executing the body of loop until the condition gets false.  In the above program we have used the if statement so while execution it checks the condition of if and gives output . Therefore the numbers divisible by 5 get printed. |

Program 10:-

For Statement:

|  |
| --- |
| Syntax:  For(initialization;test condition;increment)  {  Body of loop;  } |

Program:

|  |
| --- |
| public class Fordemo2  {  public static void main(String[] args )  {  int a;  for(a=10;a>=1;a--)  {  System.out.println(a);  }  }  } |

Output:

|  |
| --- |
| 10  9  8  7  6  5  4  3  2  1 |

Explanation:

|  |
| --- |
| In the above program we are using for loop to execute same block of statements number of times. As we can in the syntax the loop will keep on executing the body of loop until the condition gets false. |

Program 11:-

Do-while Statement:

|  |
| --- |
| Syntax:  do  {  Statements;  }  while(condition); |

Program:

|  |
| --- |
| public class Dowhiledemo1  {  public static void main(String args[])  {  int n=5;  do  {  System.out.println("Hi Hello");  }  while(2<1);  }  } |

Output:

|  |
| --- |
| run:  Hi Hello |

Explanation:

|  |
| --- |
| In do-while loop first the statements of do get executed and then it checks the condition in while so if the condition is wrong then also the statement get execute at least once as in the above program. |

Program 12:-

Do-while Statement:

|  |
| --- |
| Syntax:  do  {  Statements;  }  while(condition); |

Program:

|  |
| --- |
| public class Dowhiledemo2  {  public static void main(String args[])  {  int a=20,b=22;  do  {  System.out.println("a is equal to b");  }  while(a==b);  }  } |

Output:

|  |
| --- |
| run:  a is equal to b |

Explanation:

|  |
| --- |
| In do-while loop first the statements of do get executed and then it checks the condition in while so if the condition is wrong then also the statement get execute at least once as in the above written program. |

Program 13:-

Break statement:

|  |
| --- |
| Syntax:  If(condition)  {  Statements;  break; /\*control comes out of loop\*/  } |

Program:

|  |
| --- |
| class Breakdemo1  {  public static void main(String args[])  {  int num=22,i = 2;  while( i <= num - 1 )  {  if ( num % i == 0 )  {  System.out.println( "Not a prime number" );  break ;  }  i++ ;  }  if ( i == num )  printf ( "Prime number" ) ; }    }  } |

Output:

|  |
| --- |
| run:  Not a prime number |

Explanation:

|  |
| --- |
| In this program the moment num % i turns out to be zero, i.e. num is exactly divisible by I, the message “**Not a prime number**” is printed and the control breaks out of the while loop because the no turned out to be not a prime.  The keyword break, breaks the control only from the while in which it is placed. |

Program 14:-

Continue Statement:

|  |
| --- |
| Syntax:  for(initialization;condition;increament)  If(condition)  {  Statements;  break; /\*control goes at loop\*/  } |

Program:

|  |
| --- |
| public class Continuedemo  {  public static void main(String args[])  {  int i, j ;  for ( i = 1 ; i <= 2 ; i++ )  {  for ( j = 1 ; j <= 2 ; j++ )  {  if ( i == j )  continue ;  System.out.println( ""+i ) ;  System.out.println( ""+j ) ;    }  }  }  } |

Output:

|  |
| --- |
| run:  1  2  2  1 |

Explanation:

|  |
| --- |
| when the value of i equals that of j, the continue statement takes the control to the for loop (inner) bypassing rest of the statements pending execution in the inner for loop . |

END