JAVA PROGRAMMING LAB (R2022058)

**LAB MANUAL**

B.TECH

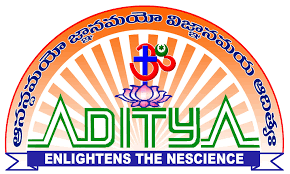
II YEAR – II SEM (R20)

(2021-22)

Prepared By

**Ms. SHAIK VAHIDA M.Tech(Ph.D)**

**Assistant Professor**



**DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY**

**Aditya College of Engineering & Technology**

Aditya Nagar, ADB Road, Surampalem – 533437

|  |
| --- |
|  |

SYLLABUS

**Exercise – 1 (BASICS)**

1. Write a JAVA program to display default value of all primitive data type of JAVA
2. Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.
3. Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.
4. Write a case study on public static void main (250 words)

**Exercise - 2 (Operations, Expressions, Control-flow, Strings)**

1. Write a JAVA program to search for an element in a given list of elements using binary search mechanism.
2. Write a JAVA program to sort for an element in a given list of elements using bubble sort (c). Write a JAVA program to sort for an element in a given list of elements using merge sort.
3. Write a JAVA program using StringBufferto delete, remove character.

**Exercise - 3 (Class, Objects)**

1. Write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method.
2. Write a JAVA program to implement constructor.

**Exercise - 4 (Methods)**

1. Write a JAVA program to implement constructor overloading
2. Write a JAVA program implement method overloading.

**Exercise - 5 (Inheritance)**

1. Write a JAVA program to implement Single Inheritance
2. Write a JAVA program to implement multi level Inheritance
3. Write a java program for abstract class to find areas of different shapes

**Exercise - 6 (Inheritance - Continued)**

1. Write a JAVA program give example for “super” keyword.
2. Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?

**Exercise - 7 (Exception)**

1. Write a JAVA program that describes exception handling mechanism
2. Write a JAVA program Illustrating Multiple catch clauses

**Exercise – 8 (Runtime Polymorphism)**

1. Write a JAVA program that implements Runtime polymorphism
2. Write a Case study on run time polymorphism, inheritance that implements in above problem

**Exercise – 9 (User defined Exception)**

1. Write a JAVA program for creation of Illustrating throw
2. Write a JAVA program for creation of Illustrating finally
3. Write a JAVA program for creation of Java Built-in Exceptions
4. Write a JAVA program for creation of User Defined Exception

**Exercise – 10 (Threads)**

1. Write a JAVA program that creates threads by extending Thread class .First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable)
2. Write a program illustrating isAlive and join ()
3. Write a Program illustrating Daemon Threads.

**Exercise - 11 (Threads continuity)**

1. Write a JAVA program Producer Consumer Problem
2. Write a case study on thread Synchronization after solving the above producer consumer problem

**Exercise – 12 (Packages)**

1. Write a JAVA program illustrate class path
2. Write a case study on including in class path in your os environment of your package.
3. Write a JAVA program that import and use the defined your package in the previous Problem

**Exercise - 13 (Applet)**

1. Write a JAVA program to paint like paint brush in applet.
2. Write a JAVA program to display analog clock using Applet.
3. Write a JAVA program to create different shapes and fill colors using Applet.

**Exercise - 14 (Event Handling)**

1. Write a JAVA program that display the x and y position of the cursor movement using Mouse.
2. Write a JAVA program that identifies key-up key-down event user entering text in a Applet.

**JAVA PROGRAMMING LAB( R2022058 )**

B.Tech 2/4, IV-SEMESTER

**COURSE OUTCOMES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Course**  **Code - CO** | **Course Outcomes** | **Blooms Taxonomy** |
| 1 | CO1 | Evaluate default value of all primitive data type, Operations, Expressions, Control-flow, Strings Determine Class, Objects, Methods, Inheritance, Exception, and Runtime Polymorphism. | Application |
| 2 | CO2 | Develop java applications using data structures searching and sorting techniques such as binary search, bubble sort, merge sort, | Application |
| 3 | CO3 | Construct applications that can handle exceptions. | Application |
| 4 | CO4 | Illustrating simple inheritance, multi-level inheritance | Application |
| 5 | CO5 | Construct Threads, Event Handling, implement packages, | Application |
| 6 | CO6 | Develop user interface components using applets that cover event handling. | Application |

**JAVA PROGRAMMING LAB( R2022058 )**

B.Tech 2/4, IV-SEMESTER

**CO & POs MAPPING**

| **Course**  **Code** | **Course Outcomes** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO 11** | **PO 12** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO1** | Evaluate default value of all primitive data type, Operations, Expressions, Control-flow, Strings Determine Class, Objects, Methods, Inheritance, Exception, and Runtime Polymorphism. | 2 | 3 | 2 | 3 | 3 |  |  |  |  |  |  | 2 |
| CO2 | Develop java applications using data structures searching and sorting techniques such as binary search, bubble sort, merge sort, | 2 | 2 | 3 | 3 | 3 |  |  |  |  |  |  | 2 |
| CO3 | Construct applications that can handle exceptions. | 2 | 3 | 3 | 3 | 3 |  |  |  |  |  |  | 2 |
| CO4 | Illustrating simple inheritance, multi-level inheritance, | 1 | 2 | 2 | 3 | 3 |  |  |  |  |  |  | 2 |
| **CO5** | Construct Threads, Event Handling, implement packages, | 1 | 2 | 2 | 3 | 3 |  |  |  |  |  |  | 2 |
| **CO6** | Develop user interface components using applets that cover event handling. | 1 | 2 | 2 | 2 | 3 |  |  |  |  |  |  | 2 |

**JAVA PROGRAMMING LAB( R2022058 )**

B.Tech 2/4, IV-SEMESTER

**CO & PSO MAPPING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course**  **Code** | **Course Outcomes** | **PSO1** | **PSO2** | **PSO3** |
| CO1 | Evaluate default value of all primitive data type, Operations, Expressions, Control-flow, Strings Determine Class, Objects, Methods, Inheritance, Exception, and Runtime Polymorphism. |  | 3 |  |
| CO2 | Develop java applications using data structures searching and sorting techniques such as binary search, bubble sort, merge sort, | 1 | 2 | 1 |
| CO3 | Construct applications that can handle exceptions. |  | 2 | 1 |
| CO4 | Illustrating simple inheritance, multi-level inheritance, |  | 3 |  |
| CO5 | Construct Threads, Event Handling, implement packages, |  | 2 |  |
| CO6 | Develop user interface components using applets that cover event handling. | **1** | **2** | **1** |

**GENERAL INSTRUCTIONS**

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.

3. Student should enter into the laboratory with:

* + Laboratory observation notes.
  + Laboratory Record updated up to the last session experiments.
  + Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.

5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.

6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.

7. Computer labs are established with sophisticated and high-end branded systems, which should be utilized properly.

8. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

**TOOLS USED DURING THE LAB**

|  |  |
| --- | --- |
| **IDE** | **C:\Users\Lenovo\Desktop\EC.png** |
| **JDK** | **C:\Users\Lenovo\Desktop\JAVA.png** |
| **DATA BASE** | MySQL |

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME OF THE EXPERIMENT** | **CO LEVEL** | **PAGE NO** |
| **1** | **Exercise – 1 (BASICS)**  a) Write a JAVA program to display default value of all primitive data type of JAVA  b) Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.  c) Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.  d) Write a case study on public static void main(250 words) | **CO1** | **13** |
| **2** | **Exercise - 2 (Operations, Expressions, Control-flow, Strings)**  a) Write a JAVA program to search for an element in a given list of elements using binary search mechanism.  b) Write a JAVA program to sort for an element in a given list of elements using bubble sort (c). Write a JAVA program to sort for an element in a given list of elements using merge sort.  c) Write a JAVA program using StringBufferto delete, remove character. | **CO1** | **18** |
| **3** | **Exercise - 3 (Class, Objects)**  a) Write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method.  b) Write a JAVA program to implement constructor. | **CO1** | **24** |
| **4** | **Exercise - 4 (Methods)**  a) Write a JAVA program to implement constructor overloading  b) Write a JAVA program implement method overloading. | **CO1** | **27** |
| **5** | **Exercise - 5 (Inheritance)**  a) Write a JAVA program to implement Single Inheritance  b) Write a JAVA program to implement multi-level Inheritance  c) Write a java program for abstract class to find areas of different shapes | **CO2** | **30** |
| **6** | **Exercise - 6 (Inheritance - Continued)**  a) Write a JAVA program give example for “super” keyword.  b) Write a JAVA program to implement Interface. What kind of Inheritance can be achieved? | **CO2** | **36** |
| **7** | **Exercise - 7 (Exception)**  a) Write a JAVA program that describes exception handling mechanism  b) Write a JAVA program Illustrating Multiple catch clauses | **CO2** | **39** |
| **8** | **Exercise – 8 (Runtime Polymorphism)**  a) Write a JAVA program that implements Runtime polymorphism  b) Write a Case study on run time polymorphism, inheritance that implements in above problem | **CO3** | **42** |
| **9** | **Exercise – 9 (User defined Exception)**  a) Write a JAVA program for creation of Illustrating throw  b) Write a JAVA program for creation of Illustrating finally  c) Write a JAVA program for creation of Java Built-in Exceptions  d) Write a JAVA program for creation of User Defined Exception | **CO3** | **44** |
| **10** | **Exercise – 10 (Threads)**   1. Write a JAVA program that creates threads by extending Thread class .First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable) 2. Write a program illustrating isAlive and join () 3. Write a Program illustrating Daemon Threads. | **CO4** | **50** |
| **11** | **Exercise - 11 (Threads continuity)**  a) Write a JAVA program Producer Consumer Problem  b) Write a case study on thread Synchronization after solving the above producer consumer problem | **CO4** | **57** |
| **12** | **Exercise – 12 (Packages)**  a) Write a JAVA program illustrate class path  b) Write a case study on including in class path in your os environment of your package.  c) Write a JAVA program that import and use the defined your package in the previous Problem | **CO5** | **60** |
| **13** | **Exercise - 13 (Applet)**  a) Write a JAVA program to paint like paint brush in applet.  b) Write a JAVA program to display analog clock using Applet.  c) Write a JAVA program to create different shapes and fill colors using Applet. | **CO6** | **63** |
| **14** | **Exercise - 14 (Event Handling)**  a) Write a JAVA program that display the x and y position of the cursor movement using Mouse.  b) Write a JAVA program that identifies key-up key-down event user entering text in a Applet. | **CO6** | **69** |
| **ADDITIONAL EXPERIMENTS** | | | |
| **1** | Write a JAVA program to build a Calculator in Swings | **CO6** | **73** |
| **2** | Write a JAVA program to display the digital watch in swing tutorial. | **CO6** | **80** |
| **3** | Write a JAVA program that to create a single ball bouncing inside a JPanel. | **CO6** | **82** |
| **4** | Write a JAVA program JTree as displaying a real tree upside down | **CO6** | **84** |

**Exercise - 1** (Basics)

**1 a). Write a JAVA program to display default value of all primitive data type of JAVA**

**AIM:** to display default value of all primitive data type of JAVA

**DESCRIPTION:** It's not always necessary to assign a value when a field is declared. Fields that are declared but not initialized will be set to a reasonable default by the compiler. Generally speaking, this default will be zero or null, depending on the data type. Relying on such default values, however, is generally considered bad programming style.

**PROCEDURE:**

1. Default values are values assigned by the compiler to the variables which are declared. They are different according to the return type of data type.
2. In order to use the default values first, declare the variable with data type and name (example: - int x, here int is the data type and x is the name of the variable).
3. If you don’t declare the variable before using it, it would result in a compile-time error.

**PROGRAM:**

// Displaying default values of all primitive types in a class

class Data {

int a;

float b;

double c;

boolean d;

char e;

short f;

byte g;

long h;

String i;

public static void main(String args[]) {

Data n = new Data();

System.out.println("the default values of primitive types:");

System.out.println("int : "+n.a);

System.out.println("float : "+n.b);

System.out.println("double : "+n.c);

System.out.println("boolean : "+n.d);

System.out.println("char : "+n.e);

System.out.println("short : "+n.f);

System.out.println("byte : "+n.g);

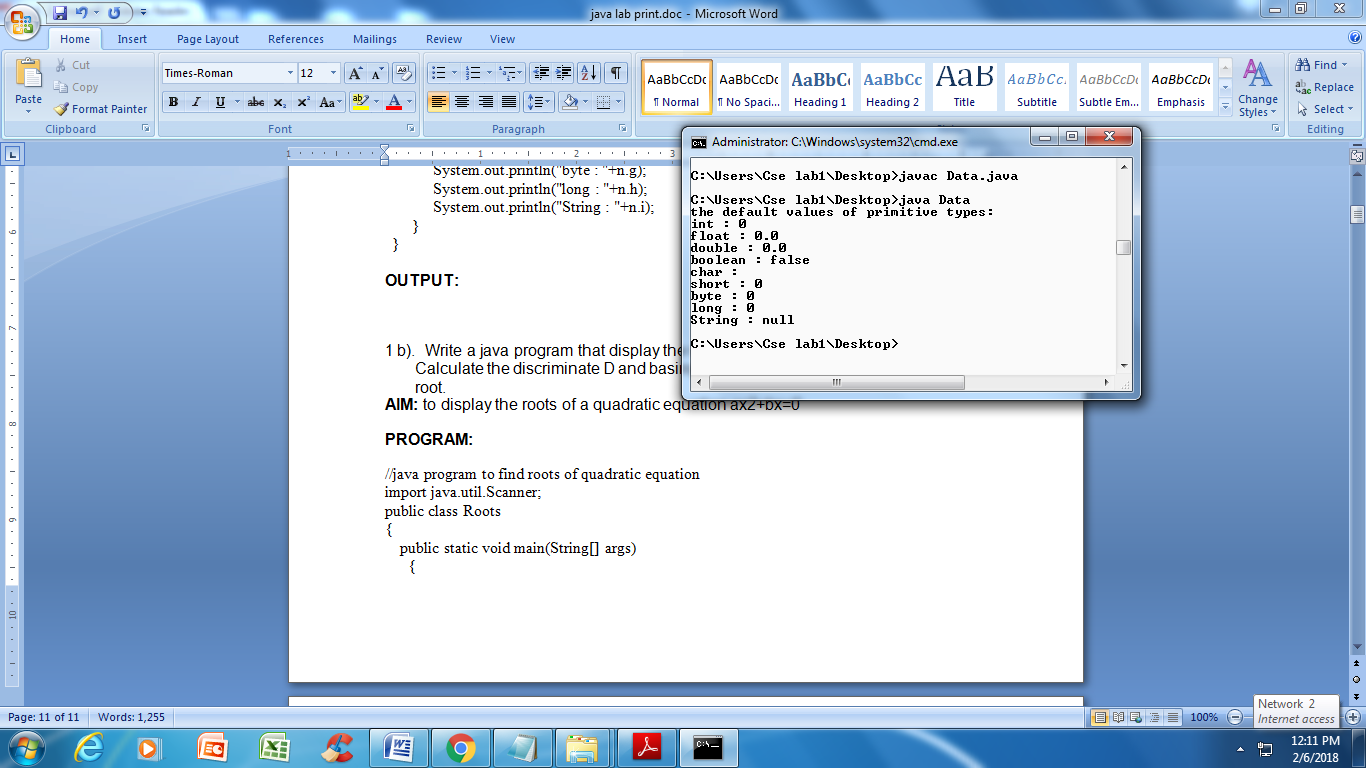
System.out.println("long : "+n.h);

System.out.println("String : "+n.i);

}

}

**OUTPUT:**



**1 b). Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.**

**AIM:** to display the roots of a quadratic equation ax2+bx=0

**ALGORITHM:**

1. Create a class and give implementation to read(), getDet(), and getRoots()

2. Using sqrt() calculate roots

3. create main class and create an instace to above class

4. Call methods using instance.

**PROGRAM:**

//java program to find roots of quadratic equation

import java.util.Scanner;

public class Roots

{

public static void main(String[] args)

{

int a, b, c;

double root1, root2, d;

Scanner s = new Scanner(System.in);

System.out.println("Given quadratic equation:ax^2 + bx + c");

System.out.print("Enter a:");

a = s.nextInt();

System.out.print("Enter b:");

b = s.nextInt();

System.out.print("Enter c:");

c = s.nextInt();

System.out.println("Given quadratic equation:"+a+"x^2 + "+b+"x + "+c);

d = b \* b - 4 \* a \* c;

if(d > 0)

{

System.out.println("Roots are real and unequal");

root1 = ( - b + Math.sqrt(d))/(2\*a);

root2 = (-b - Math.sqrt(d))/(2\*a);

System.out.println("First root is:"+root1);

System.out.println("Second root is:"+root2);

}

else if(d == 0)

{

System.out.println("Roots are real and equal");

root1 = (-b+Math.sqrt(d))/(2\*a);

System.out.println("Root:"+root1);

}

else

{

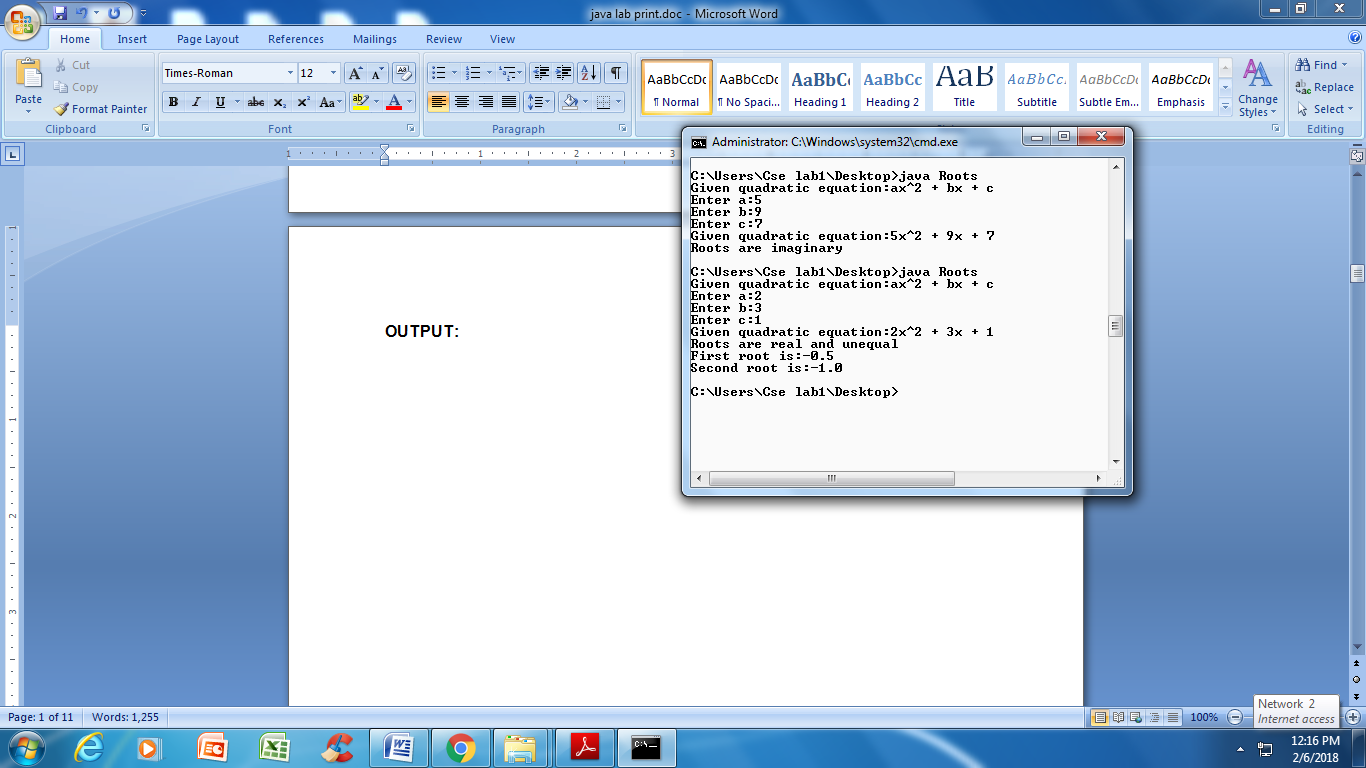
System.out.println("Roots are imaginary");

}

}

}

**OUTPUT:**



**1 c). Five Bikers Compete in a race such that they drive at a constant speed which may or may not be the same as the other. To qualify the race, the speed of a racer must be more than the average speed of all 5 racers. Take as input the speed of each racer and print back the speed of qualifying racers.**

**AIM:** to print the speed of the qualifying racer

**ALGORITHM:**

1. Take inputs as speed of five bikes.
2. Find the average speed of all five racers by finding the total speed of all five racers and dividing it by the number of racers(five in this case).
3. To print the speed of qualifying  To print the speed of qualifying racers, add a comparison testing to see if each racer's speed is greater than the avg speed.

**PROGRAM:**

import java.io.\*;

class Racer {

public static void main(String args[])throws IOException {

int i, sum = 0;

float avg;

int speed[]= new int[5];

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the values:");

for(i=0; i<5; i++) {

speed[i]=Integer.parseInt(br.readLine());

sum+= speed[i];

}

avg = sum/5;

for(i=0; i<5; i++) {

if(speed[i]>avg) {

System.out.println("the speed of qualified racer is " +speed[i]);

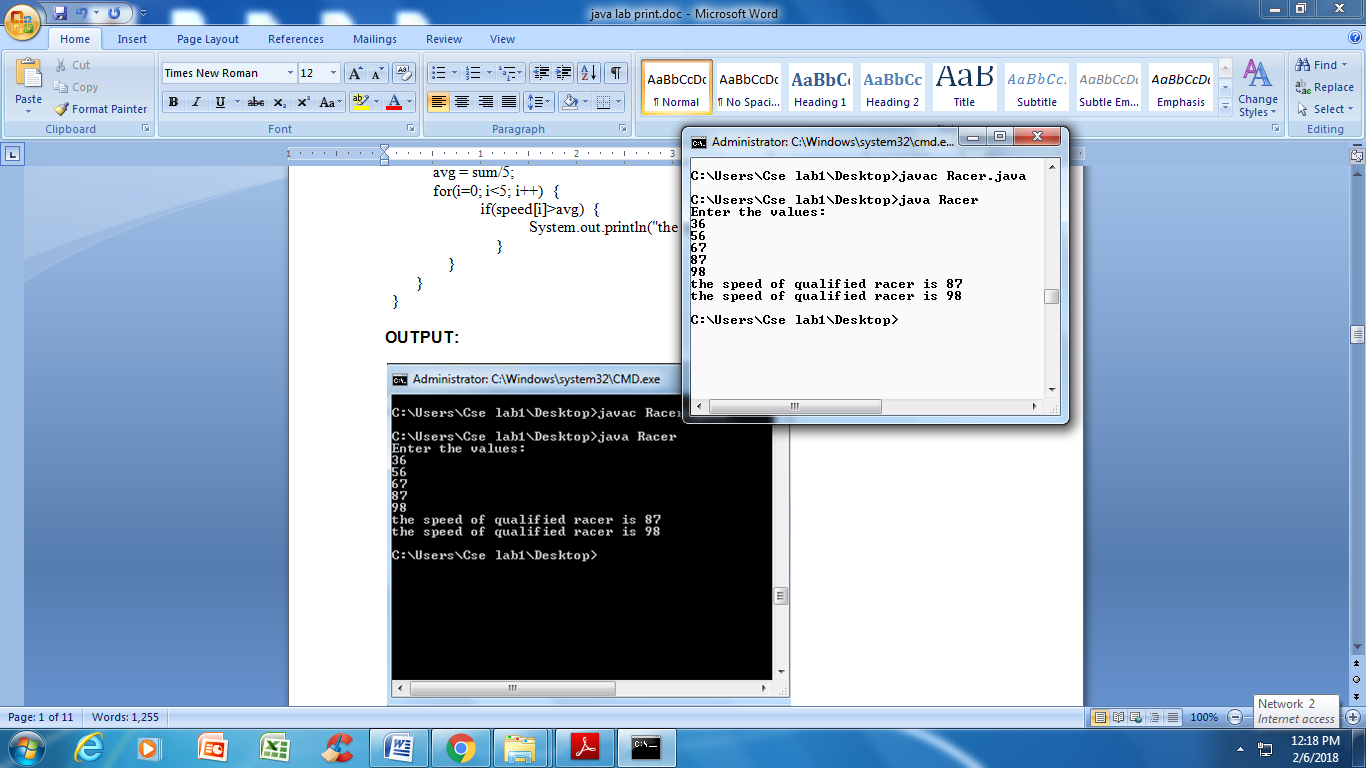
}

}

}

}

**OUTPUT:**



**1 d) Write a case study on public static void main(250 words)**

The **public** keyword is an access specifier, which allows the programmer to control the visibility of class members. When a class member is preceded by public, then that member may be accessed by code outside the class in which it is declared. (**The opposite of public is private, which prevents a member from being used by code defined outside)**

In this case, **main( )** must be declared as **public**, since it must be called by code outside of its class when the program is started. The keyword **static** allows **main( )** to be called without having to instantiate a particular instance of the class. This is necessary since main( ) is called by the Java interpreter before any objects are made. The keyword **void**simply tells the compiler that main( ) does not return a value. As you will see, methods may also return values.

As stated, **main( )** is the method called when a Java application begins. Keep in mind that Java is case-sensitive. Thus, **Main** is different from **main**. It is important to understand that the Java compiler will compile classes that do not contain a **main( )** method. But the Java interpreter has no way to run these classes. So, if you had typed **Main**instead of **main**, the compiler would still compile your program. However, the Java interpreter would report an error because it would be unable to find the **main( )** method.

Any information that you need to pass to a method is received by variables specified within the set of parentheses that follow the name of the method. These variables are called**parameters**. If there are no parameters required for a given method, you still need to include the empty parentheses. In**main( )**, there is only one parameter, but a complicated one.**String args[ ]** declares a parameter named **args**, which is an array of instances of the class **String**. (Arrays are collections of similar objects.) Objects of type String store character strings. In this case, **args** receives any command-line arguments present when the program is executed.

**Exercise - 2** (Operations, Expressions, Control-flow, Strings)

**2 a). Write a JAVA program to search for an element in a given list of elements using binary search mechanism.**

**AIM:** to implement binary search mechanism

**DESCRIPTION:** Generally, to find a value in unsorted array, we should look through elements of an array one by one, until searched value is found. In case of searched value is absent from array, we go through all elements. In average, complexity of such an algorithm is proportional to the length of the array. Situation changes significantly, when array is sorted. If we know it, random access capability can be utilized very efficiently to find searched value quick. Cost of searching algorithm reduces to binary logarithm of the array length. For reference, log2(1 000 000) ≈ 20. It means, that in worst case, algorithm makes 20 steps to find a value in sorted array of a million elements or to say, that it doesn't present it the array. Note: Elements should be in sorted order in the given array

**ALGORITHM:**

1. start

2. import package

3.create a class declare the array of elements

4. declare the method and perform binary search in which the mid value of the array is calculated and if the key value is less than the mid value then key is searched in the first part of the array otherwise it is searched in second part of the array

5. print success if key is found otherwise print failure

6. end method

7. end class

8. create another class and declare the main method

9. create a object for the first class and call the method

**PROGRAM:**

import java.util.Scanner;

class BinarySearch {

public static void main(String args[]) {

int c, first, last, middle, n, search, array[];

Scanner sc= new Scanner(System.in);

System.out.println("Enter the numberof elements:");

n = sc.nextInt();

array = new int[n];

System.out.println("enter "+n+" integers:");

for(c=0;c<n;c++)

array[c]=sc.nextInt();

System.out.println("enter a value to search");

search = sc.nextInt();

first =0;

last=n-1;

middle = (first+last)/2;

while(first<=last) {

if(array[middle]<search)

first = middle+1;

else if(array[middle]==search) {

System.out.println(search +" found at location "+(middle+1));

break; }

else

last= middle-1;

middle=(first+last)/2;

}

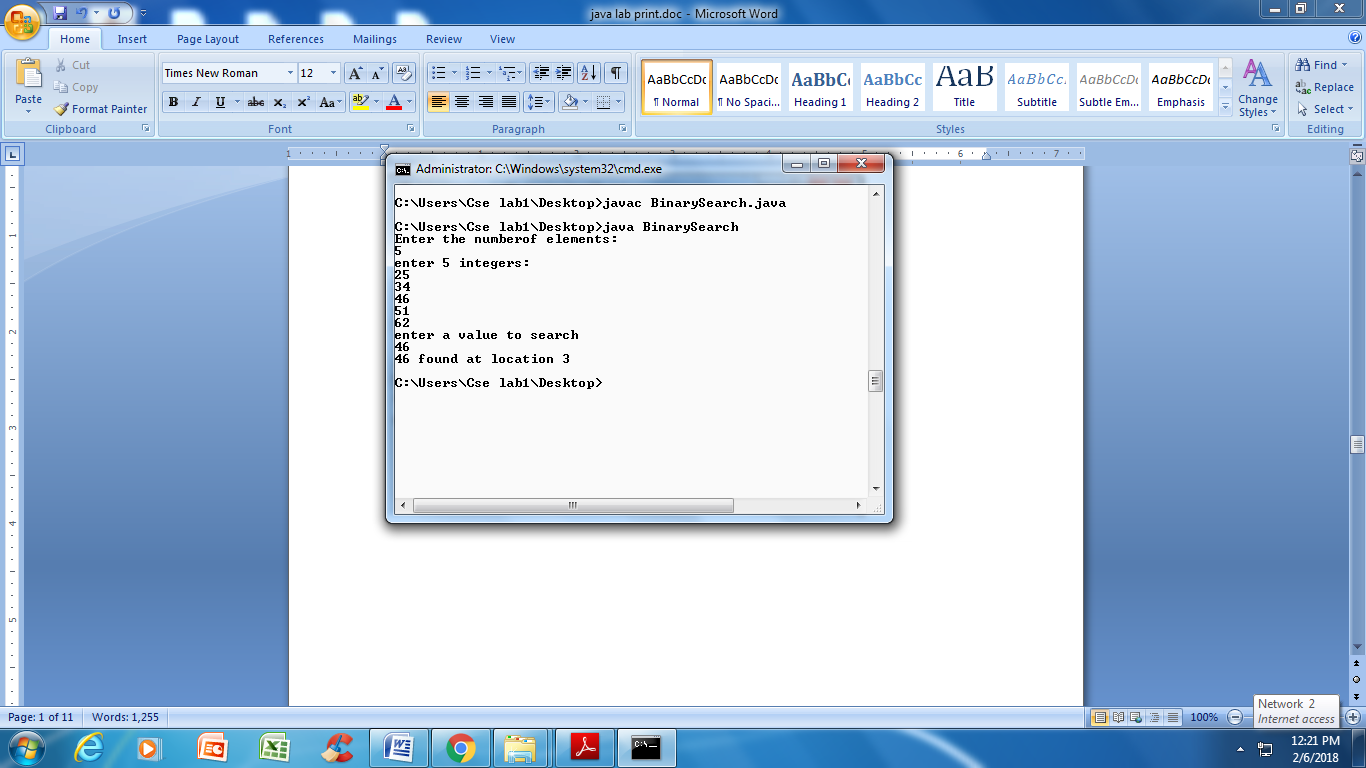
if(first>last)

System.out.println("element not found");

}

}

**OUTPUT:**



**2 b). Write a JAVA program to sort for an element in a given list of elements using bubble sort**

**AIM:** to implement bubble sort mechanism

**DESCRIPTION:** Bubble sort is one of the simplest sorting techniques in java to sort the array elements. The idea is to traverse from the starting element to the last one by comparing the adjacent elements and swapping them if they are not in the specific order.

**ALGORITHM:**

**1.Import package.**

**2. Create Bubble sort class**

**3. Declare a main method and write logic for sorting elements in array using bubble sort techniques.**

**PROGRAM:**

import java.util.Scanner;

class BubbleSort {

public static void main(String args[]) {

int n, i, j, temp, array[];

Scanner sc= new Scanner(System.in);

System.out.println("Enter the numberof elements to sort:");

n = sc.nextInt();

array=new int[n];

System.out.println("enter "+n+" integers:");

for(i=0;i<n;i++)

array[i]=sc.nextInt();

for(i=0;i<(n-1); i++) {

for(j=0;j<(n-i-1); j++) {

if(array[j]>array[j+1]) {

temp = array[j];

array[j] = array[j+1];

array[j+1] = temp;

}

}

}

System.out.println("sorted list of elements:");

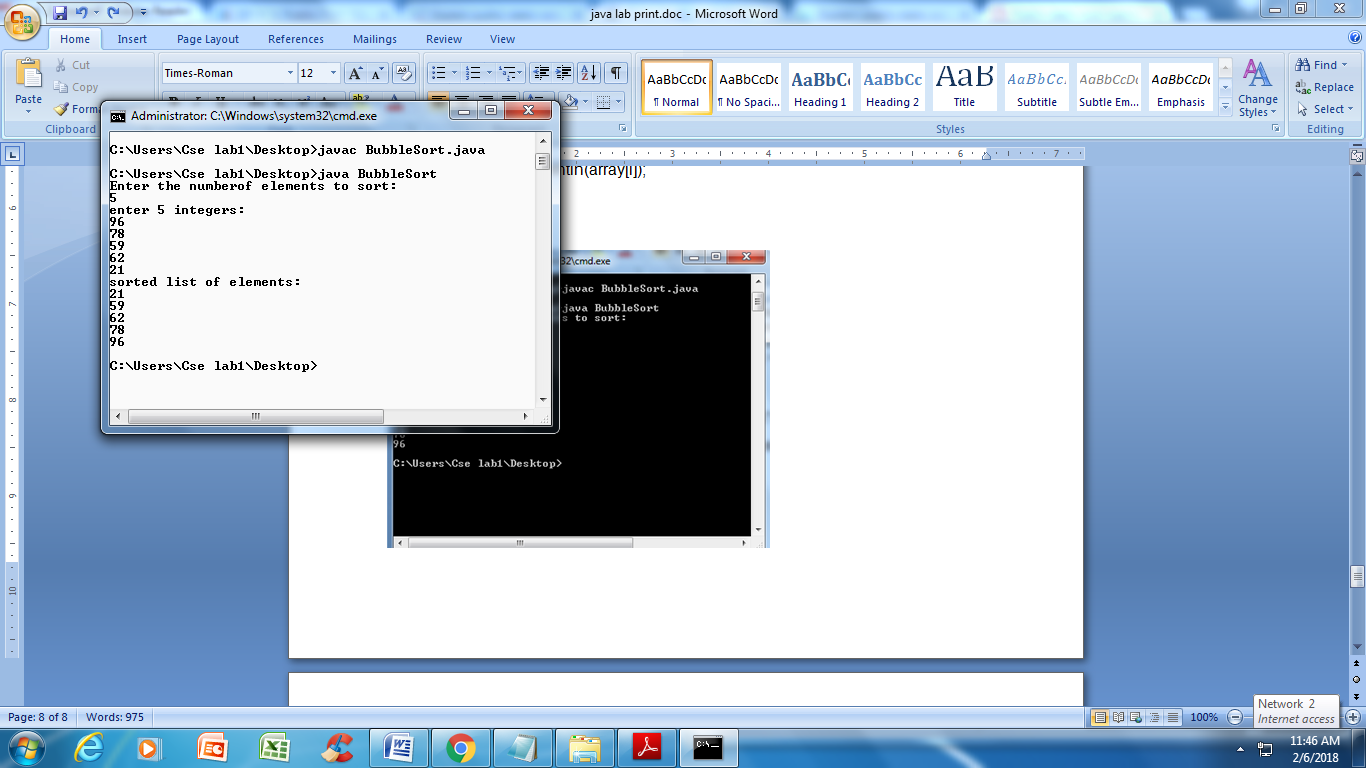
for(i=0;i<n;i++)

System.out.println(array[i]);

}

}

**OUTPUT:**



**2 (c). Write a JAVA program to sort for an element in a given list of elements using merge sort.**

**AIM:** to implement merge sort mechanism

**DESCRIPTION:** Merge sort is a “divide and conquer” algorithm wherein we first divide the problem into sub problems. When the solutions for the sub problems are ready, we combine them together to get the final solution to the problem.

**ALGORITHM:**

1. Import packages
2. Create class MergeSort
3. Create a function Sort and initialize variables inside function are array a[ ], low, high.
4. Merge two sorted sub arrays and declare main method.

**PROGRAM:**

import java.util.Scanner;

public class MergeSort {

public static void sort(int[] a, int low, int high) {

int n = high - low;

if (n <= 1)

return;

int mid = low + n/2;

sort(a, low, mid);

sort(a, mid, high);

// merge two sorted subarrays

int temp[] = new int[n];

int i = low, j = mid;

for (int k = 0; k < n; k++) {

if (i == mid)

temp[k] = a[j++];

else if (j == high)

temp[k] = a[i++];

else if (a[j]<a[i])

temp[k] = a[j++];

else

temp[k] = a[i++];

}

for (int k = 0; k < n; k++)

a[low + k] = temp[k];

}

// Main method

public static void main(String[] args) {

Scanner sc = new Scanner( System.in );

int n, i;

System.out.println("Enter number of integer elements");

n = sc.nextInt();

int arr[] = new int[n];

System.out.println("\nEnter "+ n +" integer elements");

for (i = 0; i < n; i++)

arr[i] = sc.nextInt();

/\* Call method sort \*/

sort(arr, 0, n);

System.out.println("\nElements after sorting ");

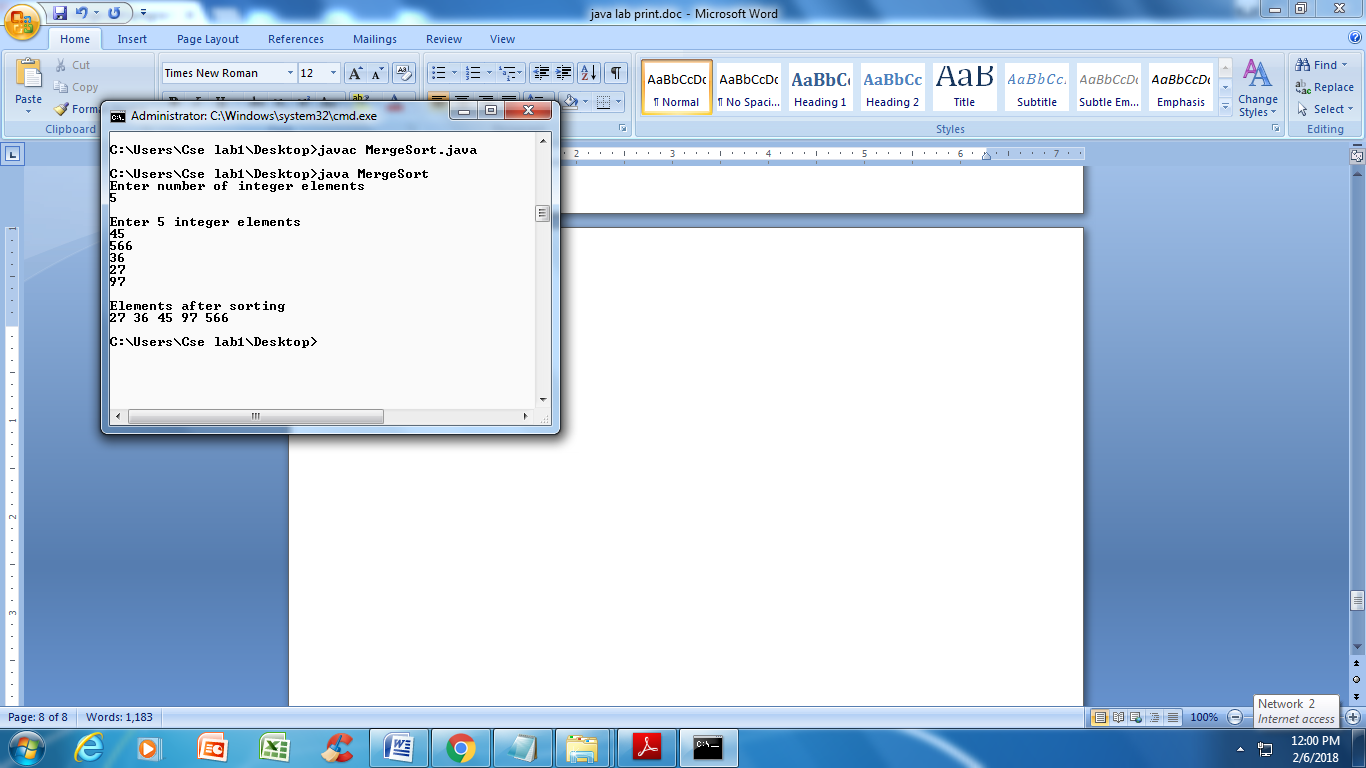
for (i = 0; i < n; i++)

System.out.print(arr[i]+" ");

}

}

**OUTPUT:**



**2 (d) Write a JAVA program using StringBufferto delete, remove character.**

**AIM:** to illustrate delete() method in StringBuffer class

**DESCRIPTION:** StringBuffer.delete() is an inbuilt method in java which is used to remove or delete the characters in a substring of this sequence. The substring starts at a specified index\_start\_point and extends to the character ate the index\_end\_point.

**ALGORITHM:**

**1.** Create DeleteMethod class

2. Declare main method and created object for StringBuffer and assigned index values to remove one or more characters or all characters.

**PROGRAM:**

public class DeleteMethod {

public static void main(String[] args) {

// to remove one character

StringBuffer str = new StringBuffer("madam");

str.deleteCharAt(2);

System.out.println(str);

// to remove more than one character

StringBuffer str1 = new StringBuffer("new delhi");

str1.delete(0,4);

System.out.println(str1);

// to remove all characters

StringBuffer str2 = new StringBuffer("madam");

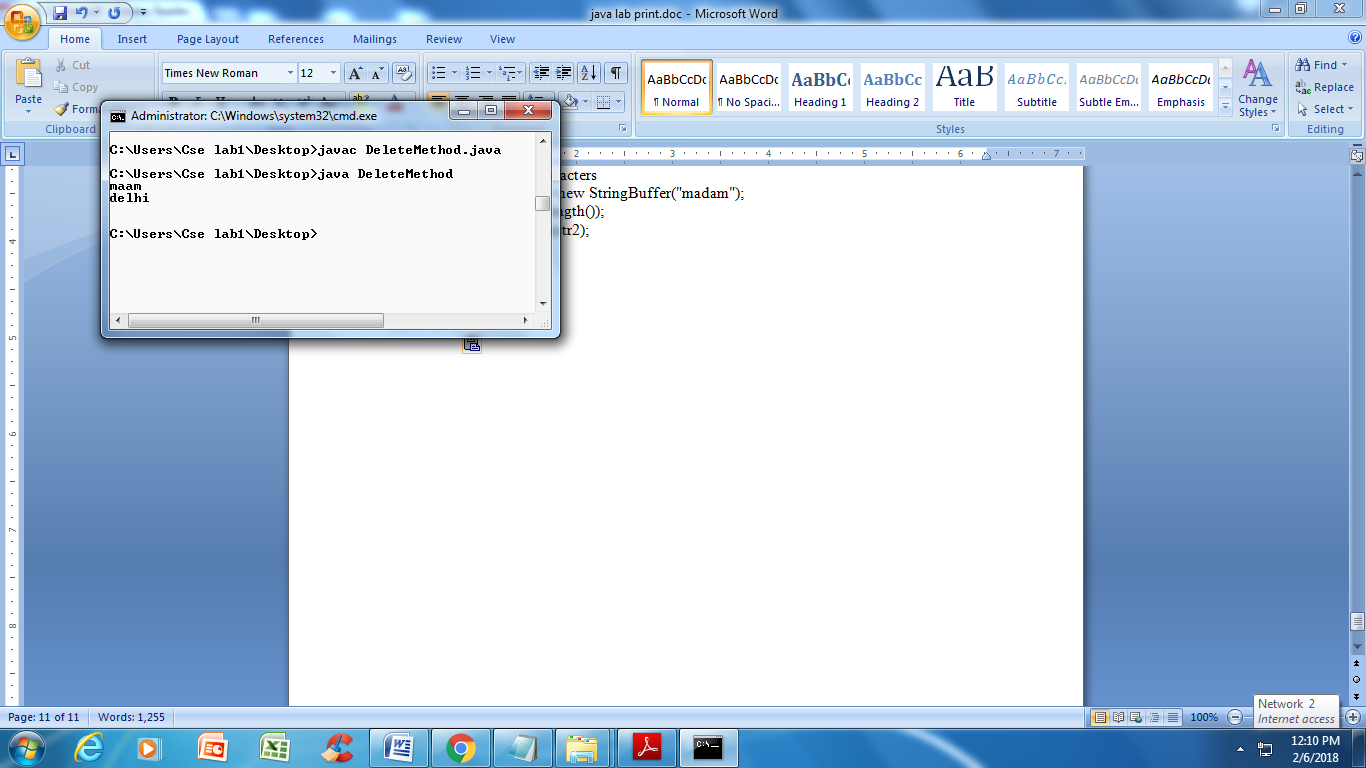
str2.delete(0,str2.length());

System.out.println(str2);

}

}

**OUTPUT:**



**Exercise - 3** (Class, Objects)

**3 a). Write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method.**

**AIM:** to implement class mechanism

**DESCRIPTION:** A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

A method in Java or Java Method is a collection of statements that perform some specific task and return the result to the caller. A Java method can perform some specific task without returning anything. Methods in Java allow us to reuse the code without retyping the code.

**ALGORITHM:**

Step1: Create a class “Circle”.

Step2: Declare two variables, radius and pi(i.e.,provide pi value=3.4285).

Step3: create method “area()”.

Step4: “a” is assigned to area of circle formula(pi\*r\*r) and Print area of circle.

Step5: create method “circum()”.

Step6: “cir” is assigned to circumference formula(2\*pi\*r) and print Circumference.

Step7: In the main method create a new object for the class circle using new operator.

Step8: call the methods by using the created object .

**PROGRAM:**

import java.util.Scanner;

class Circle {

float radius;

double pi= 3.4285;

void area() {

double a = pi\*radius\*radius;

System.out.println("area of circle: "+a);

}

void circum() {

double cir = 2\*pi\*radius;

System.out.println("diameter of circle: "+cir);

}

}

class Circle1 {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

System.out.println("enter the radius:");

Circle obj= new Circle();

obj.radius = sc.nextFloat();

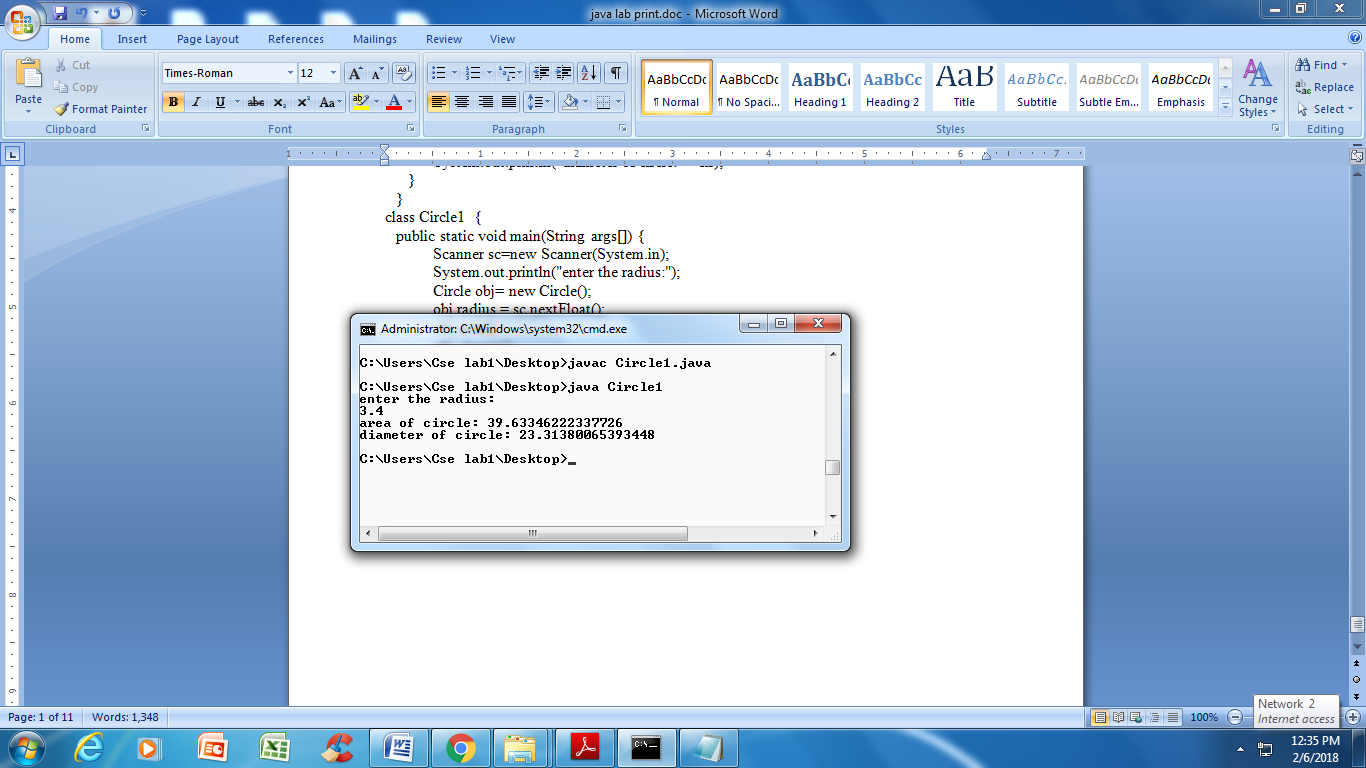
obj.area();

obj.circum();

}

}

**OUTPUT:**

****

**3 b). Write a JAVA program to implement constructor.**

**AIM:** to implement constructor.

**DESCRIPTION:** In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object.

**ALGORITHM:**

Step1: Create a Class “ Room”

Step2:declare length,breadth and height.

Step3: create a constructor(Constructor and class name must be same)

Step4: Assign values for length,breadth and height

Step5: Create a method volume and return volume of room(Length\*breadth\*height)

Step6: Create a class “Box” and Constructor “Box”

Step7: Box is a parameterized constructor here, and return volume of box.

Step8: Create a Class “Test” and Main method

Step9: Create object by using new operator and call Room() and Box().

Step10: As Box() is parameterized constructor, pass the values and print Volumes of Room & Box .

**PROGRAM:**

class Room {

double length, breadth, height;

Room() {

length = 14;

breadth = 12;

height = 10;

}

double volume() {

return length\*breadth\*height;

}

}

class Box {

double length, breadth, height;

Box(double l, double b, double h) {

length = l;

breadth = b;

height = h;

}

double volume() {

return length\*breadth\*height;

}

}

class Test {

public static void main(String args[]) {

Room r = new Room();

System.out.println("the volume of the room is: "+r.volume());

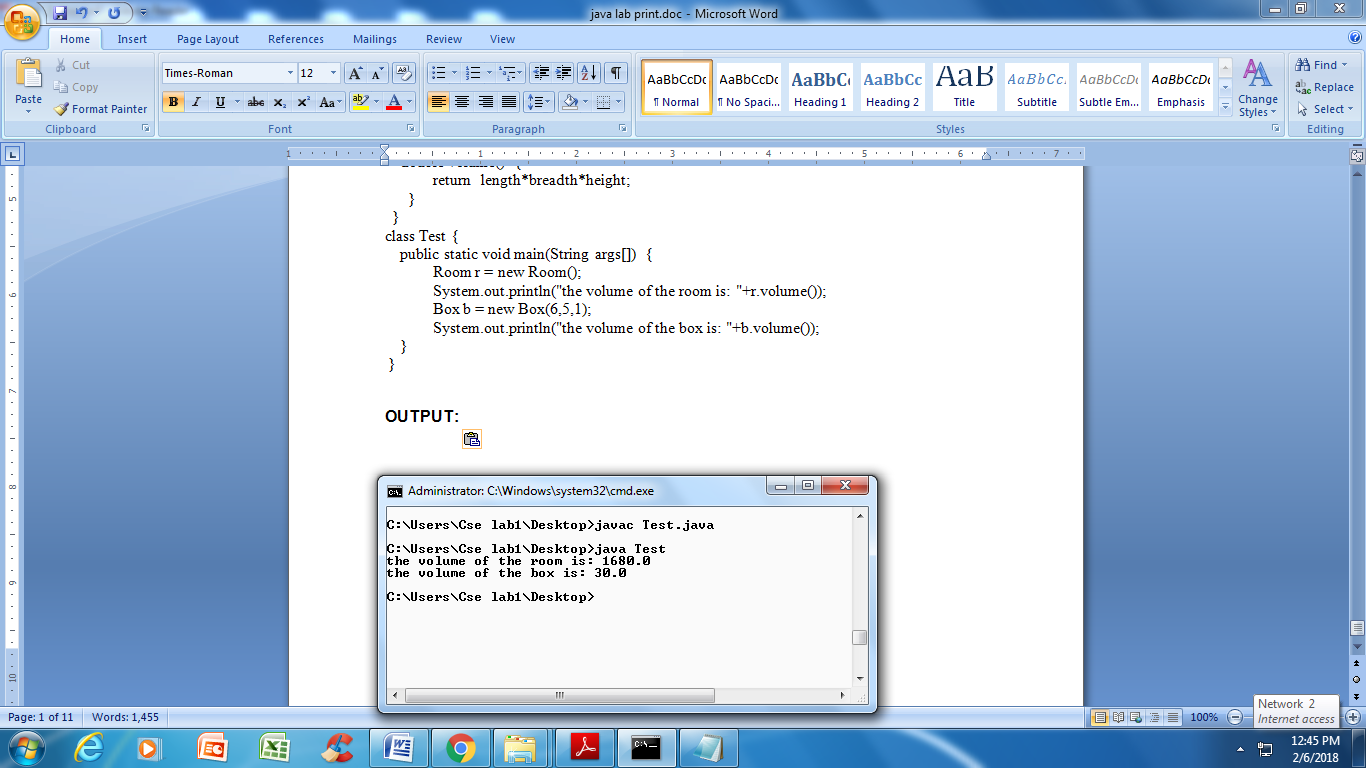
Box b = new Box(6,5,1);

System.out.println("the volume of the box is: "+b.volume());

}

}

**OUTPUT:**

****

**Exercise - 4** (Methods)

**4 a). Write a JAVA program to implement constructor overloading.**

**AIM:** to implement constructor overloading.

**DESCRIPTION:** Constructor overloading that allows a class to have more than one constructor having different argument lists. When overload method is called, java looks for a match between the arguments used to call the method and the method parameters. Finally it matches and displays the output.

**ALGORITHM:**

1. Create a class Box

2. Write two constructors and make constructor overloading-Box(), Box(double len), Box(double w, double h, double d),

3. Create an instance and call members called constructors.

**PROGRAM:**

class Box {

double width, height, depth;

Box(double w, double h, double d) {

width = w;

height = h;

depth = d;

}

Box() {

width = height = depth = 0;

}

Box(double len) {

width = height = depth = len;

}

double volume() {

return width\*height\*depth;

}

}

class Test1 {

public static void main(String args[]) {

Box b1 = new Box(10,20,15);

Box b2 = new Box();

Box b3 = new Box(7);

double vol;

vol = b1.volume();

System.out.println("the volume of the box1 is: "+b1.volume());

vol = b2.volume();

System.out.println("the volume of the box2 is: "+b2.volume());

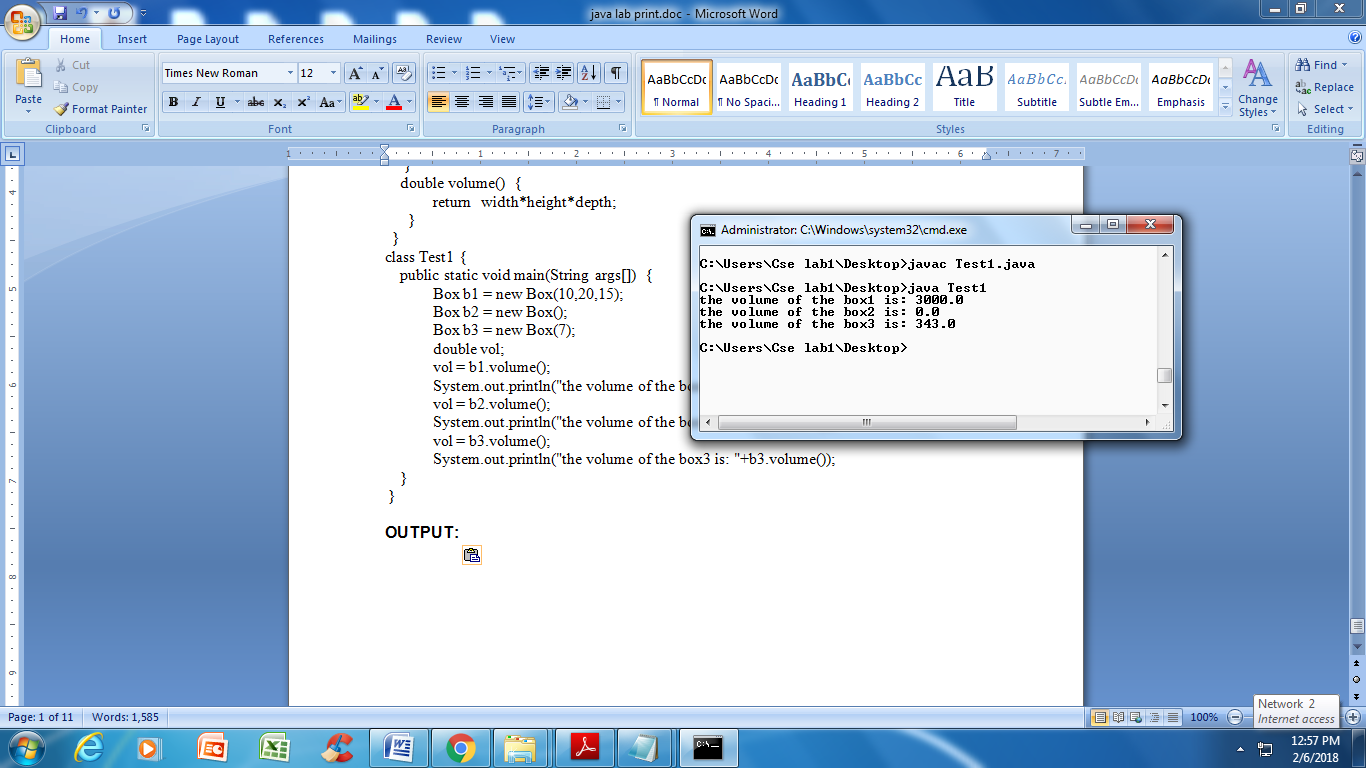
vol = b3.volume();

System.out.println("the volume of the box3 is: "+b3.volume());

}

}

**OUTPUT:**



**4 b). Write a JAVA program implement method overloading.**

**AIM:** to implement method overloading.

**DESCRIPTION:** Method overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. When overload method is called, java looks for a match between the arguments used to call the method and the method parameters. Finally it matches and displays the output.

**ALGORITHM:**

1. Create a class Overload

2. Write methods called called max(float a, float b), void max(double a, double b), and void max(long a, long b) and make method overloading.

3. Create an instance and call members called methods.

**PROGRAM:**

class Overload {

void max(float a, float b) {

System.out.println("max method with float argument invoked");

if(a>b)

System.out.println(a+" is greater");

else

System.out.println(b+" is greater");

}

void max(double a, double b) {

System.out.println("max method with double argument invoked");

if(a>b)

System.out.println(a+" is greater");

else

System.out.println(b+" is greater");

}

void max(long a, long b) {

System.out.println("max method with float argument invoked");

if(a>b)

System.out.println(a+" is greater");

else

System.out.println(b+" is greater");

}

public static void main(String args[]) {

Overload ol= new Overload();

ol.max(23L,12L);

ol.max(2,3);

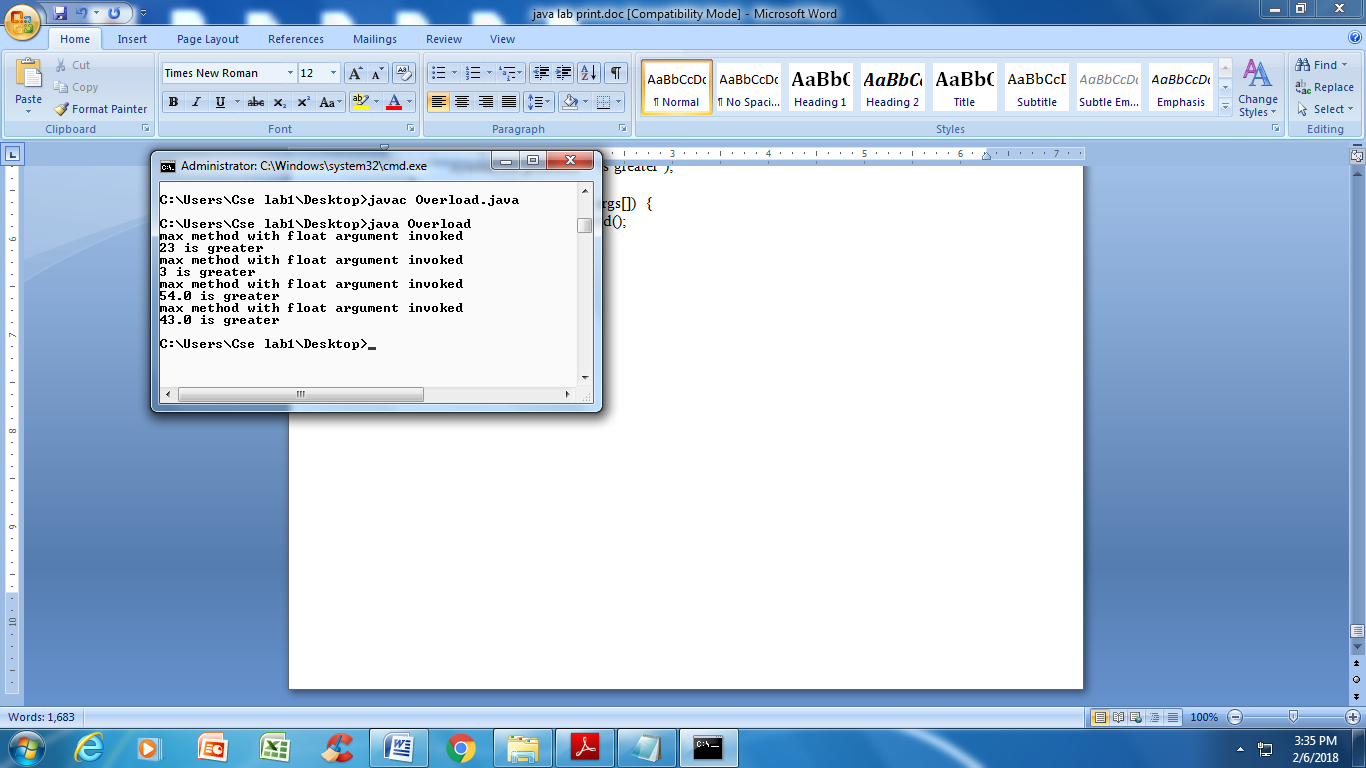
ol.max(54,.35f);

ol.max(43f,35f);

}

}

**OUTPUT:**



**Exercise - 5** (Inheritance)

**5 a). Write a JAVA program to implement Single Inheritance**

**AIM:** to implement single inheritance

**DESCRIPTION:** Inheritance is way to generate “IS-A” relationship between Parent and Child by accessing the properties and methods. It increases readability and reusability of code. It can be implemented in many ways like Simple, Multi-level, Hybrid.

**ALGORITHM:**

1. Create a class named Square Declare members like length,breadth and write a constructor
2. Create a class named Cube declare members like length,breadth and height.

2. Extends Cube class from Square class

4. Create an instance to sub class called Inherent and access members using instance

**PROGRAM:**

import java.util.Scanner;

class Square {

int length, breadth;

public void get(int x, int y) {

length =x;

breadth = y;

}

int area() {

return (length\*breadth);

}

}

class Cube extends Square {

int height;

public void getdata(int x, int y, int z) {

get(x,y);

height=z;

}

int volume() {

return (length \* breadth \* height);

}

}

public class Inherit {

public static void main(String args[]) {

Cube c =new Cube();

Scanner sc=new Scanner(System.in);

System.out.println("enter length,breadth and height: ");

int l=sc.nextInt();

int b=sc.nextInt();

int h=sc.nextInt();

c.getdata(l,b,h);

int b1= c.area();

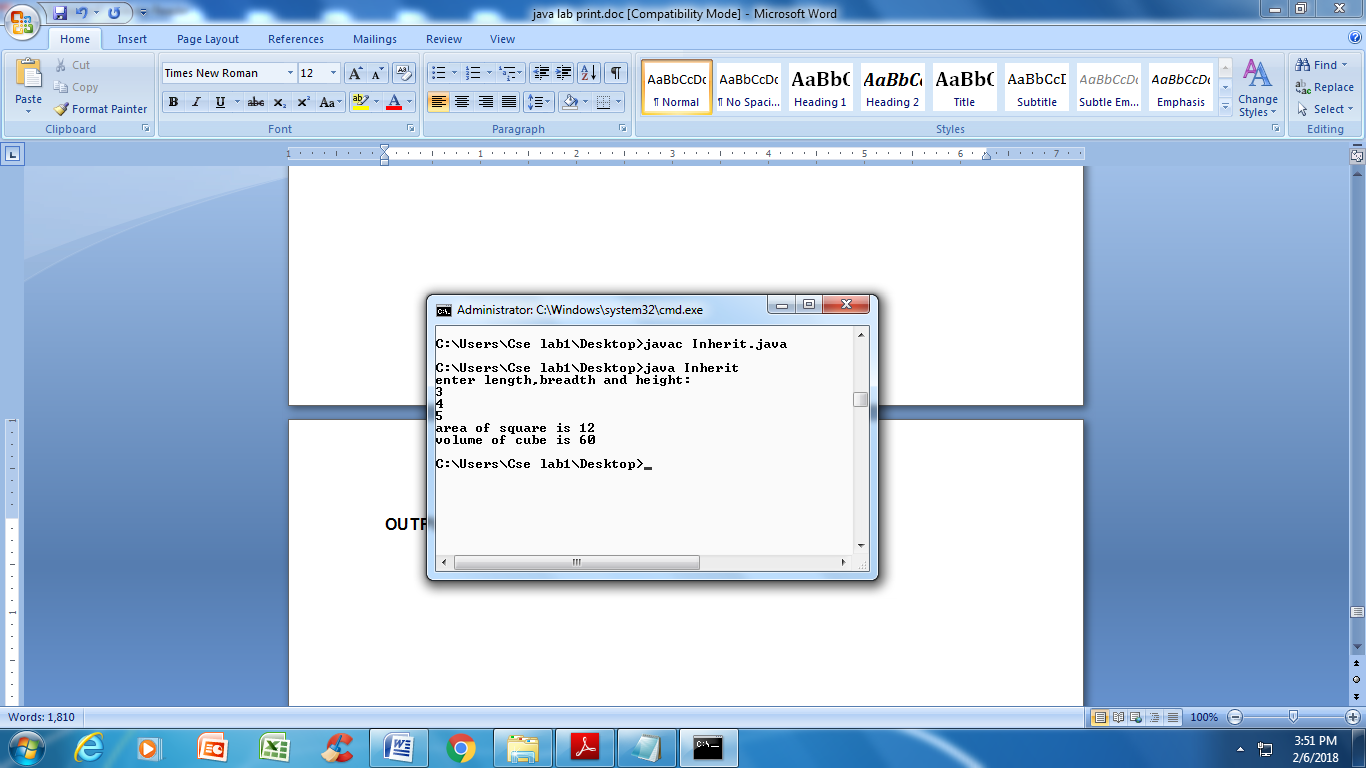
System.out.println("area of square is "+b1);

int b2= c.volume();

System.out.println("volume of cube is "+b2);

} }

**OUTPUT:**



**5 b). Write a JAVA program to implement multi level Inheritance**

**AIM:** to implement multi level inheritance.

**DESCRIPTION:** Inheritance is one way of implementing super-sub class relationship, for data usability and readability. Implementing Multi-level inheritance is better one for Objects whose data will be accessed by many objects thereby increasing readability.

CLASS A

CLASS B

CLASS C

**ALGORITHM:**

1. Create a class named Account

2. Extend SavingsAcc class from Account class

3. Extend Accountdetails class from SavingsAcc class

4. Create an instance to sub class called Accountdetails and access members of both SavingsAcc and Accountdetails using instance.

**PROGRAM:**

import java.util.Scanner;

class Account {

String cust\_name;

int acc\_no;

void display1( ) {

System.out.println("Customer name: "+ cust\_name);

System.out.println("Account No: "+ acc\_no);

}

}

class SavingAcc extends Account {

int min\_bal, saving\_bal;

void display2( ) {

System.out.println("Minimum Balance: " + min\_bal);

System.out.println("Saving Balance: "+ saving\_bal);

}

}

class AccDetails extends SavingAcc {

int deposits, withdrawls;

void display3( ) {

System.out.println(" Deposit: "+ deposits);

System.out.println(" withdrawls: "+ withdrawls);

}

}

class Multilevel {

public static void main(String args[ ]) {

Scanner sc = new Scanner(System.in);

System.out.println("enter customer\_name, Account\_no, minimum\_balance, saving\_balance, deposit and withdrawl");

AccDetails a = new AccDetails( );

a.cust\_name=sc.next( );

a.acc\_no=sc.nextInt( );

a.min\_bal = sc.nextInt( );

a.saving\_bal = sc.nextInt( );

a.deposits = sc.nextInt( );

a.withdrawls = sc.nextInt( );

a.display1( );

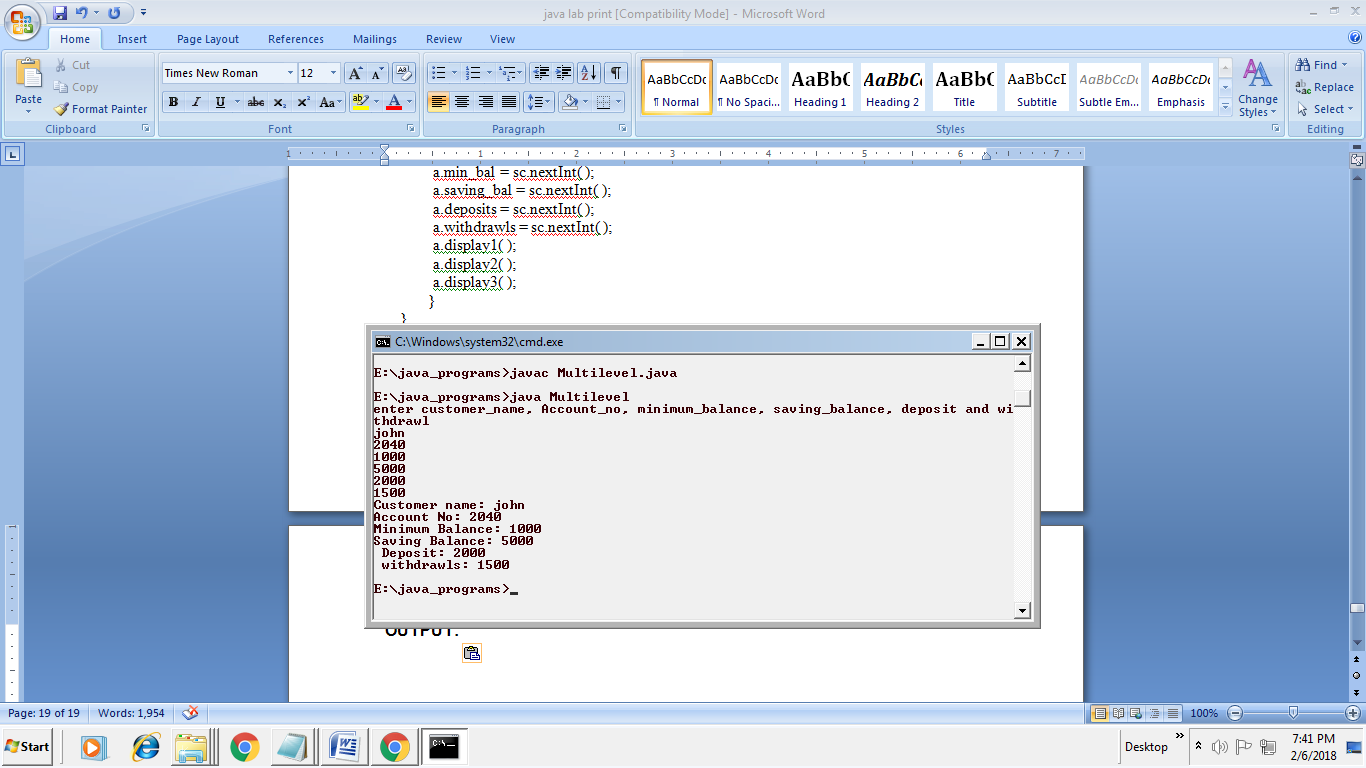
a.display2( );

a.display3( );

}

}

**OUTPUT:**



**5 c). Write a java program for abstract class to find areas of different shapes**

**AIM:** to implement multi level inheritance

**ALGORITHM:**

1. Create a abstract class named Shape

2. Extend Rectangle class from Shape class

3. Extend Triangle class from Shape class

4. Extend Ellipse class from Shape class

5.Create a Area class to find areas of different shapes.

4. Create an instance to sub class Rectangle, Triangle and Ellipse access members of Rectangle, Triangle and Ellipse using instance.

**PROGRAM:**

import java.util.Scanner;

abstract class Shape {

double dim1;

double dim2;

Shape(double a, double b) {

dim1 = a;

dim2 = b;

}

abstract double area();

}

class Rectangle extends Shape {

Rectangle(double a, double b) {

super(a, b);

}

double area() {

return dim1 \* dim2;

}

}

class Triangle extends Shape {

Triangle(double a, double b) {

super(a, b);

}

double area() {

return dim1 \* dim2 / 2;

}

}

class Ellipse extends Shape {

Ellipse(double a, double b) {

super(a, b);

}

double area() {

return 3.14159\*dim1 \* dim2;

}

}

public class Area {

public static void main(String args[]) {

Scanner sc= new Scanner(System.in);

int ch;

do {

System.out.println("1. RECTANGLE");

System.out.println("2. TRIANGLE");

System.out.println("3. ELLIPSE");

System.out.println("Enter your choice: ");

ch = sc.nextInt( );

switch(ch) {

case 1:

System.out.println("enter two dimensions for area of a rectangle ");

double a = sc.nextDouble( );

double b = sc.nextDouble( );

Rectangle d1= new Rectangle(a,b);

System.out.println("area of rectangle is "+ d1.area( ));

break;

case 2:

System.out.println("enter two dimensions for area of a triangle ");

double c = sc.nextDouble( );

double d = sc.nextDouble( );

Triangle d2= new Triangle(c,d);

System.out.println("area of triangle is "+ d2.area( ));

break;

case 3:

System.out.println("enter two dimensions for area of a elllipse ");

double e = sc.nextDouble( );

double f = sc.nextDouble( );

Ellipse d3= new Ellipse(e,f);

System.out.println("area of ellipse is "+ d3.area( ));

break;

default:

System.out.println("enter correct choice from 1 to 3");

break;

}

System.out.println("press 1 to continue or 0 to exit:");

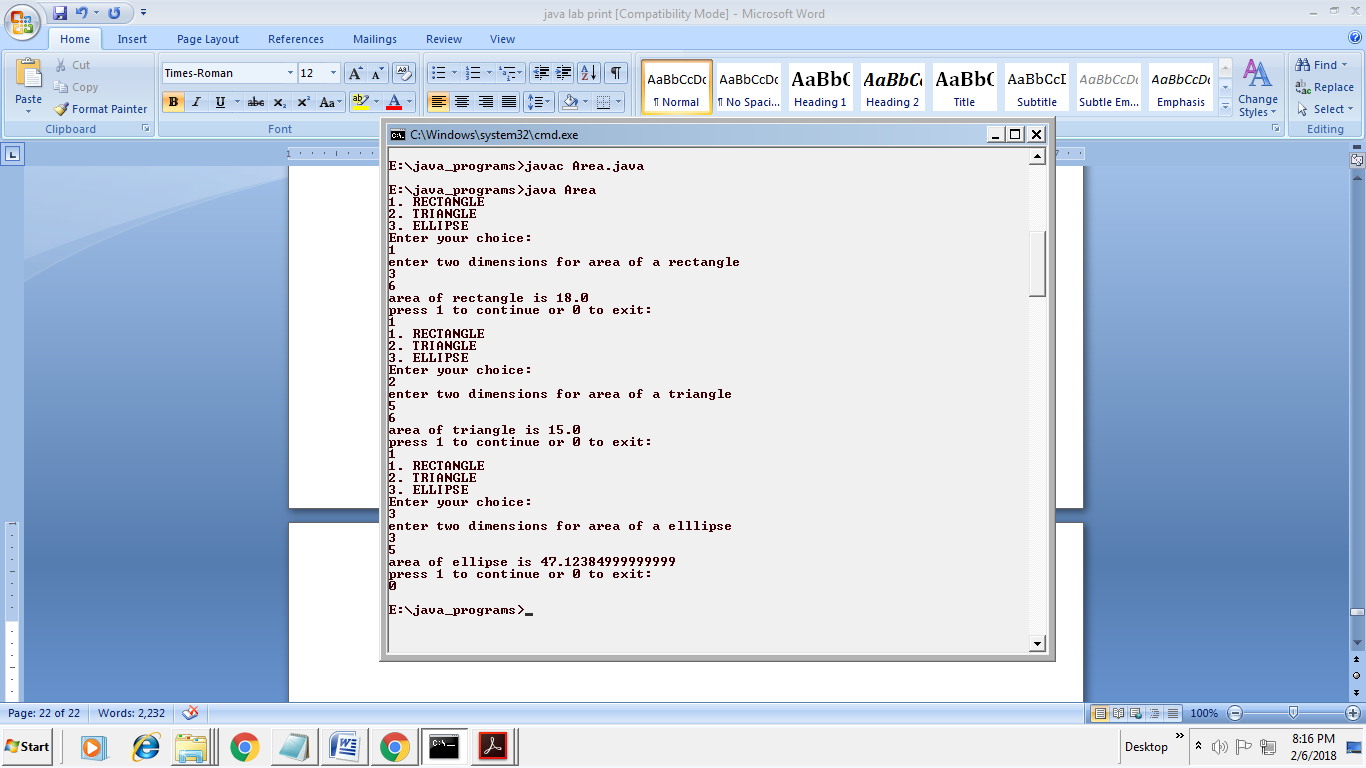
ch = sc.nextInt( );

} while(ch == 1);

}

}

**OUTPUT:**

****

**Exercise - 6** (Inheritance - Continued)

**6 a). Write a JAVA program give example for “super” keyword.**

**AIM:** JAVA program with “super ” keyword

**DESCRIPTION:** Reusability is very important feature in Inheritance, where accessing base class properties and methods is needed. Super keyword is such object which do the job, by handling the super class properties, methods and constructors. Invoking super keyword can be done in following way: Variable◊super.variable\_name; Methods◊super.method\_name(); Constructor◊super(parameter\_list).

**ALGORITHM:**

1. Create a class named Box

2. Extends BoxWeight class from Box class

3. Aceess members of super class using super keyword super.variablename; super.methodname(); super(parameter list)

4. Create an instance to sub class called BoxWeight and access members using instance.

**PROGRAM:**

class Box {

private double width;

private double height;

private double depth;

Box(Box ob) {

width = ob.width;

height = ob.height;

depth = ob.depth;

}

Box(double w, double h, double d) {

width = w;

height = h;

depth = d;

}

Box(double len) {

width=height=depth=len;

}

double volume( ) {

return width\*height\*depth;

}

}

class BoxWeight extends Box {

double weight;

BoxWeight(BoxWeight ob) {

super(ob);

weight = ob.weight;

}

BoxWeight(double w, double h, double d, double m) {

super(w,h,d);

weight = m;

}

BoxWeight(double len,double m) {

super(len);

weight = m;

}

}

class DemoSuper {

public static void main(String args[ ]) {

BoxWeight mybox1=new BoxWeight(10,20,15,34.3);

BoxWeight mybox2=new BoxWeight(2,3,4,0.076);

BoxWeight mycube=new BoxWeight(3,2);

BoxWeight myclone=new BoxWeight(mybox1);

double vol;

vol = mybox1.volume( );

System.out.println("volume of mybox1 is: "+vol);

System.out.println("height of mybox1 is "+mybox1.weight);

System.out.println( );

vol = mybox2.volume( );

System.out.println("volume of mybox2 is: "+vol);

System.out.println("height of mybox2 is "+mybox2.weight);

System.out.println( );

vol = myclone.volume( );

System.out.println("volume of myclone is: "+vol);

System.out.println("height of myclone is "+myclone.weight);

System.out.println( );

vol = mycube.volume( );

System.out.println("volume of mycube is: "+vol);

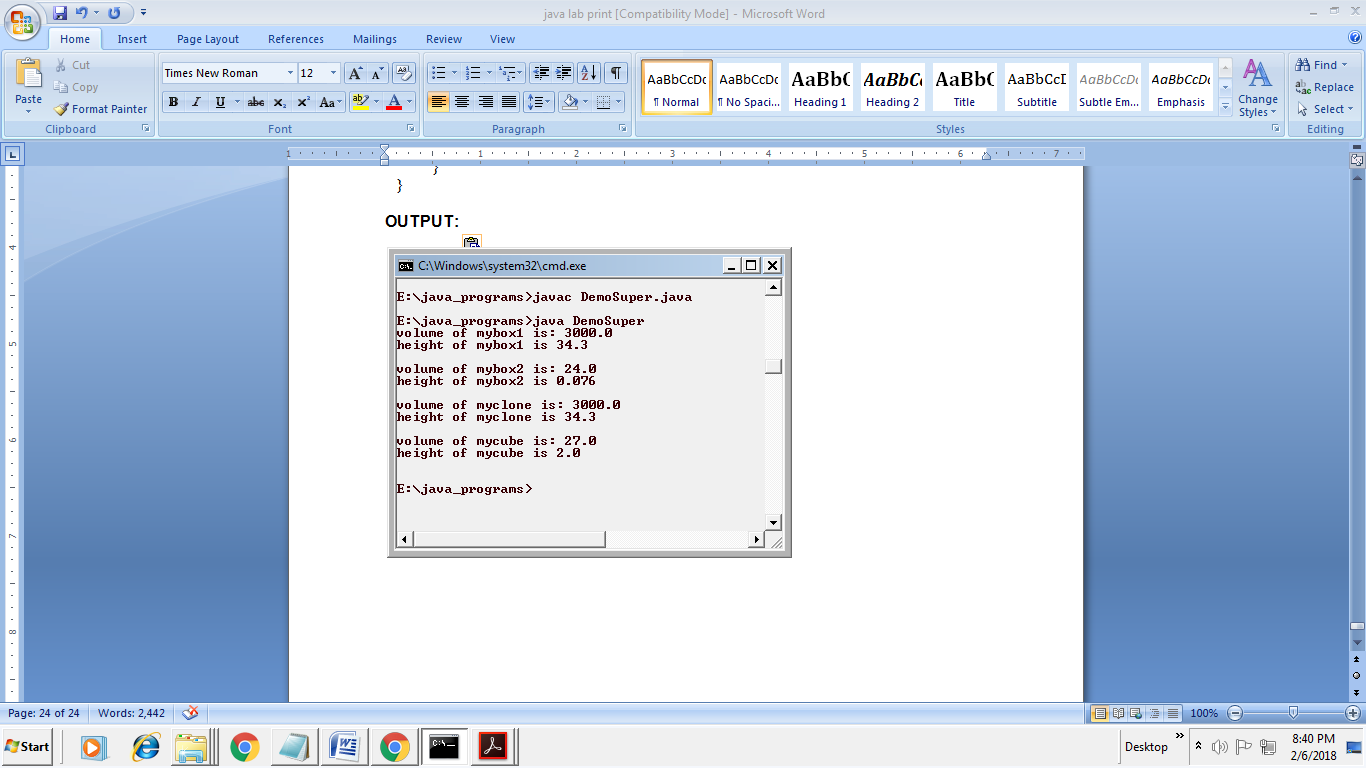
System.out.println("height of mycube is "+mycube.weight);

System.out.println( );

}

}

**OUTPUT:**



**6 b). Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?**

**AIM:** JAVA program to implement Interface

**ABSTRACT:**

1. Create interfaces Vehicleone and Vehicletwo

2. Create class Vehicle implements Vehicleone, Vehicletwo

3. Create methods distance, speed

4. Implement multiple Inheritance Using Interface create a class multiple Inheritance Using Interface

5. Create a new object and call the methods using obj.distance() and obj.speed()

**PROGRAM:**

interface Vehicleone {

int speed = 90;

public void distance( );

}

interface Vehicletwo {

int distance = 100;

public void speed( );

}

class Vehicle implements Vehicleone, Vehicletwo {

public void distance( ) {

int distance = speed\*100;

System.out.println("distance travelled is "+distance);

}

public void speed( ) {

int speed = distance/100;

System.out.println("speed of the vehicle: "+speed);

}

}

class MultipleInheritanceUsingInterface {

public static void main(String args[ ]) {

Vehicle obj= new Vehicle( );

System.out.println("vehicle");

obj.distance( );

obj.speed( );

}

}

**OUTPUT:**



**Exercise - 7** (Exception)

**7 a).Write a JAVA program that describes exception handling mechanism**

**AIM:** JAVA program to implement exception handling mechanism

**DESCRIPTION:** Exception is the run time error. It will be raised at run time. The exception handling in java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained. Checked exceptions − A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions. Unchecked exceptions − An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

**ALGORITHM:**

1. Import required packages.

2. Create a class called ExceptionHandle

3. Read values called num1,num2.

4. Check run time error called ArithmeticException

5. Place statements in try block

6. Catch runtime errors using catch block and print message

**PROGRAM:**

import java.util.Scanner;

public class ExceptionHandle {

public static void main(String args[ ] ) {

Scanner sc = new Scanner(System.in);

System.out.print("enter first number: ");

int num1=sc.nextInt( );

System.out.print("enter second number: ");

int num2=sc.nextInt( );

try {

float result = num1/num2;

System.out.println("Division result of "+num1+"/"+num2+"="+result);

}

catch(ArithmeticException e) {

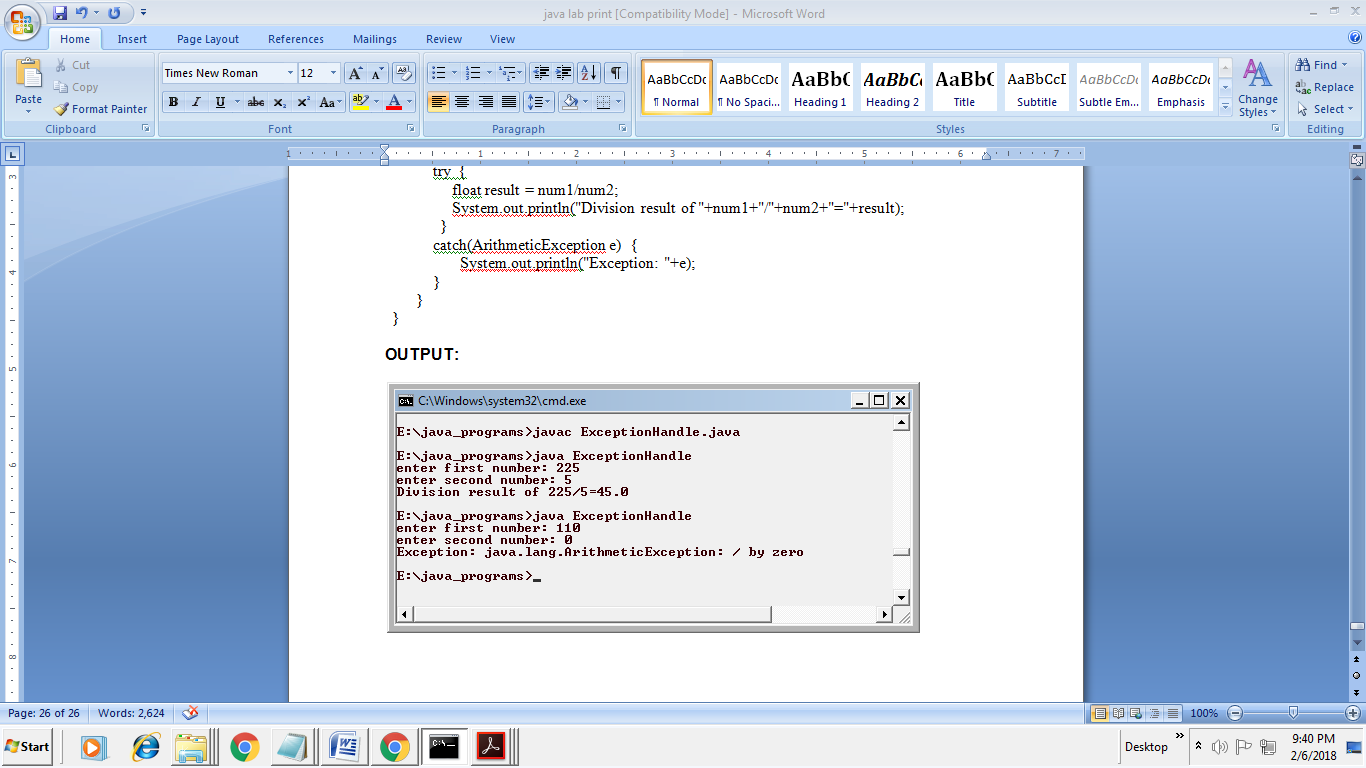
System.out.println("Exception: "+e);

}

}

}

**OUTPUT:**



**7 b).Write a JAVA program Illustrating Multiple catch clauses**

**AIM:** JAVA program Illustrating Multiple catch clauses

**DESCRIPTION:** In multiple catch blocks exception classes order should be from sub classes to super classes. Try {//body } Catch(FileNotFoundException f){} Catch(EOFException e){} Catch(IOException i){}

**ALGORITHM:**

1. start

2. create class 1 extending from Exception base class.

declare a constructor for class1 through which constructor of base class is called.

end class 1.

3. create class 2 write a method and place the statements of exception in try block. write multiple catch blocks for exceptions occurred, give precedence to subclasses first and then base class Exception.

print the message using getMessage() method.

end class2.

1. declare main class give main method as public static void main(String[] args) create object to class 2 call method declared in class 2 using object.

end main()

**PROGRAM:**

// Demonstrate multiple catch statements.

class MultiCatch {

public static void main(String args[]) {

try {

int n = args.length;

System.out.println("n = " + n);

int a = 42 / n;

System.out.println("a = " + a);

int b[ ] = {10,20,30};

b[50] = 100;

}

catch(ArithmeticException e) {

System.out.println(e);

System.out.println("please pass data while running this program");

}

catch(ArrayIndexOutOfBoundsException aie) {

aie.printStackTrace( );

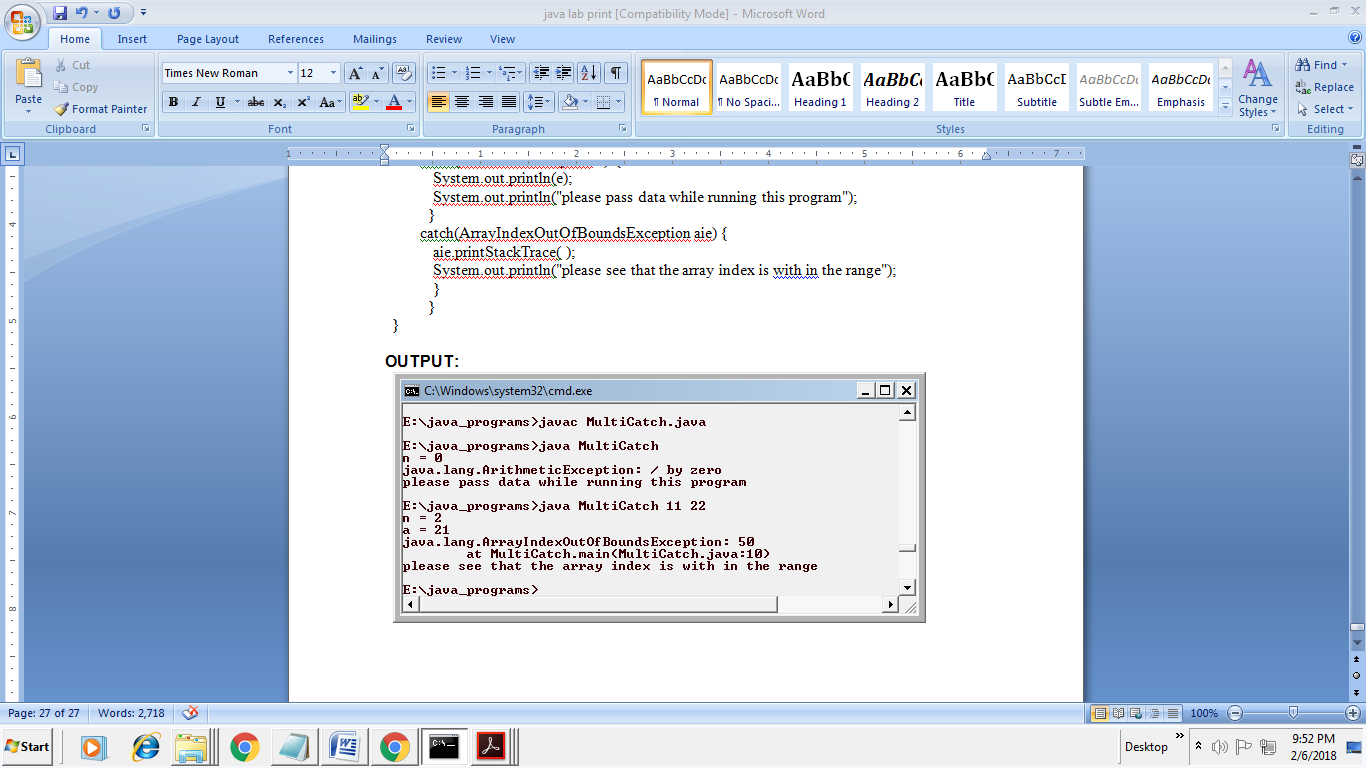
System.out.println("please see that the array index is with in the range");

}

}

}

**OUTPUT:**



**Exercise – 8 (**Runtime Polymorphism)

**8 a). Write a JAVA program that implements Runtime polymorphism**

**AIM:** JAVA program to implement Runtime polymorphism

**ALGORITHM:**

1. Create a class named figure

2. Extends Rectangle class from class Figure

3. Extends Triangle class from class Figure

4. At last find the create the class Find Area

**PROGRAM:**

class Figure {

double dim1;

double dim2;

Figure(double a, double b) {

dim1 = a;

dim2 = b;

}

double area() {

System.out.println("Area for figure is undefined ");

return 0;

}

}

class Rectangle extends Figure {

Rectangle(double a, double b) {

super(a, b);

}

// override area for rectangle

double area() {

System.out.println("Inside Area for Rectangle.");

return dim1 \* dim2;

}

}

class Triangle extends Figure {

Triangle(double a, double b) {

super(a, b);

}

// override area for right triangle

double area() {

System.out.println("Inside Area for Triangle.");

return dim1 \* dim2 / 2;

}

}

class FindArea {

public static void main(String args[ ]) {

Figure f = new Figure(10,10);

Rectangle r = new Rectangle(9,5);

Triangle t = new Triangle(10,8);

Figure figref;

figref = r;

System.out.println("area is "+figref.area( ));

figref = t;

System.out.println("area is "+figref.area( ));

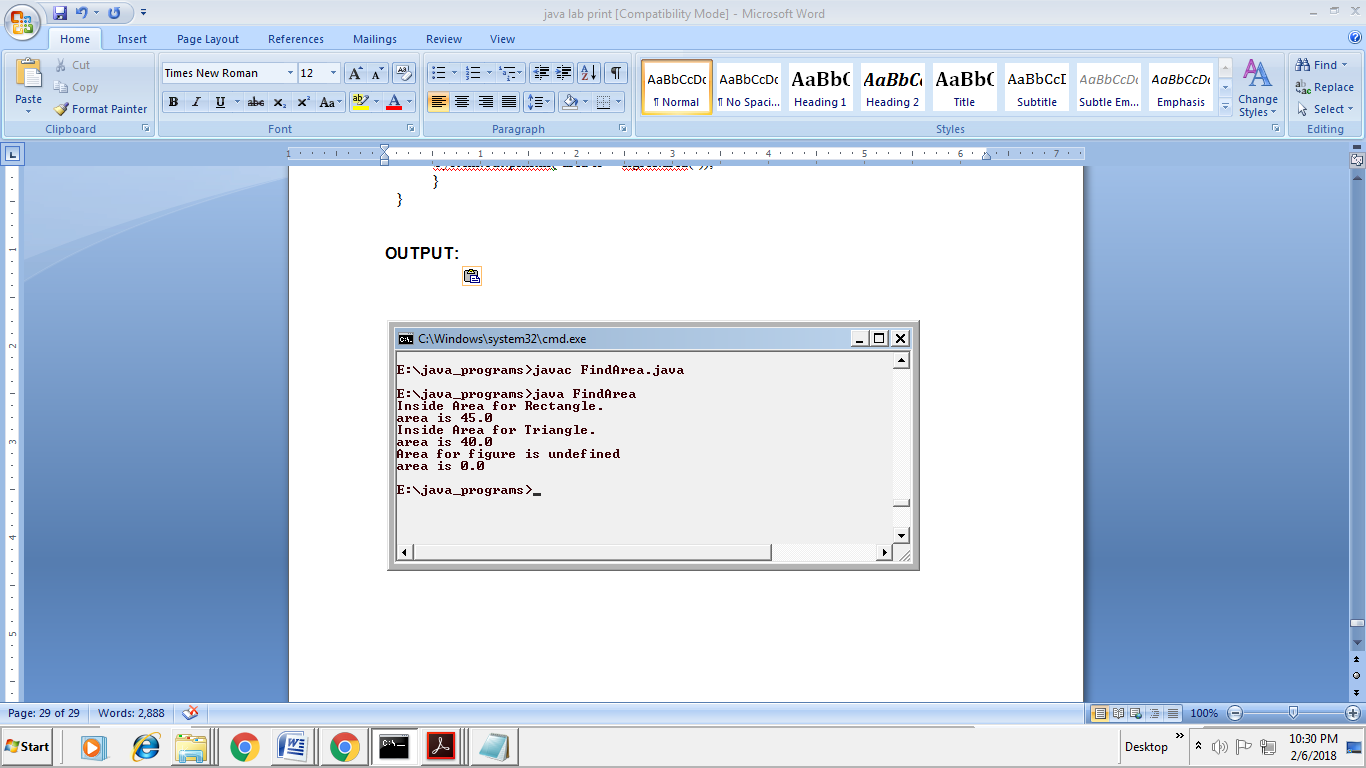
figref = f;

System.out.println("area is "+figref.area( ));

}

}

**OUTPUT:**



**8 b). Write a Case study on run time polymorphism, inheritance that implement in above problem**

Overridden methods allow java to support run-time polymorphism; polymorphism is essential to object-oriented program because it allows a general class to specify methods that will be common to all its derivatives, while allowing subclasses to define specific implementation of some or all of those methods. Overridden methods are another way that java implements the “one interface multiple methods” aspect of polymorphism.

The fundamental idea of polymorphism is that a compiler does not know which method (Super class method or subclass method) to call during the compile time. It only knows during runtime (based on object). This is called as “late binding”. This type of polymorphism is called as polymorphism by interface.

The super class provides all elements that a subclass can use directly. It also defines methods in subclass the flexibility to define its own methods, yet still enforces a consistent interfaces. This by combining inheritance with overridden methods, a super class can define the general form of the methods that will be used by all of its sub classes.

Dynamic, runtime polymorphism is one of the most powerful mechanisms that object-oriented design brings to bear on code reuse and robustness. The ability of existing code libraries to call methods on instances of new classes without recompiling while maintaining a clean abstract interface is a profoundly powerful tool.

**Exercise – 9** (User defined Exception)

**9 a). Write a JAVA program for creation of Illustrating throw**

**AIM:** JAVA program to Illustrating throw

**ALGORITHM:**

1. Create TheExample class

2. Create Scanner method

3.Then create try method taking ArithmeticExpection which predefined expection

4.Then catch method

**PROGRAM:**

import java.util.Scanner;

class ThExample {

public static void main(String args[ ]) {

System.out.println("enter age:");

Scanner sc= new Scanner(System.in);

int a = sc.nextInt( );

try {

if (a < 18)

throw new ArithmeticException("error you are not adult");

else

System.out.println("you are adult, eligible for voting");

}

catch(ArithmeticException ae) {

System.out.println(ae);

}

}

}

**OUTPUT:**



**9 b). Write a JAVA program for creation of Illustrating finally**

**AIM:** JAVA program to Illustrating finally

**ALGORITHM:**

1. Create class FinallyDemo

2. Then create static void procA() method then use try method

3.then in try method create RuntimeException then finally method

4.like wise create static void procB() method in that create try and finally methods

**PROGRAM:**

class FinallyDemo {

static void procA( ) {

try {

System.out.println("Inside procA");

throw new RuntimeException("demo");

}

finally{

System.out.println("procA's finally");

}

}

static void procB( ) {

try {

System.out.println("Inside procB");

}

finally{

System.out.println("procB's finally");

}

}

public static void main(String args[ ]) {

try {

procA( );

}

catch (Exception e) {

System.out.println("exception cought");

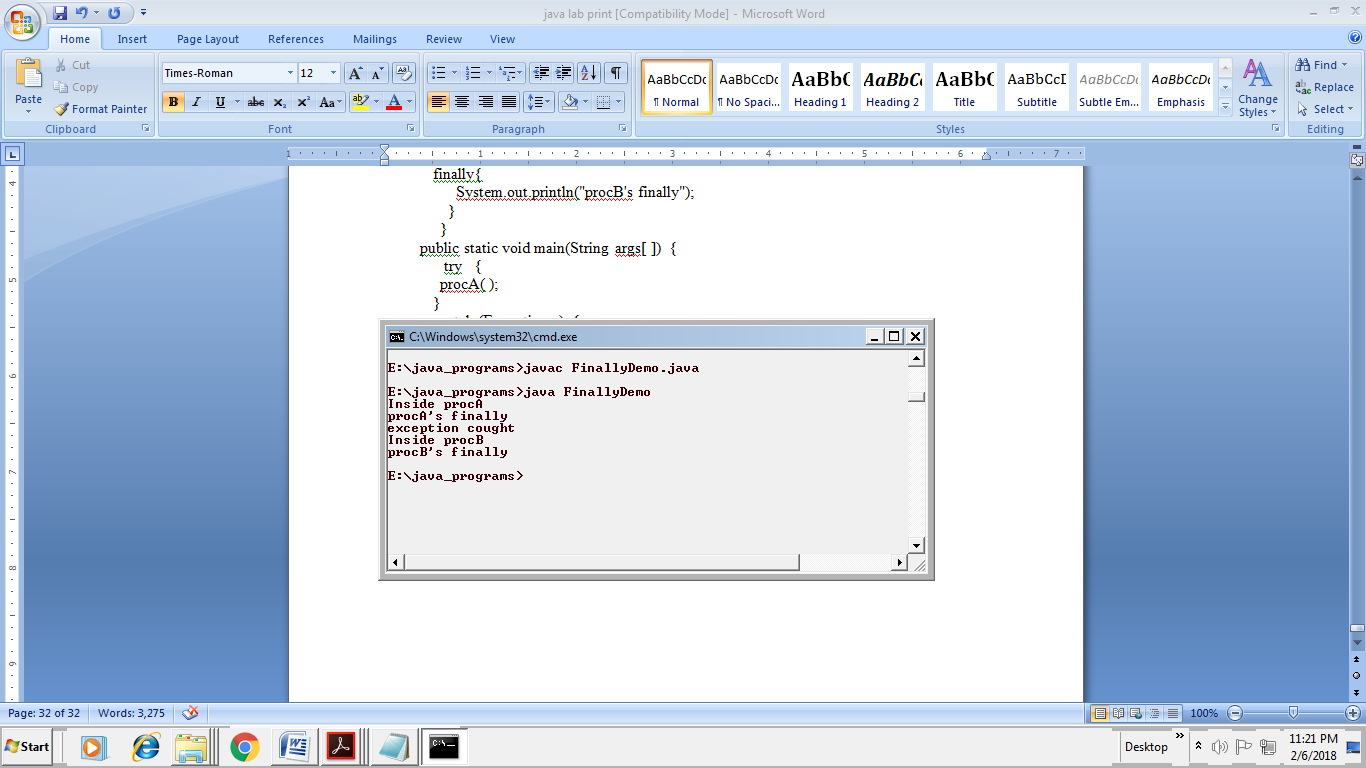
}

procB( );

}

}

**OUTPUT:**



**9 c). Write a JAVA program for creation of Java Built-in Exceptions**

**AIM:** JAVA program for Built-in Exceptions

**ALGORITHM:**

1. Import required packages.

2. Create a class called BuiltInExceptions

3. Read values called a,b

4. Check two run time errors called ArrayIndexOutOfBoundsException, ArithmeticException, FileNotFoundException, NullPointerException, NumberFormatException, StringIndexOutOfBoundsException.

5. Place statements in try block

6. Catch runtime errors using catch block and print message

**PROGRAM:**

import java.io.\*;

class BuiltInExceptions {

public static void main(String args[])

{

int x = 0;

try {

int a = 30, b = 0;

int c = a / b; // cannot divide by zero

System.out.println("Result = " + c);

}

catch (ArithmeticException e) {

System.out.println(e);

}

try{

int d[] = new int[5];

d[6] = 9;

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println(e);

}

try {

// Following file does not exist

File file = new File("E:// file.txt");

FileReader fr = new FileReader(file);

}

catch (FileNotFoundException e) {

System.out.println(e);

}

try {

String a = null; // null value

System.out.println(a.charAt(0));

}

catch (NullPointerException e) {

System.out.println(e);

}

try {

// "hello" is not a number

int num = Integer.parseInt("hello");

System.out.println(num);

}

catch (NumberFormatException e) {

System.out.println(e);

}

try {

String a = "This is like chipping "; // length is 22

char c = a.charAt(24); // accessing 25th element

System.out.println(c);

}

catch (StringIndexOutOfBoundsException e) {

System.out.println(e);

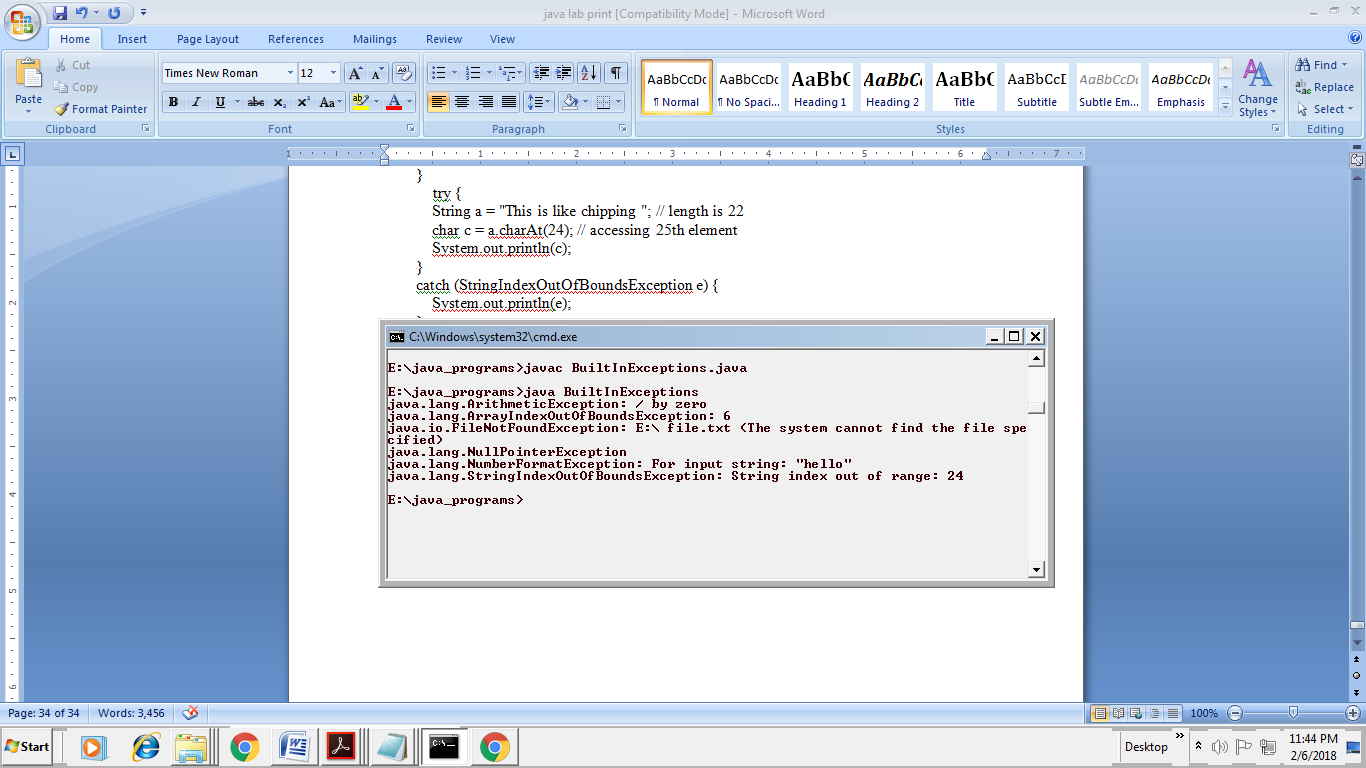
}

assert(x >= 10);

}

}

**OUTPUT:**



**9 d).Write a JAVA program for creation of User Defined Exception**

**AIM:** JAVA program for Built-in Exceptions

**DESCRIPTION:** If you are creating your own Exception that is known as custom exception or user-defined exception. Java custom exceptions are used to customize the exception according to user need. By the help of custom exception, you can have your own exception and message.

**ALGORITHM:**

1. start
2. create a class 1 that extends from Exception base class .

declare a constructor that calls super class constructor. write a method and place the statements of exception in try block. throw the exception from try block. write a catch block that prints message of exception. end class

3. Create main class give main method as public static void main(String[] args) create object to class 1and call method in previous class.

4. End main class.

5. Stop.

**PROGRAM:**

class MyException extends Exception {

String str1;

MyException(String str2) {

str1= str2;

}

public String toString( )

{

return("my exception occured: " +str1);

}

}

class Demo {

public static void main(String args[ ]) {

try {

System.out.println("Starting of try block ");

throw new MyException("this is my error message");

}

catch(MyException me) {

System.out.println("catch block");

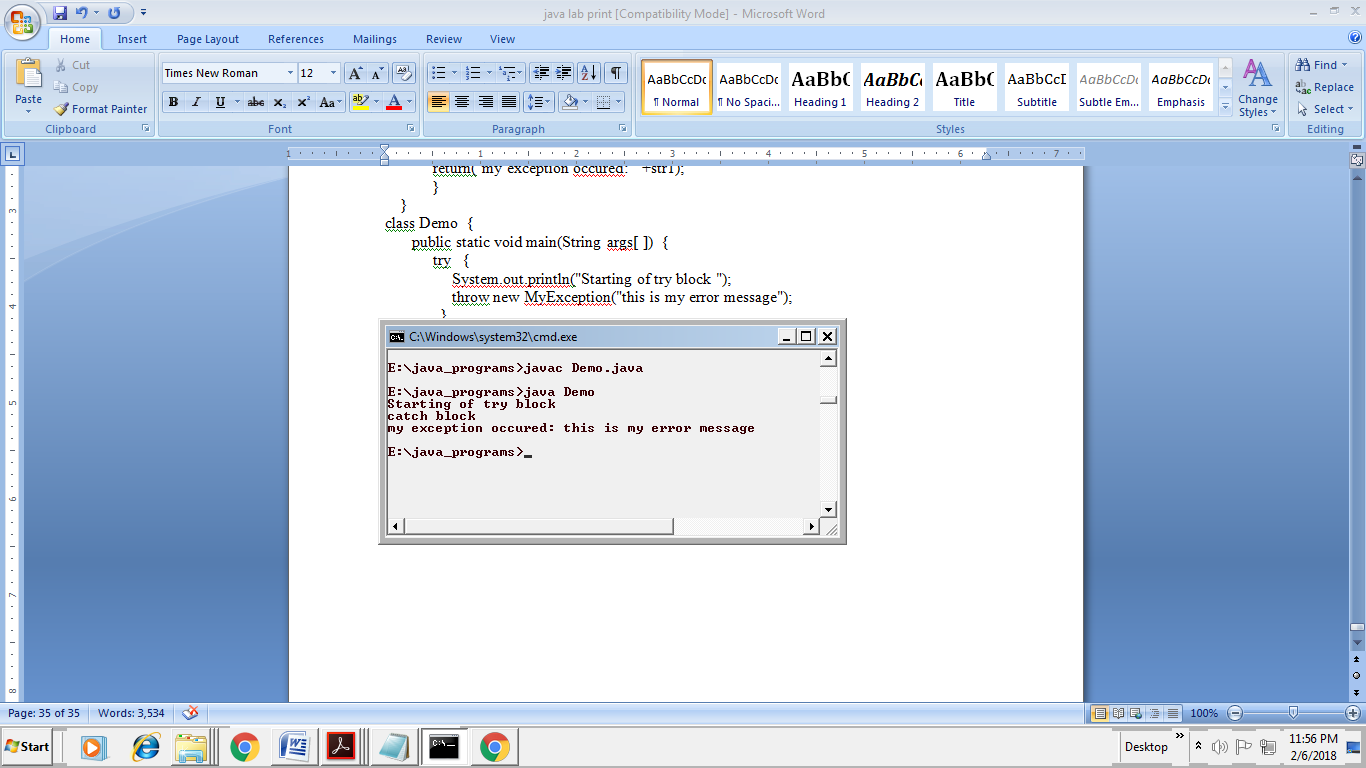
System.out.println(me);

}

}

}

**OUTPUT:**



**Exercise – 10** (Threads)

**10 a). Write a JAVA program that creates threads by extending Thread class .First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable)**

**AIM:** JAVA program for Thread creation by using Thread class and Runnable interface

**DESCRIPTION:** In this program created a thread using Runnable interface. In the Runnable interface a main thread and child threads are created, along with a run method is used. In the run method there is a thread sleep method for children and the children is suspending for 500ms and used join().

**Syntax:** Try

{

Thread.sleep(500)

} Catch(InterruptedException ie){}

**ALGORITHM:**

1. Start

2. Create a class1 that implements Runnable interface.

declare a constructor for declaring message. start the thread using start() declare run () method in which statements of exception(i.e interrupted exception) are placed in try block write for loop for some n times.

print thread name using Thread.currentThread().getMessage() give sleep(500) for suspending of thread for 500 milliseconds.

end for write catch block for handling Exception occured.

end method end class

3. Create a class main() in this create two runnable objects (i.e object to class1passing strings to constructor) then pass this runnable object as parameter to Thread class constructor creating 2 objects to Thread class. start threads 1and2 using start () end main class

4. Stop

**PROGRAM:**

class ThreadOne extends Thread {

public void run(){

try {

for(int i = 1; i<= 5; i++) {

Thread.sleep(1000);

System.out.println("sec : "+ i +" --> " + " Good Morning ");

}

} catch(InterruptedException e){

System.out.println("child thread1 interrupted");

}

}

}

class ThreadTwo extends Thread{

public void run(){

try {

for(int j = 1; j <= 5; j++){

Thread.sleep(2000);

System.out.println("sec : "+ j\*2 +" --> " + " Hello");

}

} catch(InterruptedException e){

System.out.println("child thread 2 interrupted");

}

}

}

class ThreadThree extends Thread {

public void run(){

try {

for(int k = 1; k <= 5; k++) {

Thread.sleep(3000);

System.out.println("sec : "+ k\*3 +" --> " + " Welcome");

}

} catch(InterruptedException e){

System.out.println("child thread 3 interrupted");

}

}

}

class ThreadDemoOne {

public static void main(String arg[]) {

ThreadOne a = new ThreadOne();

a.start();

ThreadTwo b = new ThreadTwo();

b.start();

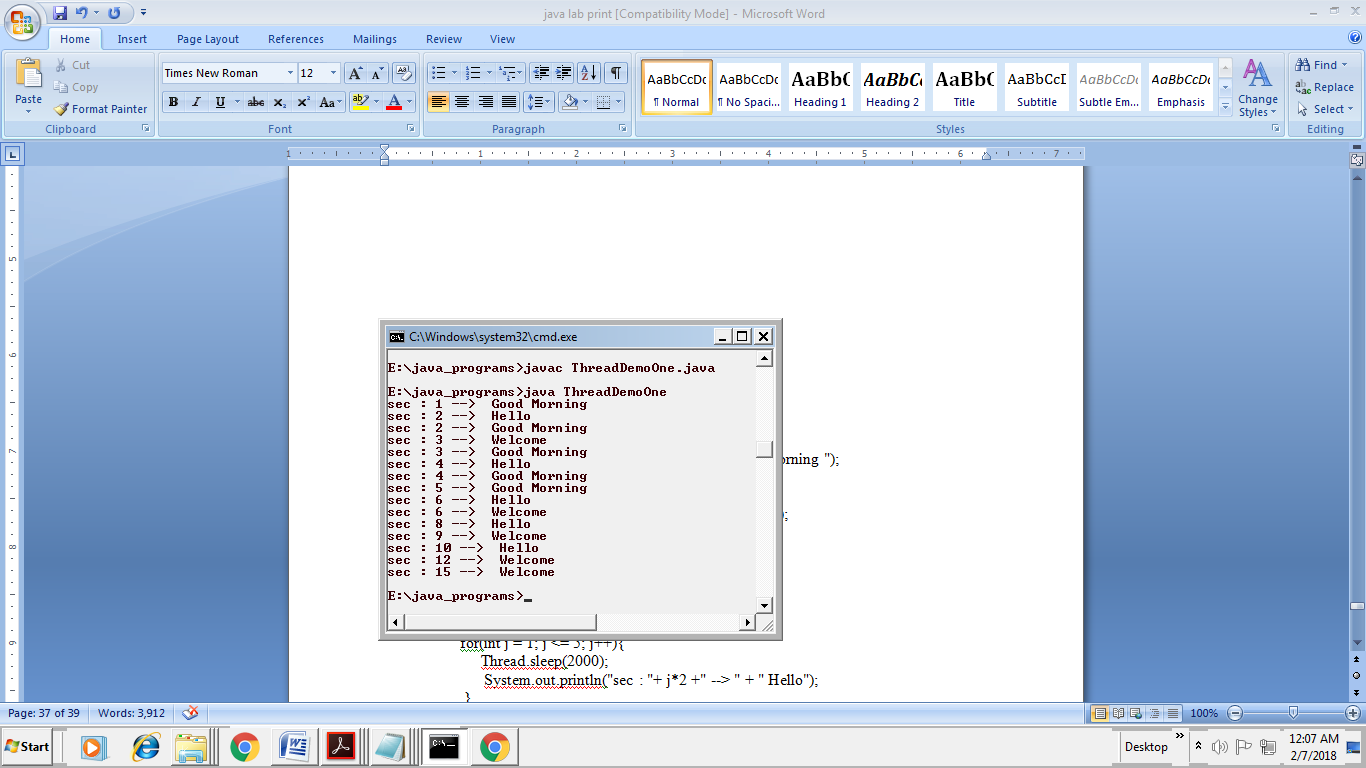
ThreadThree c = new ThreadThree();

c.start();

}

}

**OUTPUT:**

****

**PROGRAM:**

class ThreadOne implements Runnable {

public void run(){

try {

for(int i = 1; i<= 5; i++) {

Thread.sleep(1000);

System.out.println("sec : "+ i +" --> " + " Good Morning ");

}

} catch(InterruptedException e){

System.out.println("child thread1 interrupted");

}

}

}

class ThreadTwo implements Runnable{

public void run(){

try {

for(int j = 1; j <= 5; j++){

Thread.sleep(2000);

System.out.println("sec : "+ j\*2 +" --> " + " Hello");

}

} catch(InterruptedException e){

System.out.println("child thread 2 interrupted");

}

}

}

class ThreadThree implements Runnable {

public void run(){

try {

for(int k = 1; k <= 5; k++) {

Thread.sleep(3000);

System.out.println("sec : "+ k\*3 +" --> " + " Welcome");

}

} catch(InterruptedException e){

System.out.println("child thread 3 interrupted");

}

}

}

class ThreadDemo {

public static void main(String arg[]) {

ThreadOne obj1 = new ThreadOne();

Thread a=new Thread(obj1);

a.start();

ThreadTwo obj2 = new ThreadTwo();

Thread b=new Thread(obj2);

b.start();

ThreadThree obj3 = new ThreadThree();

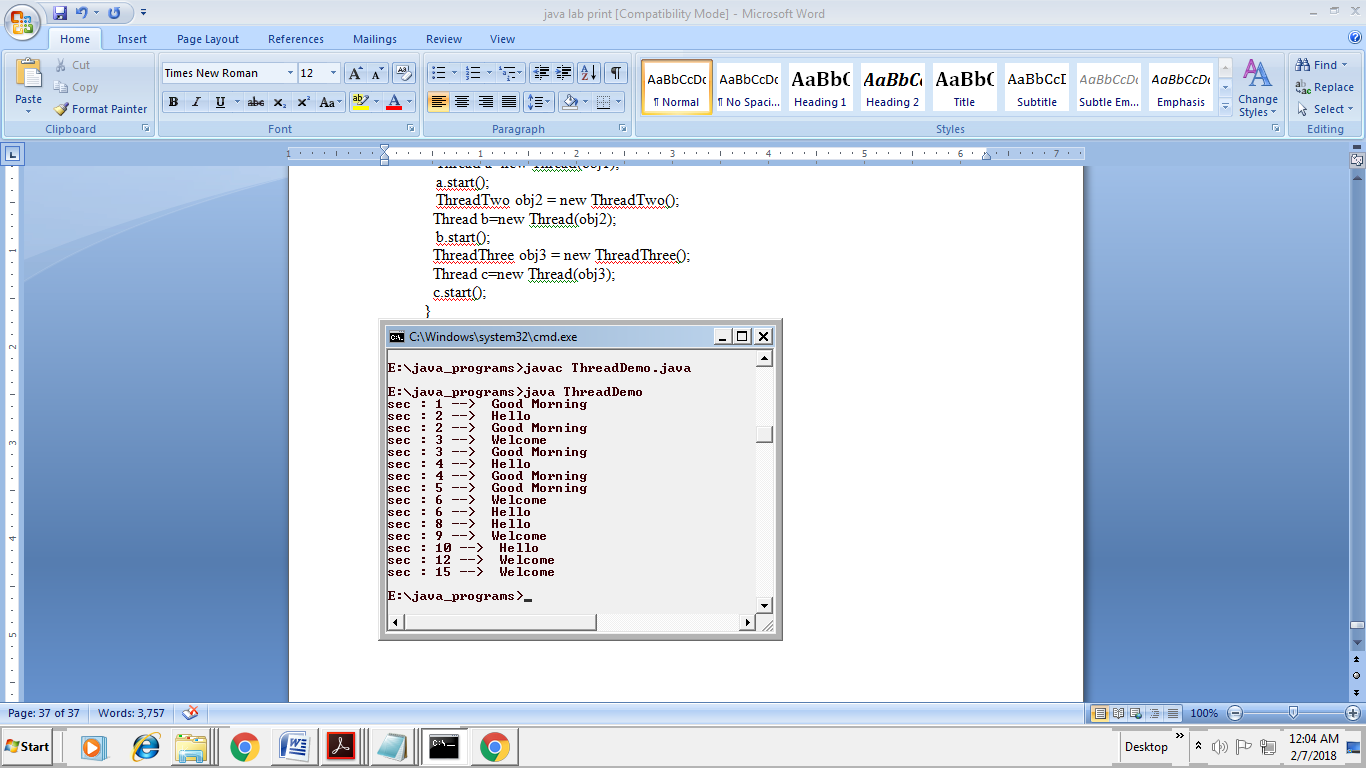
Thread c=new Thread(obj3);

c.start();

}

}

**OUTPUT:**



**10 b). Write a program illustrating isAlive and join ( )**

**AIM:** JAVA program for illustrating **isAlive** and **join ( )**

**ALGORITHM:**

1.Create a class Threadjoin

2.Implements runnable from Threadjoin class

3.Create two threads ThreadA ThreadB

4.First ThreadA starts its process,after printing the print statement that's in try block,it goes to sleep for 2000ms.

5.Next ThreadB starts it's process and prints try block print statement in the console and goes to sleep for 2000ms.

6.Now ThreadA wakes from sleep and prints catch block print statement.

7.Similarly Thread2 wakes up and prints catch block print statement.

**PROGRAM:**

class ThreadJoin implements Runnable {

String thread;

Thread thrd;

ThreadJoin (String threadName) {

thread = threadName;

thrd = new Thread (this, thread);

thrd.start();

}

public void run() {

try {

Thread.sleep(2000);

for(int i = 1; i<= 3; i++) {

System.out.println("\t From child thread " + thread + " : i = "+i);

}

} catch(InterruptedException e ) {

System.out.println("Exception: Thread "+ thread + " interrupted");

}

System.out.println("Terminating thread: " + thread );

}

}

class JoinDemo {

public static void main (String args []) {

ThreadJoin threadA = new ThreadJoin ("A");

ThreadJoin threadB = new ThreadJoin ("B");

System.out.println("Thread Status: Alive");

System.out.println("Thread A: " +threadA.thrd.isAlive());

System.out.println("Thread B: " +threadB.thrd.isAlive());

try {

System.out.println("Threads Joining......");

threadA.thrd.join();

threadB.thrd.join();

} catch (InterruptedException e){

System.out.println("Exception: Thread main interrupted.");

}

System.out.println("Thread Status: Alive");

System.out.println("Thread A: " + threadA.thrd.isAlive());

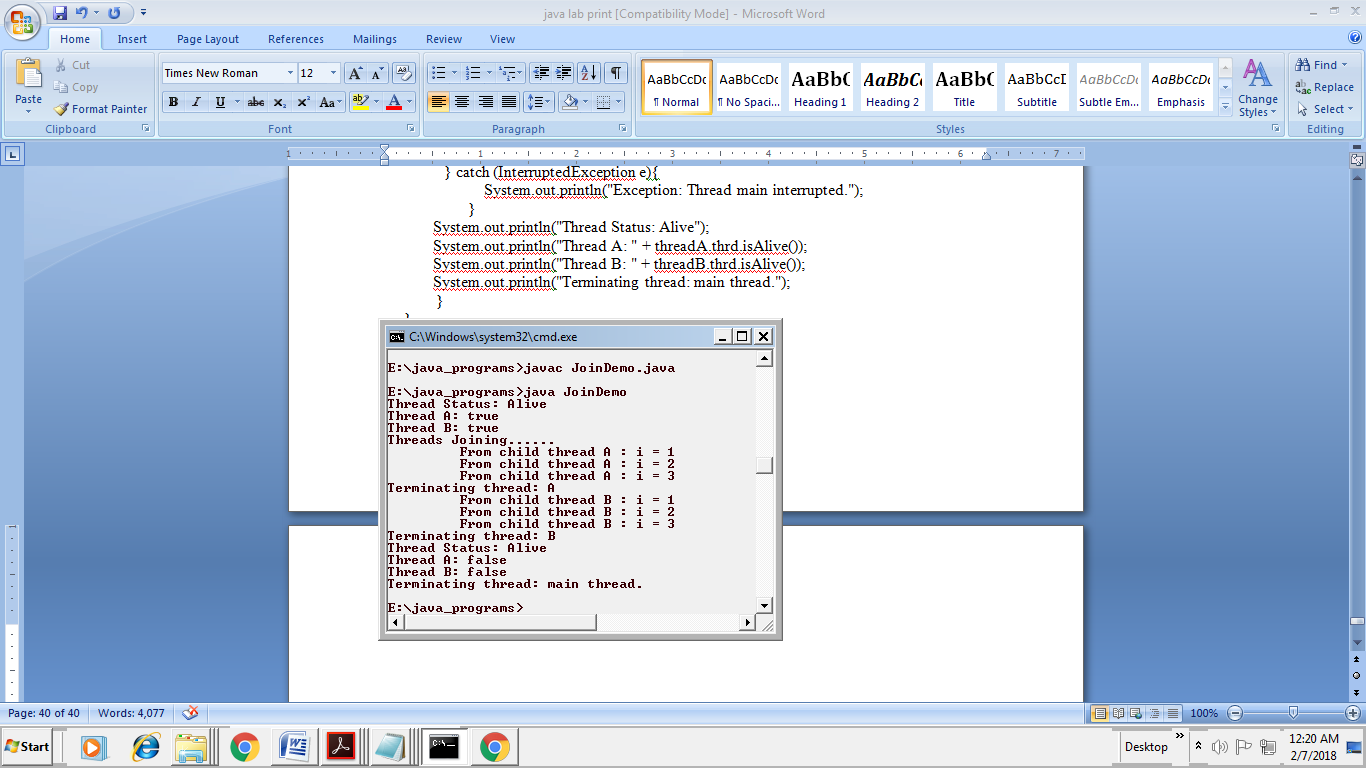
System.out.println("Thread B: " + threadB.thrd.isAlive());

System.out.println("Terminating thread: main thread.");

}

}

**OUTPUT:**



**10 c). Write a Program illustrating Daemon Threads.**

**AIM:** JAVA program for illustrating Daemon Threads.

**DESCRIPTION:** Daemon thread in java is a low-priority thread that runs in the background to perform tasks such as garbage collection. Daemon thread in java is also a service provider thread that provides services to the user thread.

**ALGORITHM:**

1. Create a class named DaemonThread

2. Extends Thread class from DaemonThread class

3. The main thread creates a daemon thread that displays a message every half second.

4. The main thread then

5. Sleeps for five seconds. While the daemon thread is still executing, the program ends because the only currently executing threads are daemon threads.

**PROGRAM:**

class DaemonThread extends Thread

{

public void run(){

if(Thread.currentThread().isDaemon()){

System.out.println("daemon thread work");

}

else{

System.out.println("user thread work");

}

}

public static void main(String[] args){

DaemonThread t1=new DaemonThread();

DaemonThread t2=new DaemonThread();

DaemonThread t3=new DaemonThread();

t1.setDaemon(true);

t1.start();//starting threads

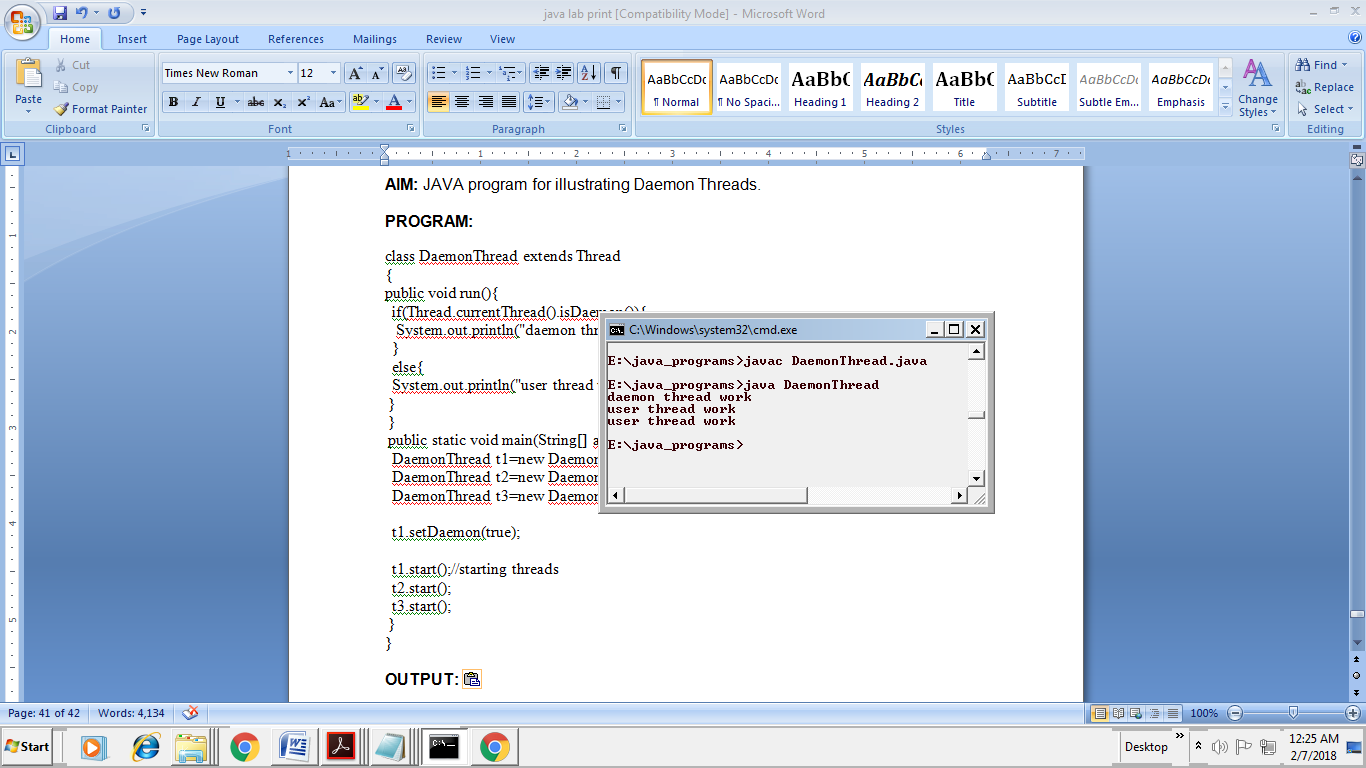
t2.start();

t3.start();

}

}

**OUTPUT:**



**Exercise - 11** (Threads continuity)

**11 a).Write a JAVA program Producer Consumer Problem**

**AIM:** JAVA program for Producer Consumer Problem

**ALGORITHM:**

1: create a class Queue with get and put synchronised methods

2: create two classes producer and consumer implementing runnable interface and the default run method

3: create a class PCFixed which has main method.An instance to the class queue to perform put get functionalities.

**PROGRAM:**

class Queue {

int n;

boolean valueSet = false;

synchronized int get() {

while(!valueSet)

try {

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException caught");

}

System.out.println("Got: " + n);

valueSet = false;

notify();

return n;

}

synchronized void put(int n) {

while(valueSet)

try {

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException caught");

}

this.n = n;

valueSet = true;

System.out.println("Put: " + n);

notify();

}

}

class Producer implements Runnable {

Queue q;

Producer(Queue q) {

this.q = q;

new Thread(this, "Producer").start();

}

public void run() {

int i = 0;

while(true) {

q.put(i++);

}

}

}

class Consumer implements Runnable {

Queue q;

Consumer(Queue q) {

this.q = q;

new Thread(this, "Consumer").start();

}

public void run() {

while(true) {

q.get();

}

}

}

class PCFixed {

public static void main(String args[]) {

Queue q = new Queue();

new Producer(q);

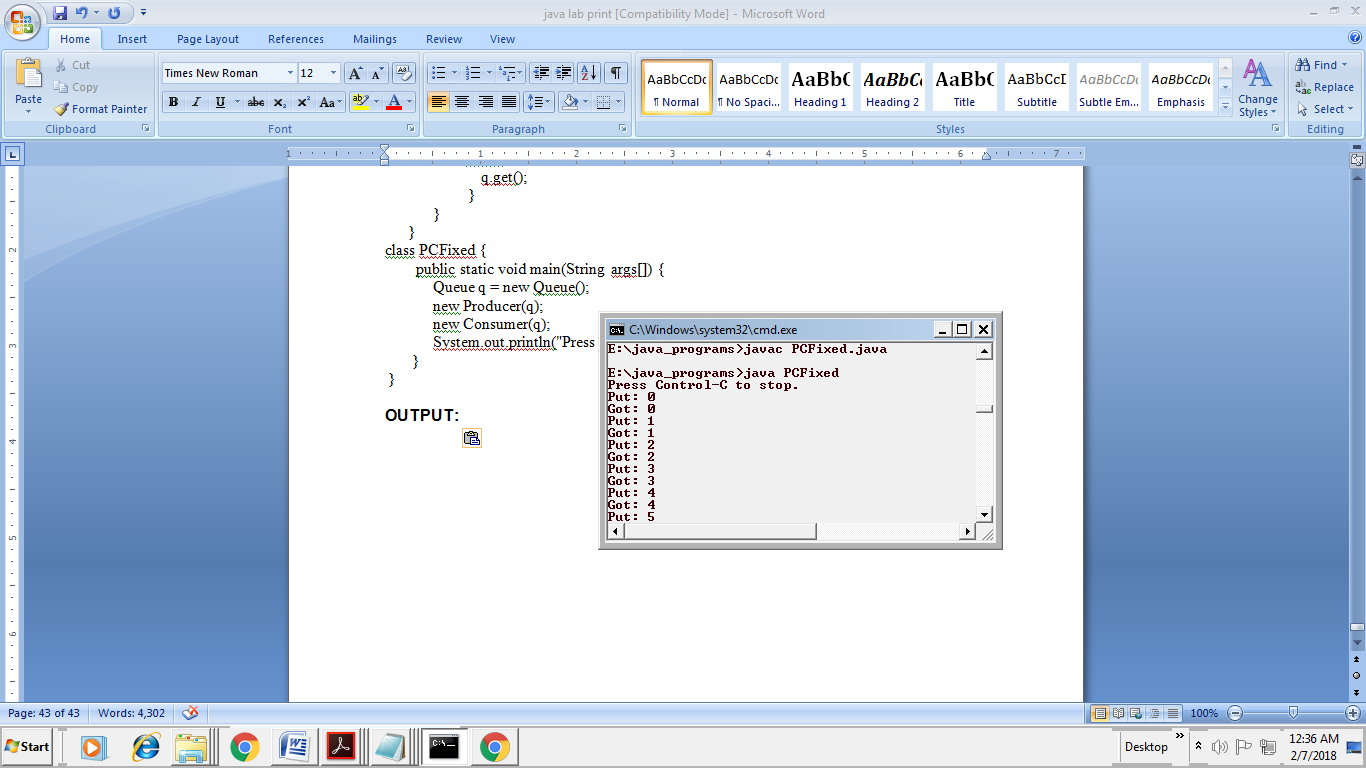
new Consumer(q);

System.out.println("Press Control-C to stop.");

}

}

**OUTPUT:**



**11 b).Write a case study on thread Synchronization after solving the above producer consumer problem**

The fact that a thread can execute in parallel with other threads is the main power of the concurrent programming. But this property causes usually a lot of trouble for programmers which must assure a correct sharing of the resources used in common by different threads. It means that the threads that execute in parallel and use common critical resources must be synchronized at some points to assure a correct functioning of the system. The correct functioning of a parallel or concurrent program is possible only if the conflicts of simultaneously performing a critical section operation by several threads are avoided. A critical section operation is an operation (a portion of the program code) that cannot be done in parallel by several processes or treads, like incrementing a sheared counter, writing in a shared buffer, modifying a shared object, etc.

The synchronization means that a thread must wait for another thread to leave the critical section and to enter into this section using some security measures, e.g. locking the critical section when entering it. To show a situation when the synchronization is strictly necessary considers the following program. The program creates two threads that use a shared object to count the total number of iterations done by both threads.

You can use wait, notify and notifyAll methods to communicate between threads in Java. For example, if you have two threads running in your program e.g.Producer and Consumer then producer thread can communicate to the consumer that it can start consuming now because there are items to consume in the queue. Similarly, a consumer thread can tell the producer that it can also start putting items now because there is some space in the queue, which is created as a result of consumption. A thread can use wait() method to pause and do nothing depending upon some condition. For example, in the [producer-consumer problem](http://java67.blogspot.sg/2012/12/producer-consumer-problem-with-wait-and-notify-example.html), producer thread should wait if the queue is full and consumer thread should wait if the queue is empty.  
  
 If some thread is waiting for some condition to become true, you can use notify and notifyAll methods to inform them that condition is now changed and they can wake up.Both notify() and notifyAll() method sends a notification but notify sends the notification to only one of the waiting thread, no guarantee which thread will receive notification and notifyAll() sends the notification to all threads.So if only one thread is waiting for an object lock, also known as a monitor then both notify and notifyAll will send the notification to it. If [multiple threads](http://javarevisited.blogspot.com/2013/02/how-to-join-multiple-threads-in-java-example-tutorial.html) are waiting on a monitor then notify will only inform one of the lucky thread and rest will not receive any notification, but notifyAll will inform all threads.

**Exercise – 12** (Packages)

**12 a). Write a JAVA program illustrate class path**

**AIM:** JAVA program to illustrate class path

**ALGORITHM:**

1. Import packages

2. Create Factorial class

3. Write Fact function to calculate factorial.

**PROGRAM:**

package mypack;

public class Factorial{

public int fact(int a){

if(a == 1)

return 1;

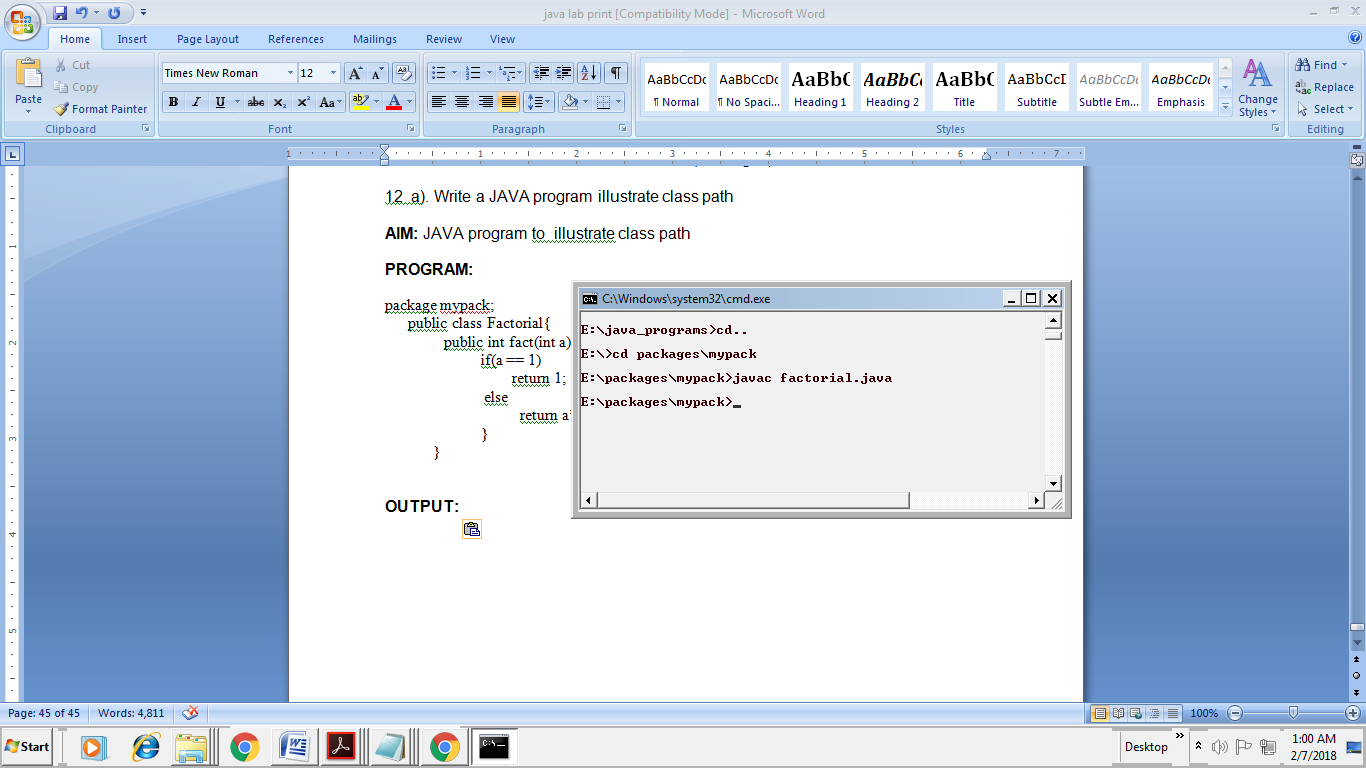
else

return a\*fact(a-1);

}

}

**OUTPUT: compile the program of the specific class path**



This program will not run it does not contain main method.

**12 b). Write a case study on including in class path in your os environment of your package.**

Classpath is used for storing the path of the third-party and user-defined classes. Whenever we execute/compile any class file, jdk tools javac and java, search the package/class file in the user classpath which is the current directory by default. If the classes are not in the current directory,then we need to set the classpath.

The classpath can be set in two ways:

1. It is an environment variable which can be set using the *System* utility in the control

panel or at the DOS prompt as shown.

Set CLASSPATH = %CLASSPATH%;c:\pack;

%Classpath% is used to keep the existing path intact and append our new path to it. Now L3 of both the above cases will execute.

2. Use classpath option –classpath or –cp of javac/java tools to override the user-defined

classpath and find the user-defined specific package/classes used in the Java source

files.

//syntax: javac –cp path of the directory/package used in java source file

followed by name of the java source file

C:\pack\packexample> javac –cp c:\javaeg DemoClass.java

-cp specifies that the user-defined package/classes used in DemoClass.java will be found

at c:\javaeg.

**12 c). Write a JAVA program that import and use the defined your package in the previous Problem**

**AIM:** JAVA program to implement user defined package

**ALGORITHM:**

1. Import packages
2. Create class FactorialOne
3. Declare main method and declare objects for class to access fact.

**PROGRAM:**

import mypack.\*;

import java.util.Scanner;

class FactorialOne{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.print("enter an integer:");

int a = sc.nextInt();

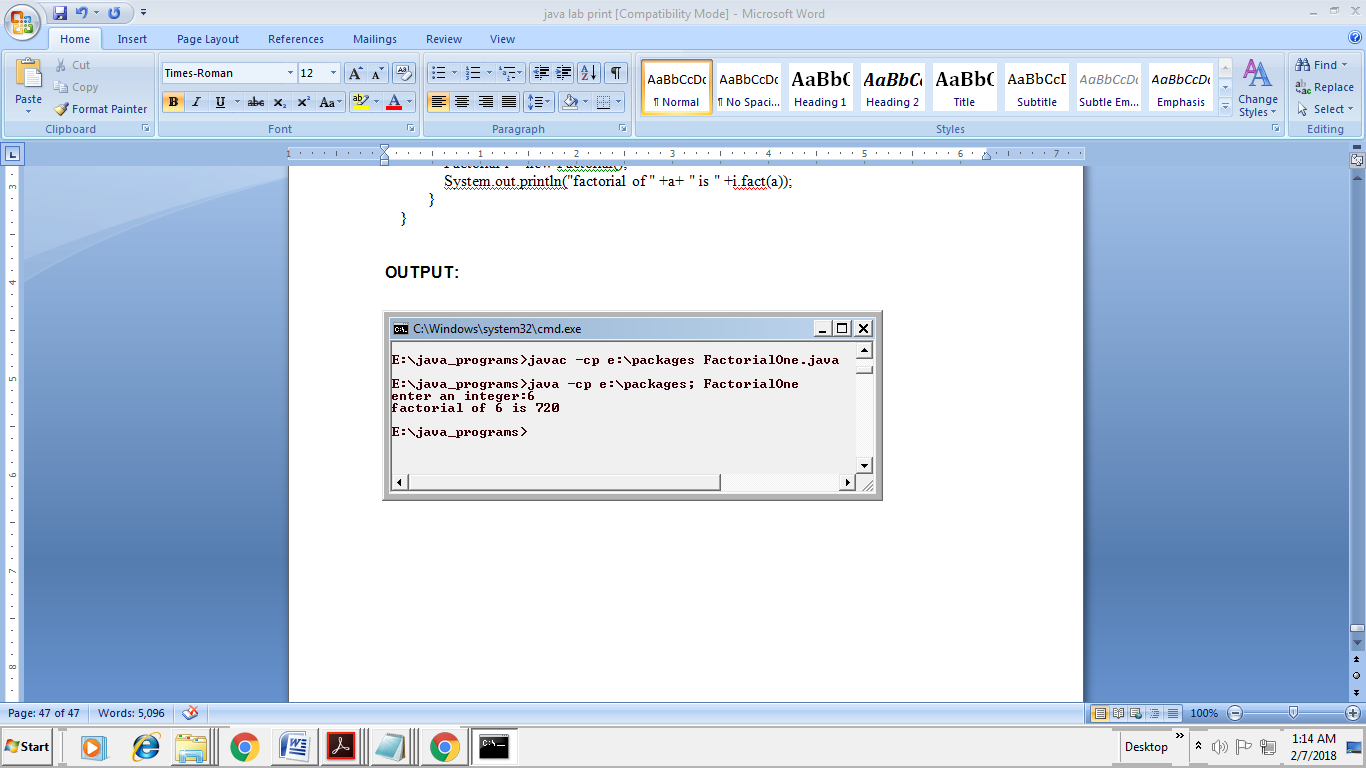
Factorial i = new Factorial();

System.out.println("factorial of " +a+ " is " +i.fact(a));

}

}

**OUTPUT:**

****

**Exercise - 13** (Applet)

13 a).Write a JAVA program to paint like paint brush in applet.

**AIM:** JAVA program to paint like paint brush in applet

**ALGORITHM:**

1. Create the class ‘MouseDrag’ which inherits the Applet class and implements ‘MouseMotionListener’.
2. Create the Constructor to ‘MouseDrag’ class and invoke the methods ‘addMouseMotionListener()’ and ‘setBackground()’.
3. Create the function ‘mouseDragged()’
4. Inside this function get the instance for Graphics function.
5. Make use of this instance to set the colors and positions of the cursor.
6. Also define the ‘mouseMoved()’ function as it is a part of the ‘MouseMotionListener’ interface.
7. End of the class.

**PROGRAM:**

/\*<applet code="MouseDrag.class" width="300" height="300"> </applet>\*/

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

public class MouseDrag extends Applet implements MouseMotionListener{

public void init(){

addMouseMotionListener(this);

setBackground(Color.white);

}

public void mouseDragged(MouseEvent me){

Graphics g=getGraphics();

g.setColor(Color.green);

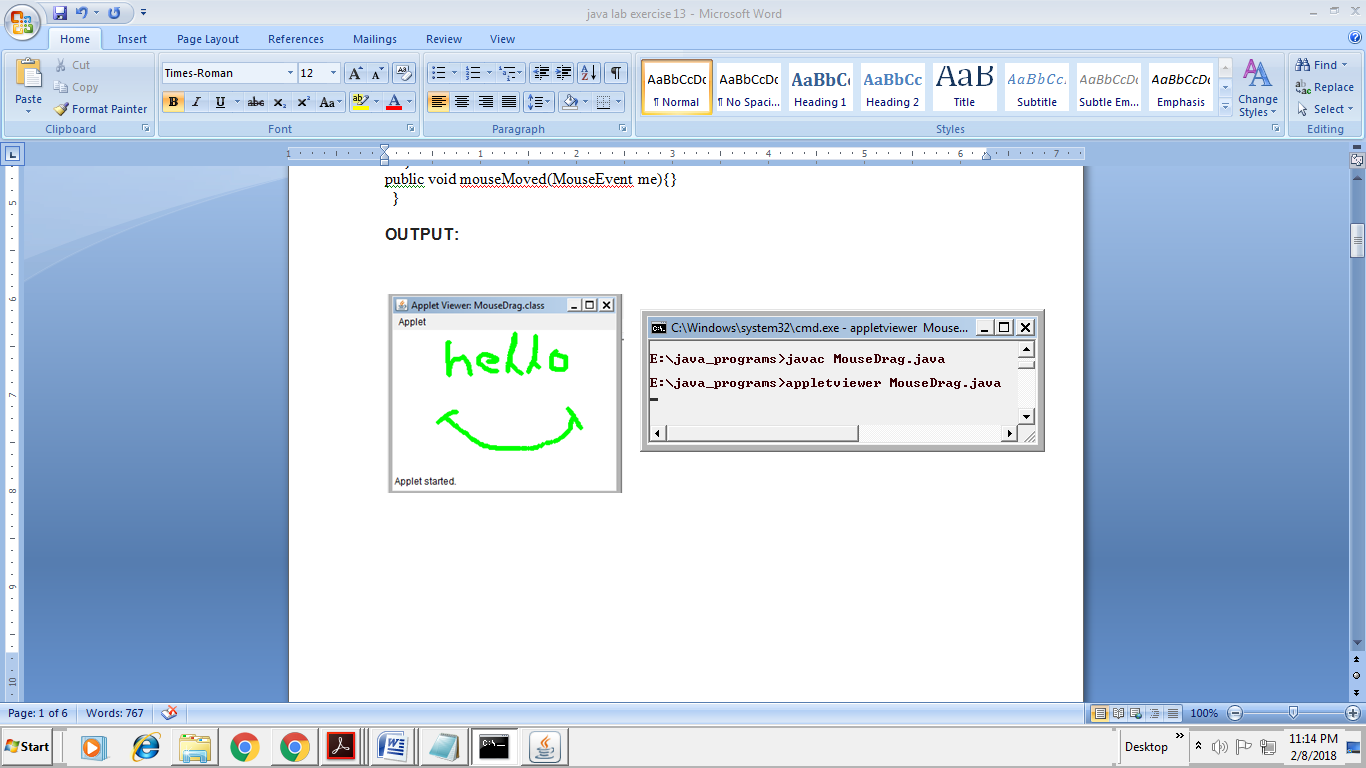
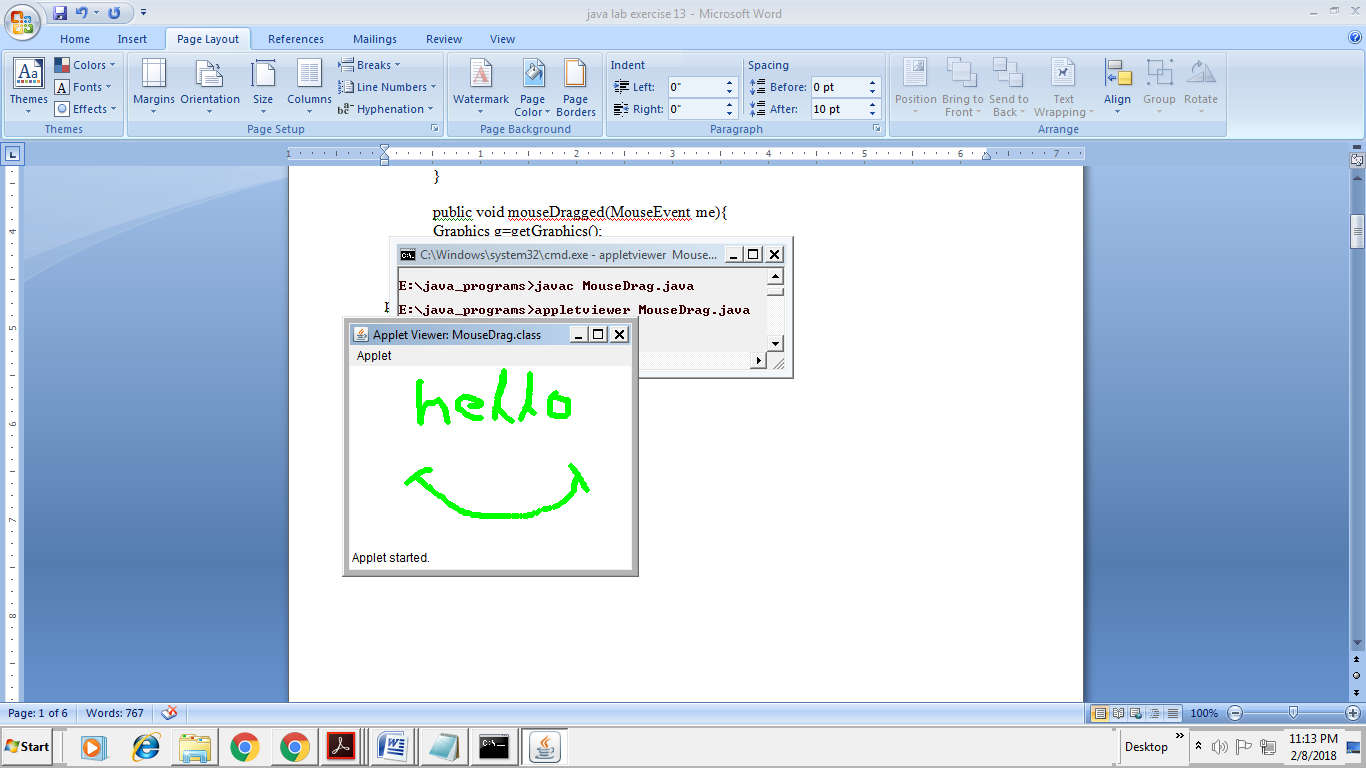
g.fillOval(me.getX(),me.getY(),6,6);

}

public void mouseMoved(MouseEvent me){}

}

**OUTPUT:**

****

13 b) Write a JAVA program to display analog clock using Applet.

**AIM:** JAVA program to display analog clock using Applet.

**ALGORITHM:**

1. Create a class ‘MyClock’ which inherits the Applet class and implements the ‘Runnable’ interface.
2. Create a constructor to this class to initialize the parameters like ‘width’, ‘height’ and ‘setBackground()’ function.
3. Implement the function called ‘start’ to create the ‘Thread’ instances and to make necessary function calls.
4. Implement the ‘stop()’ function to manage the Boolean properties of the ‘threadSuspended’ attribute.
5. Implement the ‘run()’ function, Inside that create the instance of ‘Calendar’ class.
6. With instance created, call the ‘get()’ function to access the minutes and seconds.
7. Initialize the ‘SimpleDateFormat’ and format the time.
8. Set the sleep time.
9. Handle the Exceptions (if any).
10. Create the function ‘drawHand()’ and draw the clock requirements with the help of ‘Graphics’ class.
11. Create the function ‘drawWedge’ which is used to manage the handle moments in the clock according to the time.
12. Finally implement the ‘paint()’ function to draw the clock equipment design earlier.
13. End of the class.

**PROGRAM:**

/\*<applet code="MyClock.class" width="300" height="300"> </applet>\*/

import java.applet.\*;

import java.awt.\*;

import java.util.\*;

import java.text.\*;

public class MyClock extends Applet implements Runnable {

int width, height;

Thread t = null;

boolean threadSuspended;

int hours=0, minutes=0, seconds=0;

String timeString = "";

public void init() {

width = getSize().width;

height = getSize().height;

setBackground( Color.black );

}

public void start() {

if ( t == null ) {

t = new Thread( this );

t.setPriority( Thread.MIN\_PRIORITY );

threadSuspended = false;

t.start();

}

else {

if ( threadSuspended ) {

threadSuspended = false;

synchronized( this ) {

notify();

}

}

}

}

public void stop() {

threadSuspended = true;

}

public void run() {

try {

while (true) {

Calendar cal = Calendar.getInstance();

hours = cal.get( Calendar.HOUR\_OF\_DAY );

if ( hours > 12 ) hours -= 12;

minutes = cal.get( Calendar.MINUTE );

seconds = cal.get( Calendar.SECOND );

SimpleDateFormat formatter

= new SimpleDateFormat( "hh:mm:ss", Locale.getDefault() );

Date date = cal.getTime();

timeString = formatter.format( date );

// Now the thread checks to see if it should suspend itself

if ( threadSuspended ) {

synchronized( this ) {

while ( threadSuspended ) {

wait();

}

}

}

repaint();

t.sleep( 1000 ); // interval specified in milliseconds

}

}

catch (Exception e) { }

}

void drawHand( double angle, int radius, Graphics g ) {

angle -= 0.5 \* Math.PI;

int x = (int)( radius\*Math.cos(angle) );

int y = (int)( radius\*Math.sin(angle) );

g.drawLine( width/2, height/2, width/2 + x, height/2 + y );

}

void drawWedge( double angle, int radius, Graphics g ) {

angle -= 0.5 \* Math.PI;

int x = (int)( radius\*Math.cos(angle) );

int y = (int)( radius\*Math.sin(angle) );

angle += 2\*Math.PI/3;

int x2 = (int)( 5\*Math.cos(angle) );

int y2 = (int)( 5\*Math.sin(angle) );

angle += 2\*Math.PI/3;

int x3 = (int)( 5\*Math.cos(angle) );

int y3 = (int)( 5\*Math.sin(angle) );

g.drawLine( width/2+x2, height/2+y2, width/2 + x, height/2 + y );

g.drawLine( width/2+x3, height/2+y3, width/2 + x, height/2 + y );

g.drawLine( width/2+x2, height/2+y2, width/2 + x3, height/2 + y3 );

}

public void paint( Graphics g ) {

g.setColor( Color.gray );

drawWedge( 2\*Math.PI \* hours / 12, width/5, g );

drawWedge( 2\*Math.PI \* minutes / 60, width/3, g );

drawHand( 2\*Math.PI \* seconds / 60, width/2, g );

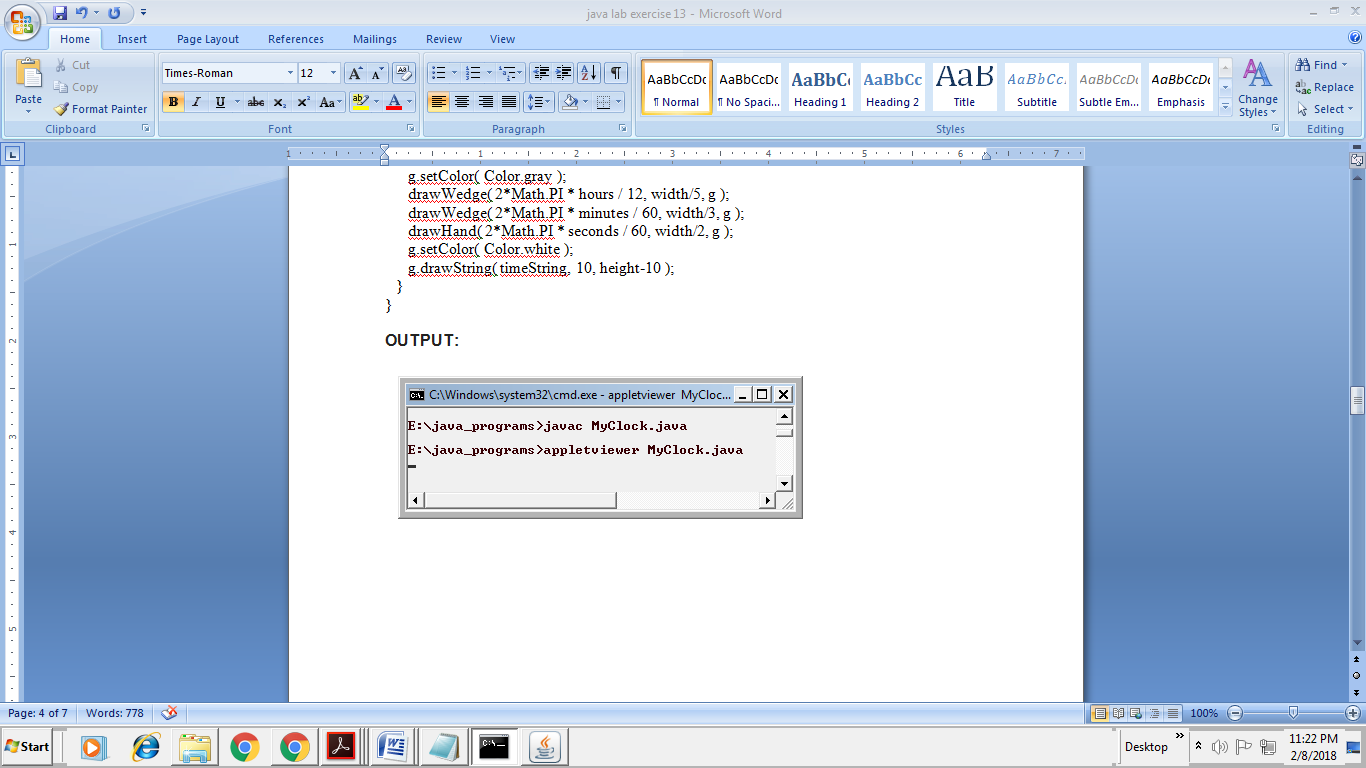
g.setColor( Color.white );

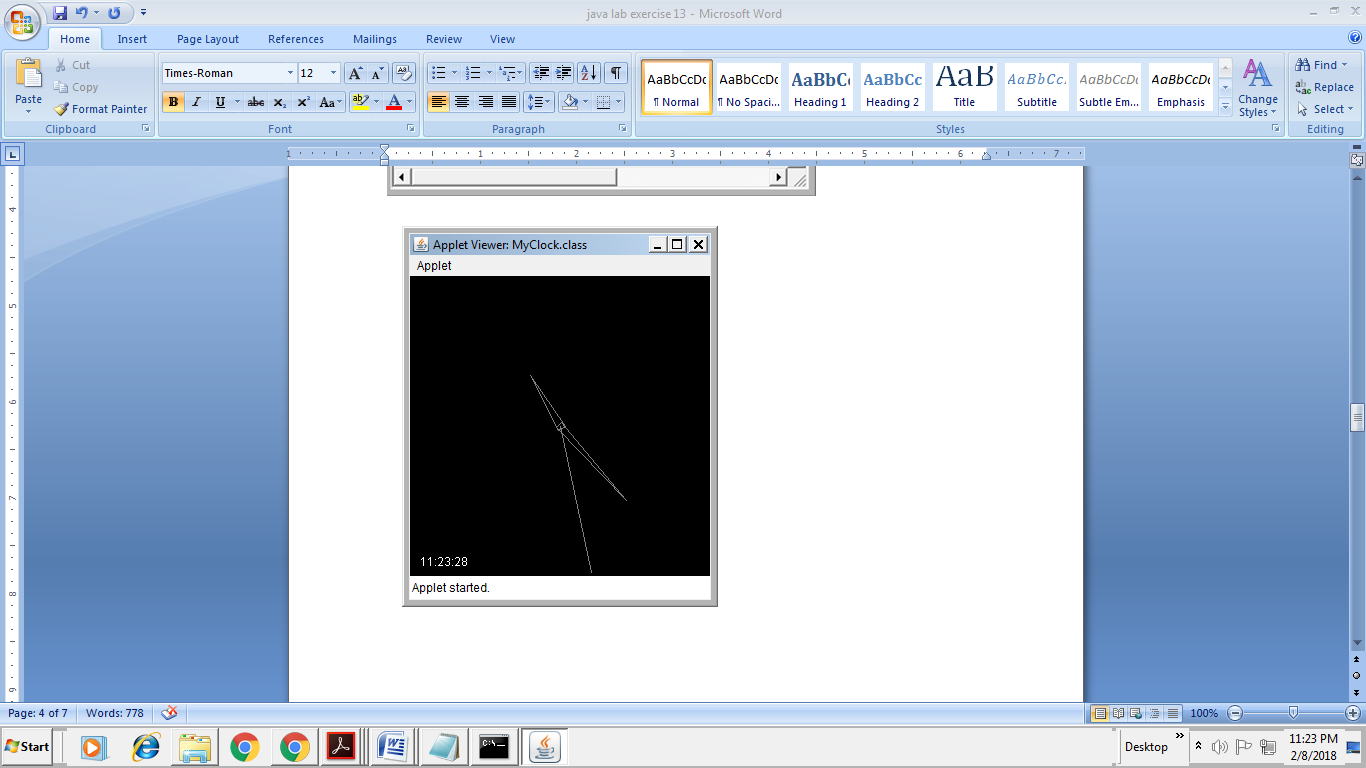
g.drawString( timeString, 10, height-10 );

}

}

**OUTPUT:**

****

****

13 c). Write a JAVA program to create different shapes and fill colors using Applet.

**AIM:** JAVA program to create different shapes and fill colors using Applet.

**ALGORITHM:**

**1.** Create the class named ‘ShapColor’ which inherits the Applet class.

2. Initialize the the variable values of ‘x’, ‘y’ and ‘r’.

3.Create a function named ‘pait()’ with thr Graphics instance as a parameters.

4.Use this instance ‘g’ to call the drawing function like ‘setColor()’, ‘drawLine()’, ‘fillOval()’ and many other by passing the necessary requirements.

**PROGRAM:**

/\*<applet ALIGN = "CENTER" CODE = "ShapColor.class" WIDTH = 800 HEIGHT = 200> </applet>\*/

import java.applet.\*;

import java.awt.\*;

public class ShapColor extends Applet{

int x=300,y=100,r=50;

public void paint(Graphics g){

g.setColor(Color.red); //Drawing line color is red

g.drawLine(3,100,100,10);

g.setColor(Color.magenta);

g.drawString("Line",100,100);

g.drawOval(x-r,y-r,100,100);

g.setColor(Color.yellow); //Fill the yellow color in circle

g.fillOval( x-r,y-r, 100, 100 );

g.setColor(Color.magenta);

g.drawString("Circle",275,100);

g.drawRect(400,50,200,100);

g.setColor(Color.yellow); //Fill the yellow color in rectangel

g.fillRect( 400, 50, 200, 100 );

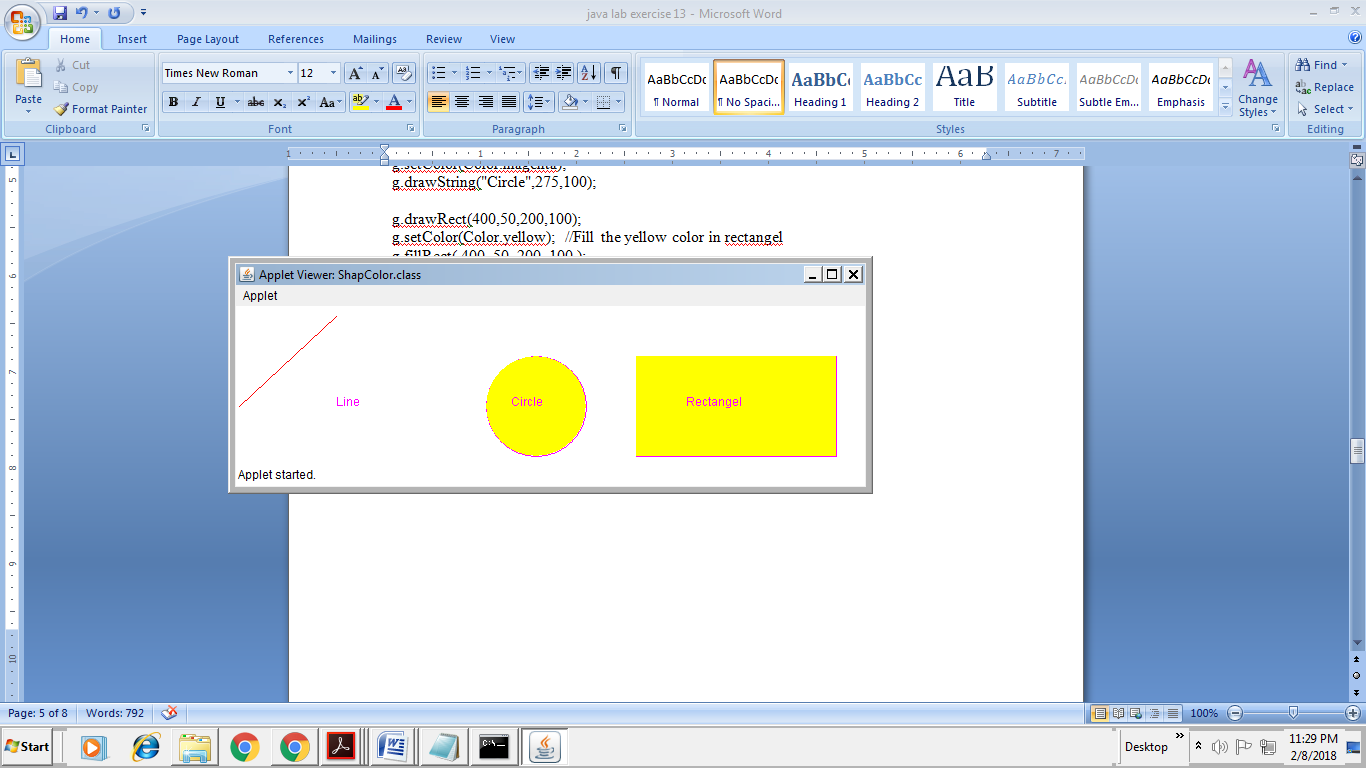
g.setColor(Color.magenta);

g.drawString("Rectangel",450,100);

}

}

**OUTPUT:**



**Exercise - 14** (Event Handling)

14 a).Write a JAVA program that display the x and y position of the cursor movement using Mouse.

**AIM:** JAVA program that display the x and y position of the cursor movement using Mouse.

**ALGORITHM:**

1. Create the class called ‘Mouseevent’ which extends the ‘Applet’ class
2. Initialize the variables ‘x’, ‘y’ to 0.
3. Create the constructor which invokes the ’addMouseListener’ and instantiate it.
4. Define the start() function as it is a part of the interface.
5. Create the paint() function with graphics instance as the default parameter.
6. Create another class named ‘mymouselistener’ which extends ‘MouseAdapter’ inside this function and create the function called ‘mouseClicked()’ and assign the x , y values i.e., the positions
7. End of the mymouselistener class
8. End of the Mouseevent class

**PROGRAM:**

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

/\* <applet code = "Mouseevent.class" width = 400 height = 200> </applet> \*/

public class Mouseevent extends Applet

{

int x=0;

int y=0;

public void init()

{

addMouseListener(new mymouselistener());

}

public void start()

{

}

public void paint(Graphics g)

{

g.drawLine(x,y,x,y);

g.drawString(x + ","+ y, x,y);

}

public class mymouselistener extends MouseAdapter

{

public void mouseClicked(MouseEvent e)

{

x = e.getX();

y = e.getY();

repaint();

} } }

**OUTPUT:**

****

14 b).Write a JAVA program that identifies key-up key-down event user entering text in a Applet.

**AIM:** JAVA program that identifies key-up key-down event user entering text in a Applet.

**ALGORITHM:**

1. Create the class called ‘Key’ which extends the ‘Applet’ and the ‘KeyListener’ interface.
2. Initialize the variables x, y and the string msg.
3. Create the constructor to instialize the addKeyListener(), requestFocus(), setBackground() and setForeground() functions.
4. Create the keyPressed() function which implements the switch case to see the multiple cases.
5. Create the function ‘KeyReleased()’ which is used to check the status using the ‘showStatus()’ function.
6. Create the function ‘keyTuned()’ which accepts the characters and attaches to the ‘msg’ variable.
7. Create the function ‘paint()’ which includes the ’Graphics’ instance as default attribute.
8. Use the default instance to call the required functions to draw.
9. End the Class.

**PROGRAM:**

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

/\*

<applet code="Key.class" width=800 height=200>

</applet>

\*/

public class Key extends Applet

implements KeyListener

{

int X=20,Y=30;

String msg="Type here: ";

public void init()

{

addKeyListener(this);

requestFocus();

setBackground(Color.green);

setForeground(Color.black);

}

public void keyPressed(KeyEvent k)

{

showStatus("KeyDown");

int key=k.getKeyCode();

switch(key)

{

case KeyEvent.VK\_UP:

showStatus("Move to Up");

break;

case KeyEvent.VK\_DOWN:

showStatus("Move to Down");

break;

case KeyEvent.VK\_LEFT:

showStatus("Move to Left");

break;

case KeyEvent.VK\_RIGHT:

showStatus("Move to Right");

break;

}

repaint();

}

public void keyReleased(KeyEvent k)

{

showStatus("Key Up");

}

public void keyTyped(KeyEvent k)

{

msg+=k.getKeyChar();

repaint();

}

public void paint(Graphics g)

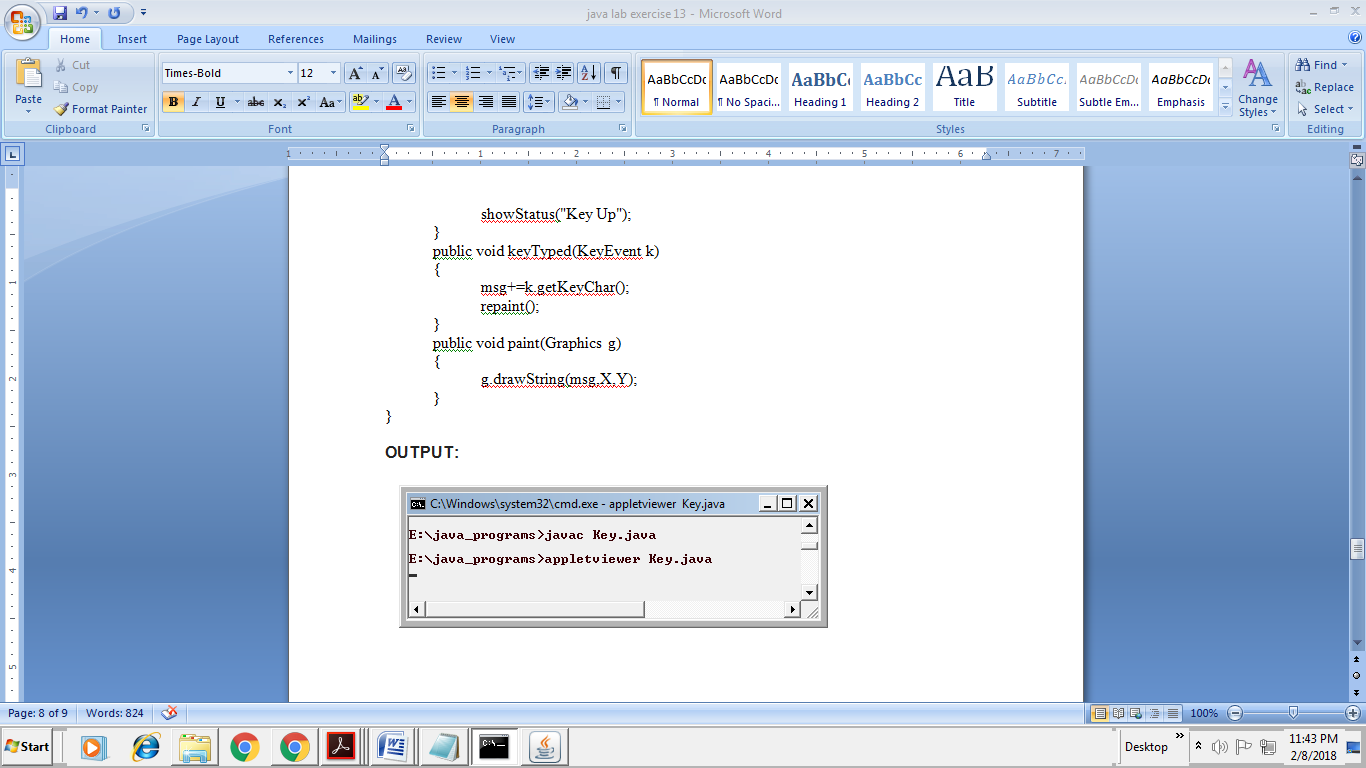
{

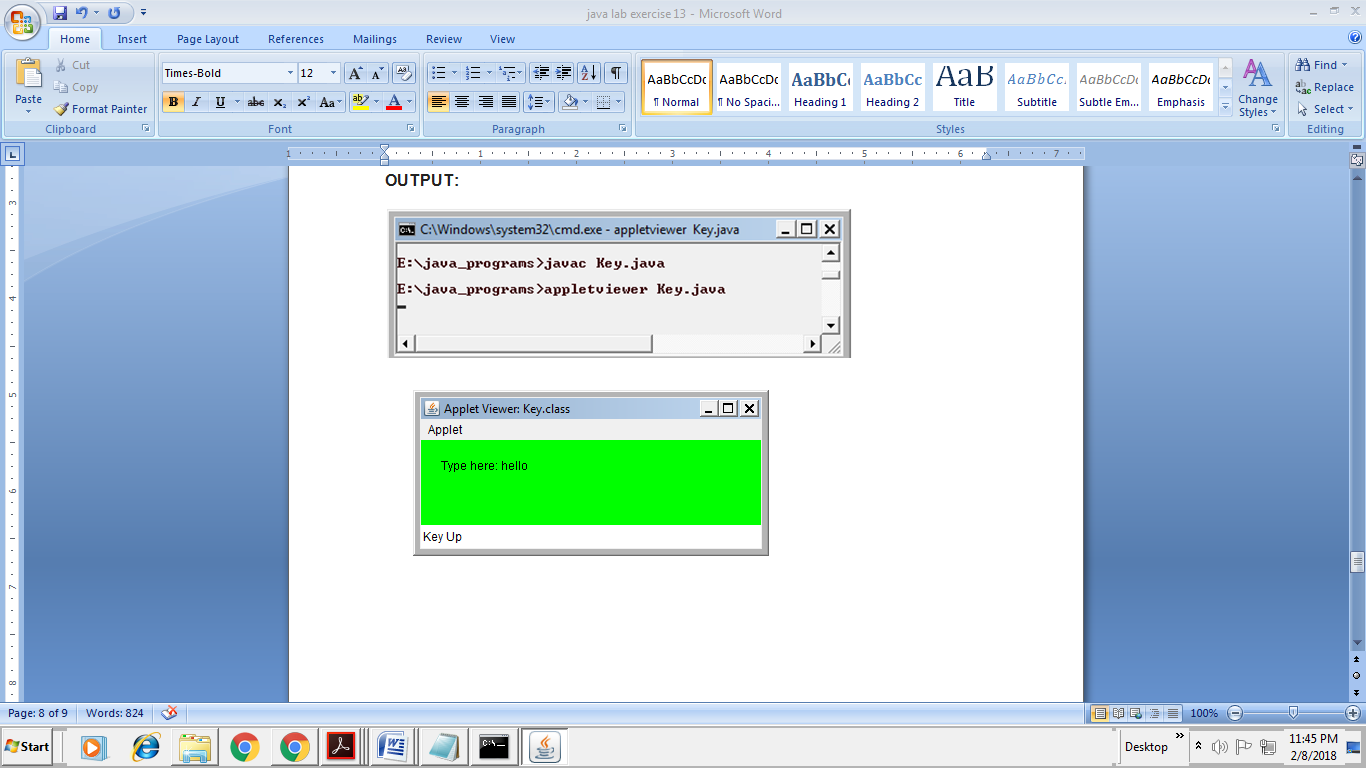
g.drawString(msg,X,Y);

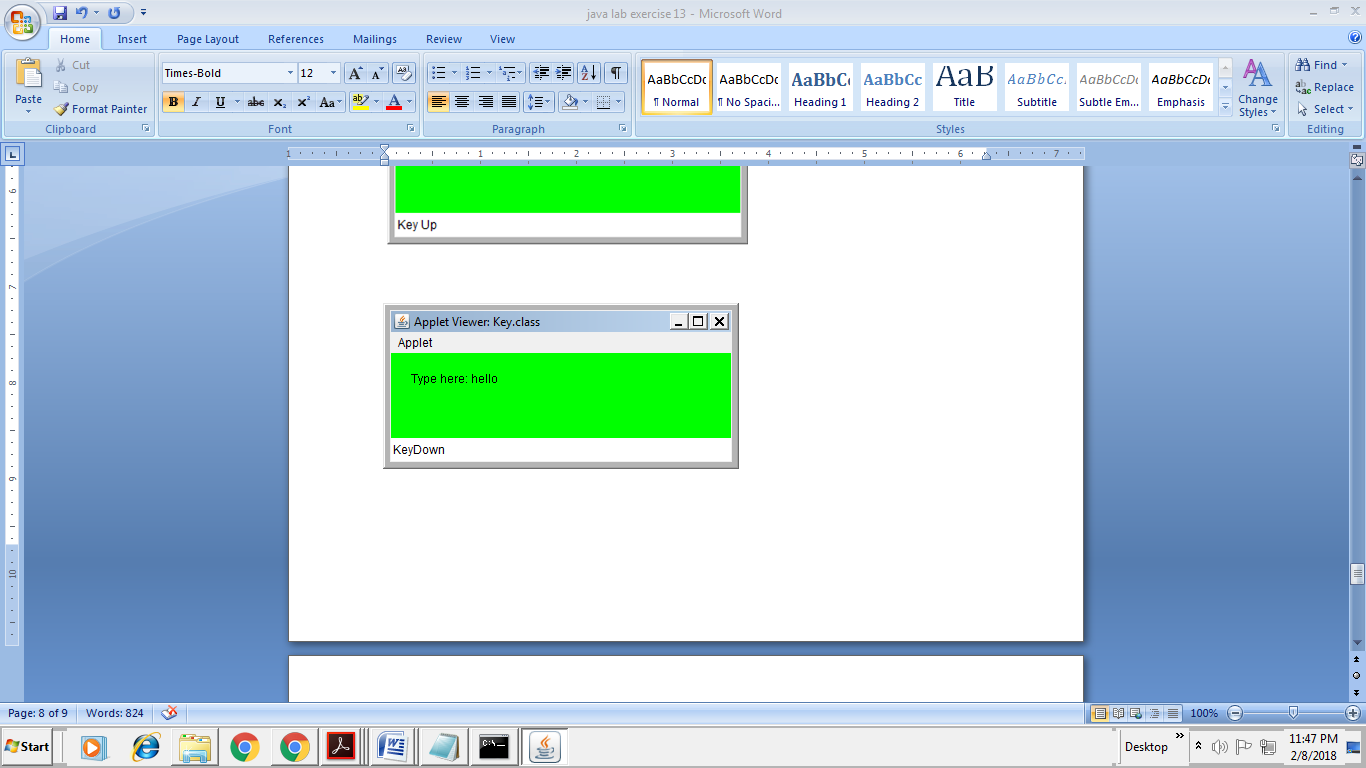
}

}

**OUTPUT:**

****

****

****

**Exercise - 15** (Swings)

15 a).Write a JAVA program to build a Calculator in Swings

**AIM:** JAVA program to build a Calculator in Swings

**ALGORITHM:**

1. Import packages
2. Creating an another seperate class CalculatorFrame
3. Creating all components and a container
4. Creating two double and one integer type variables
5. Adjusting alignment to right for text entered in textfield
6. Creating constructor of CalculatorFrame class
7. Setting size and location of components using setBounds() method
8. Adding components to container using add() method
9. Registering action listener to buttons and Performing actionPerformed() method for each registered buttons.
10. Created two functions enable and disable.
11. Creating a class Calculator
12. Creating object of CalculatorFrame class and setting some of its properties

**PROGRAM:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

//Creating an another seperate class CalculatorFrame(Step 4)

class CalculatorFrame extends JFrame implements ActionListener

{

/\* Creating all components and a container

that is going to be used in our program (Step 6)\*/

Container c;

JLabel lbl;

JTextField tf;

JButton zero;

JButton one;

JButton two;

JButton three;

JButton four;

JButton five;

JButton six;

JButton seven;

JButton eight;

JButton nine;

JButton dot;

JButton clear;

JButton backspace;

JButton equal;

JButton mul;

JButton div;

JButton plus;

JButton minus;

//Creating two double and one integer type variables(Step 7)

double num,ans;

int calculation;

// Creating constructor of CalculatorFrame class (Step 8)

CalculatorFrame()

{

//getting the content pane using getContentPane() method (Step 9)

c=this.getContentPane();

//Setting layout of the container to null (Step 10);

c.setLayout(null);

// Setting the background color of container (Step 11)

c.setBackground(Color.PINK);

// creating objects of all components (Step 12)

tf=new JTextField();

lbl=new JLabel();

seven=new JButton("7");

eight=new JButton("8");

nine=new JButton("9");

four=new JButton("4");

five=new JButton("5");

six=new JButton("6");

one=new JButton("1");

two=new JButton("2");

three=new JButton("3");

dot=new JButton(".");

zero=new JButton("0");

equal=new JButton("=");

plus=new JButton("+");

minus=new JButton("-");

mul=new JButton("x");

div=new JButton("/");

clear=new JButton("C");

backspace=new JButton("<--");

//Adjusting alignment to right for text entered in textfield(Step 13)

tf.setHorizontalAlignment(SwingConstants.RIGHT);

//Setting font size for components(Step 15)

Font fo=new Font("Arial",Font.BOLD,20);

tf.setFont(fo);

seven.setFont(fo);

eight.setFont(fo);

nine.setFont(fo);

four.setFont(fo);

five.setFont(fo);

six.setFont(fo);

one.setFont(fo);

two.setFont(fo);

three.setFont(fo);

dot.setFont(fo);

zero.setFont(fo);

equal.setFont(fo);

plus.setFont(fo);

minus.setFont(fo);

mul.setFont(fo);

div.setFont(fo);

clear.setFont(fo);

Font font=new Font("Arial",Font.BOLD,15);

backspace.setFont(font);

/\*Setting size and location of components

using setBounds() method(Step 16)\*/

tf.setBounds(10,40,270,40);

lbl.setBounds(250,0,50,50);

clear.setBounds(80,110,60,40);

seven.setBounds(10,170,60,40);

eight.setBounds(80,170,60,40);

nine.setBounds(150,170,60,40);

four.setBounds(10,230,60,40);

five.setBounds(80,230,60,40);

six.setBounds(150,230,60,40);

one.setBounds(10,290,60,40);

two.setBounds(80,290,60,40);

three.setBounds(150,290,60,40);

dot.setBounds(10,350,60,40);

zero.setBounds(80,350,60,40);

equal.setBounds(150,350,60,40);

plus.setBounds(220,110,60,40);

minus.setBounds(220,170,60,40);

mul.setBounds(220,230,60,40);

div.setBounds(220,290,60,40);

backspace.setBounds(150,110,60,40);

/\*Adding components to container using

add() method(Step 17)\*/

c.add(tf);

c.add(lbl);

c.add(seven);

c.add(eight);

c.add(nine);

c.add(four);

c.add(five);

c.add(six);

c.add(one);

c.add(two);

c.add(three);

c.add(dot);

c.add(zero);

c.add(equal);

c.add(plus);

c.add(minus);

c.add(mul);

c.add(div);

c.add(clear);

c.add(backspace);

//Registering action listener to buttons(Step 18)

seven.addActionListener(this);

eight.addActionListener(this);

nine.addActionListener(this);

four.addActionListener(this);

five.addActionListener(this);

six.addActionListener(this);

one.addActionListener(this);

two.addActionListener(this);

three.addActionListener(this);

dot.addActionListener(this);

zero.addActionListener(this);

equal.addActionListener(this);

plus.addActionListener(this);

minus.addActionListener(this);

mul.addActionListener(this);

div.addActionListener(this);

clear.addActionListener(this);

backspace.addActionListener(this);

}

// Performing actionPerformed() method for each registered buttons(Step 19)

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==clear) {

tf.setText(" ");

lbl.setText(" ");

}

else if(e.getSource()==backspace) {

int length=tf.getText().length();

int number=length-1;

if(length>0) {

StringBuilder back=new StringBuilder(tf.getText());

back.deleteCharAt(number);

tf.setText(back.toString());

}

}

else if(e.getSource()==zero) {

tf.setText(tf.getText()+"0");

}

else if(e.getSource()==one) {

tf.setText(tf.getText()+"1");

}

else if(e.getSource()==two) {

tf.setText(tf.getText()+"2");

}

else if(e.getSource()==three) {

tf.setText(tf.getText()+"3");

}

else if(e.getSource()==four) {

tf.setText(tf.getText()+"4");

}

else if(e.getSource()==five) {

tf.setText(tf.getText()+"5");

}

else if(e.getSource()==six) {

tf.setText(tf.getText()+"6");

}

else if(e.getSource()==seven) {

tf.setText(tf.getText()+"7");

}

else if(e.getSource()==eight) {

tf.setText(tf.getText()+"8");

}

else if(e.getSource()==nine) {

tf.setText(tf.getText()+"9");

}

else if(e.getSource()==dot) {

tf.setText(tf.getText()+".");

}

else if(e.getSource()==plus) {

num=Double.parseDouble(tf.getText());

tf.setText(" ");

lbl.setText(num+"+");

calculation=1;

}

else if(e.getSource()==minus) {

num=Double.parseDouble(tf.getText());

tf.setText(" ");

lbl.setText(num+"-");

calculation=2;

}

else if(e.getSource()==mul) {

num=Double.parseDouble(tf.getText());

tf.setText(" ");

lbl.setText(num+"x");

calculation=3;

}

else if(e.getSource()==div) {

num=Double.parseDouble(tf.getText());

tf.setText(" ");

lbl.setText(num+"/");

calculation=4;

}

else if(e.getSource()==equal) {

switch(calculation)

{

case 1: ans=num + Double.parseDouble(tf.getText());

tf.setText(Double.toString(ans));

lbl.setText(" ");

break;

case 2: ans=num - Double.parseDouble(tf.getText());

tf.setText(Double.toString(ans));

lbl.setText(" ");

break;

case 3: ans=num \* Double.parseDouble(tf.getText());

tf.setText(Double.toString(ans));

lbl.setText(" ");

break;

case 4: ans=num / Double.parseDouble(tf.getText());

tf.setText(Double.toString(ans));

lbl.setText(" ");

break;

}

}

}

public void disable()

{

tf.setText(" ");

lbl.setText(" ");

tf.setEnabled(false);

dot.setEnabled(false);

zero.setEnabled(false);

one.setEnabled(false);

two.setEnabled(false);

three.setEnabled(false);

four.setEnabled(false);

five.setEnabled(false);

six.setEnabled(false);

seven.setEnabled(false);

eight.setEnabled(false);

nine.setEnabled(false);

plus.setEnabled(false);

minus.setEnabled(false);

mul.setEnabled(false);

div.setEnabled(false);

equal.setEnabled(false);

clear.setEnabled(false);

backspace.setEnabled(false);

}

public void enable()

{

tf.setEnabled(true);

dot.setEnabled(true);

zero.setEnabled(true);

one.setEnabled(true);

two.setEnabled(true);

three.setEnabled(true);

four.setEnabled(true);

five.setEnabled(true);

six.setEnabled(true);

seven.setEnabled(true);

eight.setEnabled(true);

nine.setEnabled(true);

plus.setEnabled(true);

minus.setEnabled(true);

mul.setEnabled(true);

div.setEnabled(true);

equal.setEnabled(true);

clear.setEnabled(true);

backspace.setEnabled(true);

}

}

// Creating a class Calculator(Step 2)

class Calculator {

//Creating main method (Step 3)

public static void main(String args[]) {

/\*Creating object of CalculatorFrame

class and setting some of its properties (step 5) \*/

CalculatorFrame f=new CalculatorFrame();

f.setTitle("Calculator");

f.setVisible(true);

f.setBounds(100,100,300,440);

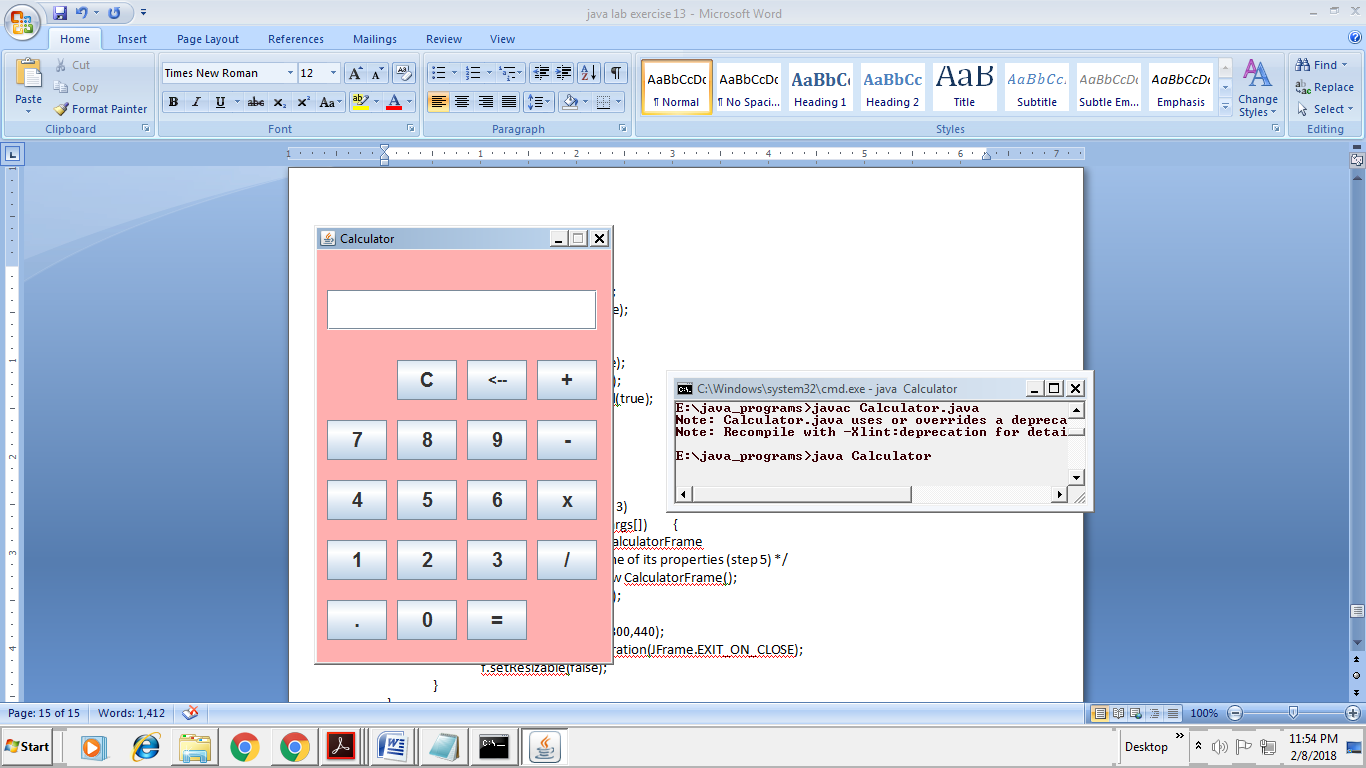
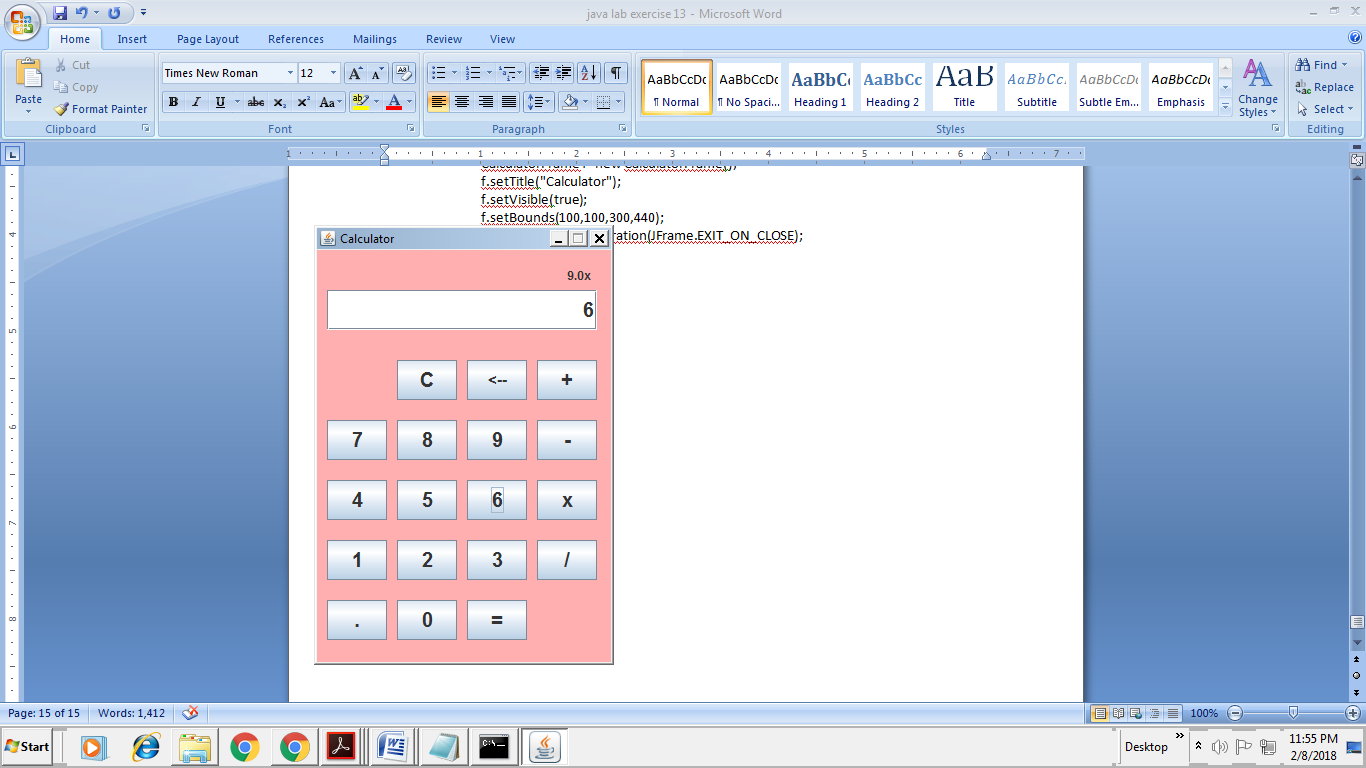
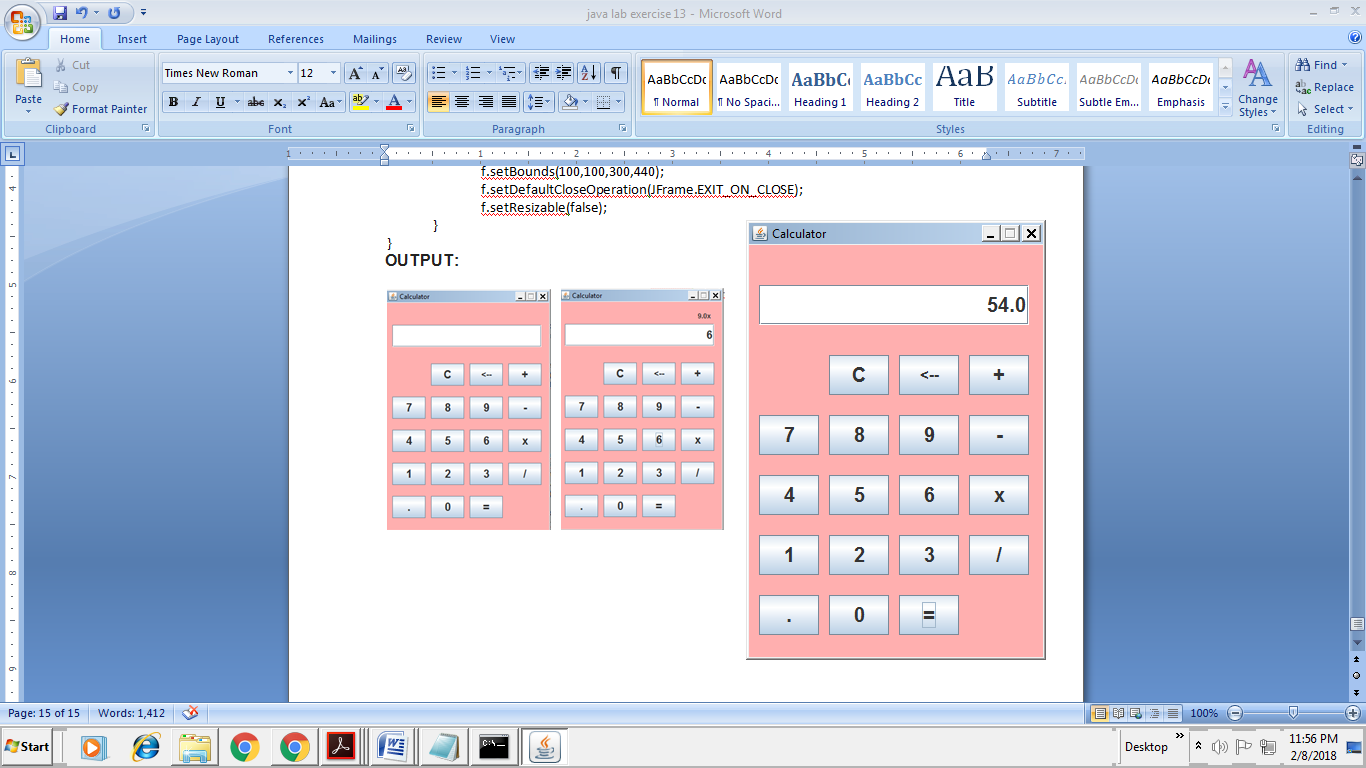
f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setResizable(false);

}

}

**OUTPUT:**

15 b). Write a JAVA program to display the digital watch in swing tutorial.

**AIM:** JAVA program to display the digital watch in swing tutorial.

**ALGORITHM:**

1. Import packages.
2. Create class DigitalClock
3. Declare main method and created objects for class ClockLabel
4. Created class ClockLabel extends from JLabel and implements from ActionListener
5. Create ClockLabel(String type) function and actionPerformed(ActionEvent ae) function to perform action and to display digital watch.

**PROGRAM:**

import java.awt.Font;

import java.awt.Color;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.Timer;

import javax.swing.SwingConstants;

import java.util.\*;

import java.text.\*;

public class DigitalClock {

public static void main(String[] arguments) {

ClockLabel dateLable = new ClockLabel("date");

ClockLabel timeLable = new ClockLabel("time");

ClockLabel dayLable = new ClockLabel("day");

JFrame.setDefaultLookAndFeelDecorated(true);

JFrame f = new JFrame("Digital Clock");

f.setSize(300,150);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setLayout(new GridLayout(3, 1));

f.add(dateLable);

f.add(timeLable);

f.add(dayLable);

f.getContentPane().setBackground(Color.black);

f.setVisible(true);

}

}

class ClockLabel extends JLabel implements ActionListener {

String type;

SimpleDateFormat sdf;

public ClockLabel(String type) {

this.type = type;

setForeground(Color.green);

switch (type) {

case "date" : sdf = new SimpleDateFormat(" MMMM dd yyyy");

setFont(new Font("sans-serif", Font.PLAIN, 12));

setHorizontalAlignment(SwingConstants.LEFT);

break;

case "time" : sdf = new SimpleDateFormat("hh:mm:ss a");

setFont(new Font("sans-serif", Font.PLAIN, 40));

setHorizontalAlignment(SwingConstants.CENTER);

break;

case "day" : sdf = new SimpleDateFormat("EEEE ");

setFont(new Font("sans-serif", Font.PLAIN, 16));

setHorizontalAlignment(SwingConstants.RIGHT);

break;

default : sdf = new SimpleDateFormat();

break;

}

Timer t = new Timer(1000, this);

t.start();

}

public void actionPerformed(ActionEvent ae) {

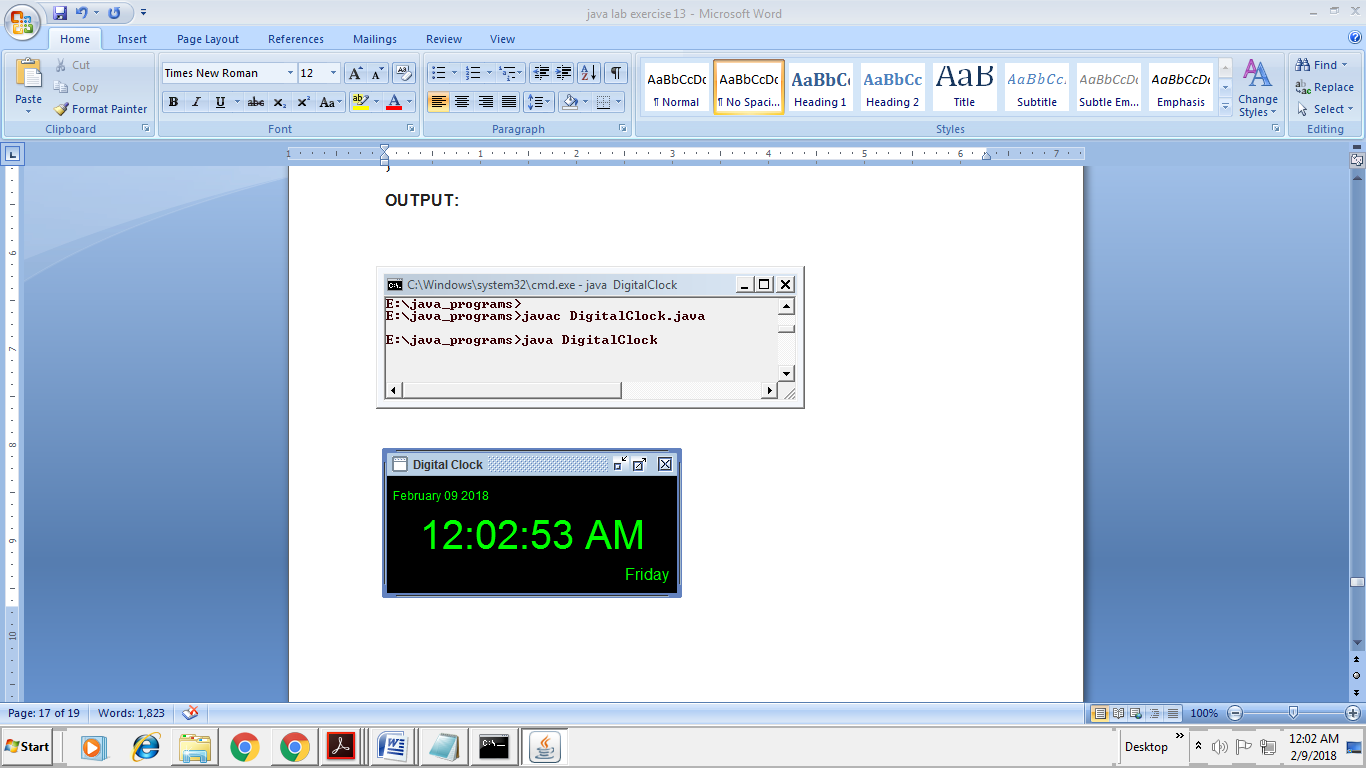
Date d = new Date();

setText(sdf.format(d));

}

}

**OUTPUT:**



**Exercise – 16 (Swings - Continued)**

16 a). Write a JAVA program that to create a single ball bouncing inside a JPanel.

**AIM:** JAVA program to a single ball bouncing inside a JPanel.

**ALGORITHM:**

1. Import packages
2. Create a class with named BouncingBall and it extends from JPanel
3. Declared variables are width, height, radius and size.
4. Create a constructor BouncingBall().
5. Create a function paintComponent(Graphics g) to fill the colour.
6. Declare a main method and access with objects.

**PROGRAM:**

import java.awt.\*;

import javax.swing.\*;

public class BouncingBall extends JPanel {

// Box height and width

int width;

int height;

// Ball Size

float radius = 10;

float diameter = radius \* 2;

// Center of Call

float X = radius + 50;

float Y = radius + 20;

// Direction

float dx = 3;

float dy = 3;

public BouncingBall() {

Thread thread = new Thread() {

public void run() {

while (true) {

width = getWidth();

height = getHeight();

X = X + dx ;

Y = Y + dy;

if (X - radius < 0) {

dx = -dx;

X = radius;

} else if (X + radius > width) {

dx = -dx;

X = width - radius;

}

if (Y - radius < 0) {

dy = -dy;

Y = radius;

} else if (Y + radius > height) {

dy = -dy;

Y = height - radius;

}

repaint();

try {

Thread.sleep(50);

} catch (InterruptedException ex) {

}

}

}

};

thread.start();

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(Color.BLUE);

g.fillOval((int)(X-radius), (int)(Y-radius), (int)diameter, (int)diameter);

}

public static void main(String[] args) {

JFrame.setDefaultLookAndFeelDecorated(true);

JFrame frame = new JFrame("Bouncing Ball");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(300, 200);

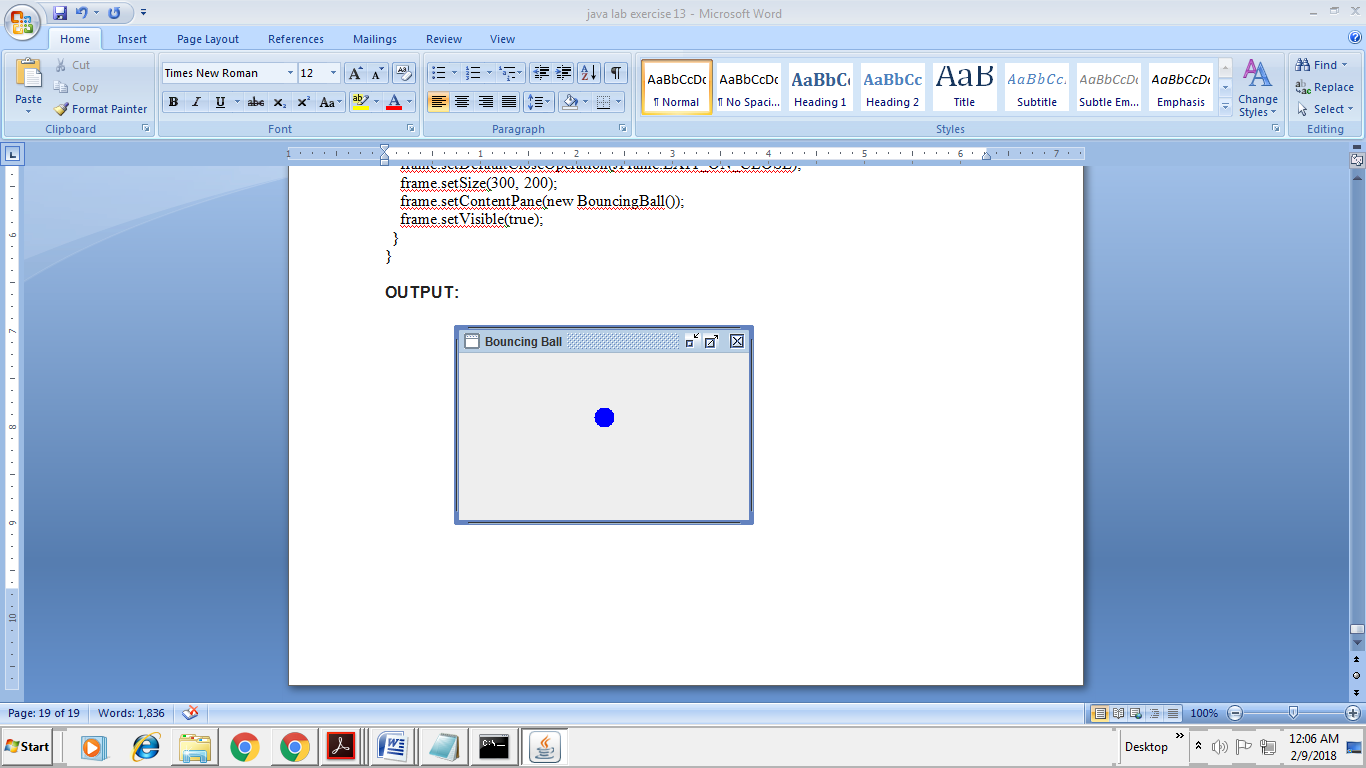
frame.setContentPane(new BouncingBall());

frame.setVisible(true);

}

}

**OUTPUT:**



16 b). Write a JAVA program JTree as displaying a real tree upside down

**AIM:** JAVA program JTree as displaying a real tree upside down

**ALGORITHM:**

1. Import packages
2. Create class JTreeDemo extends from JFrame
3. Declaring three functions in JTreeDemo class are JTree jt, DefaultTreeModel dtm, DefaultMutableTreeNode dtm1,JTreeDemo(String title)
4. Using the DefaultMutableTreeNode class to represent our nodes(one, oneInNumber,oneInNumber2)
5. Initialize expansionListener functions to perform expansion eventtreeCollapsed(TreeExpansionEvent te) and treeExpanded( TreeExpansionEvent te) and close class.
6. To exit JFrame using setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
7. Declare a main method.

**PROGRAM:**

import javax.swing.\*;

import javax.swing.tree.\*;

import javax.swing.event.\*;

public class JTreeDemo extends JFrame

{

JTree jt;

DefaultTreeModel dtm;

DefaultMutableTreeNode dtm1;

JTreeDemo(String title)

{

setTitle(title);

dtm1 = new DefaultMutableTreeNode("Numbers");

DefaultMutableTreeNode one=new DefaultMutableTreeNode ("One");

DefaultMutableTreeNode oneInNumber=new DefaultMutableTreeNode("1");

DefaultMutableTreeNode oneInNumber2=new DefaultMutableTreeNode("1.1", false);

oneInNumber.add(oneInNumber2);

one.add(oneInNumber);

dtm1.add(one);

DefaultMutableTreeNode two=new DefaultMutableTreeNode("Two");

DefaultMutableTreeNode twoInNumber=new DefaultMutableTreeNode("2");

two.add(twoInNumber);

dtm1.add(two);

dtm = new DefaultTreeModel (dtm1,true);

jt = new JTree(dtm1);

jt.addTreeExpansionListener(new TreeExpansionListener()

{

public void treeCollapsed(TreeExpansionEvent te)

{

System.out.println("Collapsed");

}

public void treeExpanded( TreeExpansionEvent te)

{

System.out.println(jt.getLeadSelectionPath());

System.out.println(jt.getLastSelectedPathComponent());

}

});

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

add(jt);

setSize(200,200);

setVisible(true);

}

public static void main(String args[])

{

new JTreeDemo("My Tree"); } }

**OUTPUT:**

