**EXPERIMENT 2**

1. How to handle declaration of two or more class declaration.

class Student

{

int id;

String name;

public void print()

{

System.out.println(id);

System.out.println(name);

}

}

class Demo

{

public static void main(String[] args) {

Student x = new Student();

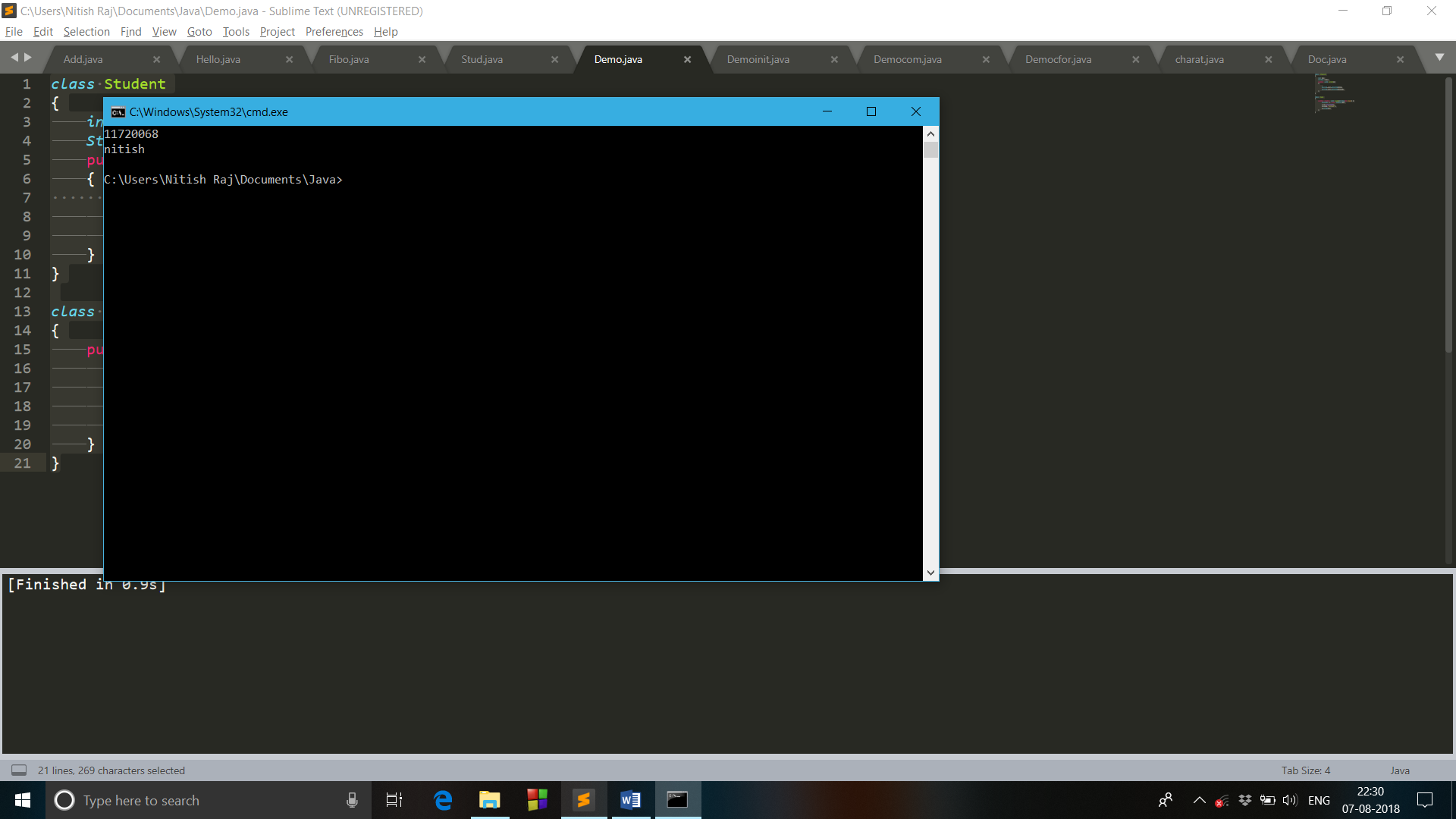
x.id=11720068;

x.name="nitish";

x.print();

}

}



1. Verify that args is an identifier or not?

class Democom

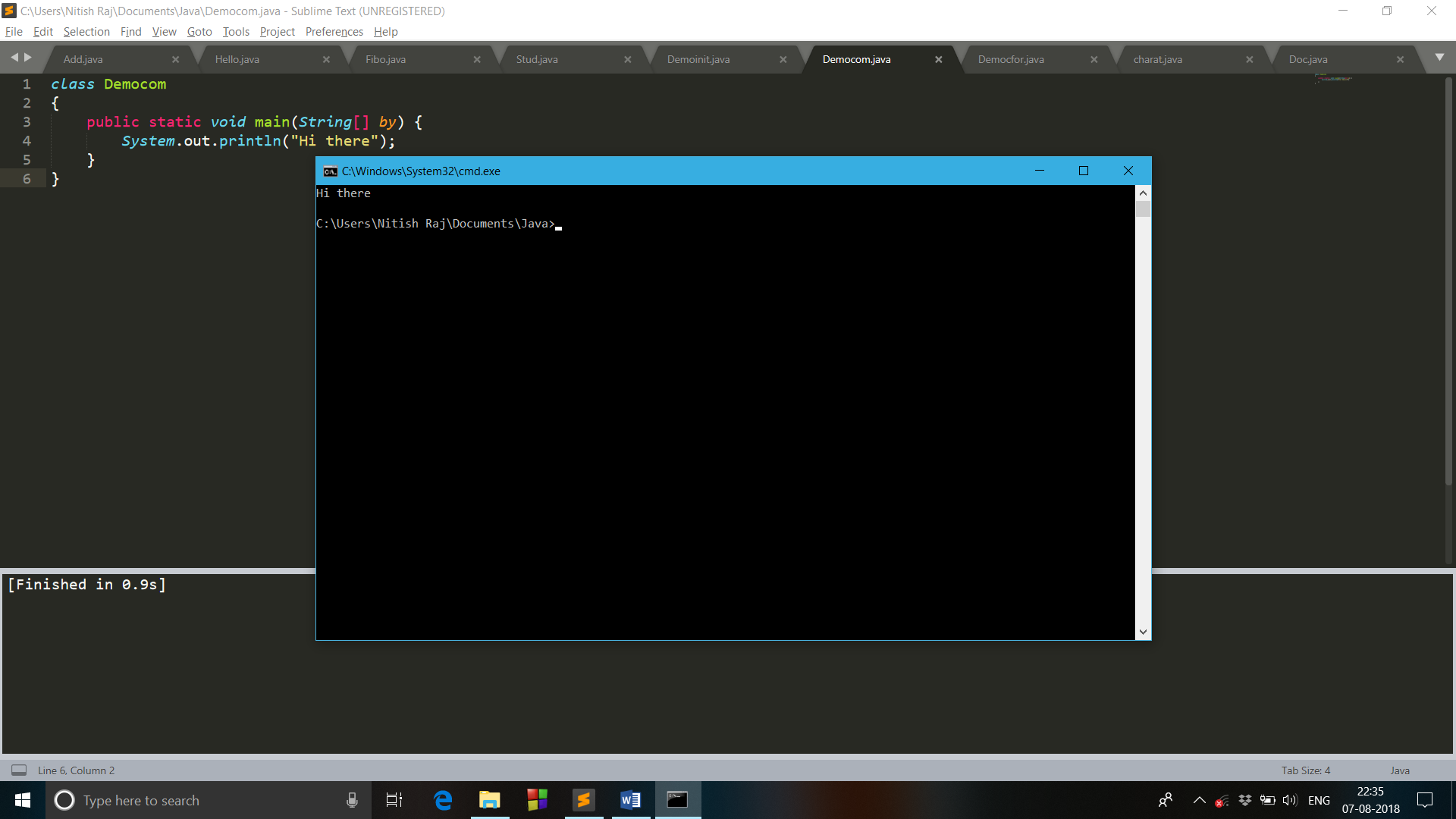
{

public static void main(String[] by) {

System.out.println("Hi there");

}

}



1. Write a Java program that’ll prove that the local variable must be explicitly initialize before being use.

class Student

{

public void print()

{

int id

String name;

System.out.println(id);

System.out.println(name);

}

}

class Demoinit

{

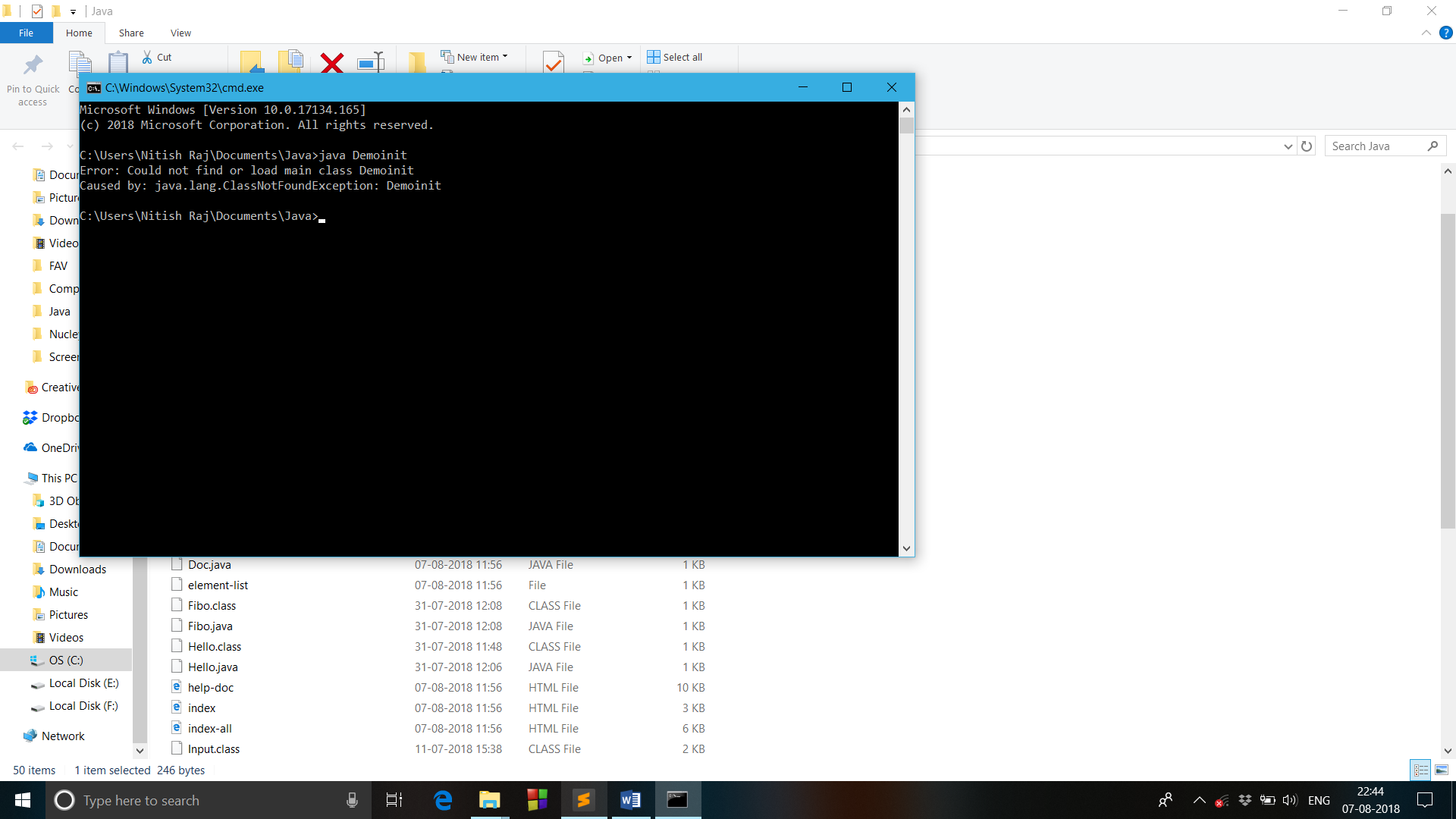
public static void main(String[] args) {

Student x = new Student();

x.print();

}

}



1. Write a Java program that show that the default value of fields are of their types.

class Student

{

int id;

String name;

public void print()

{

System.out.println(id);

System.out.println(name);

}

}

class Demo

{

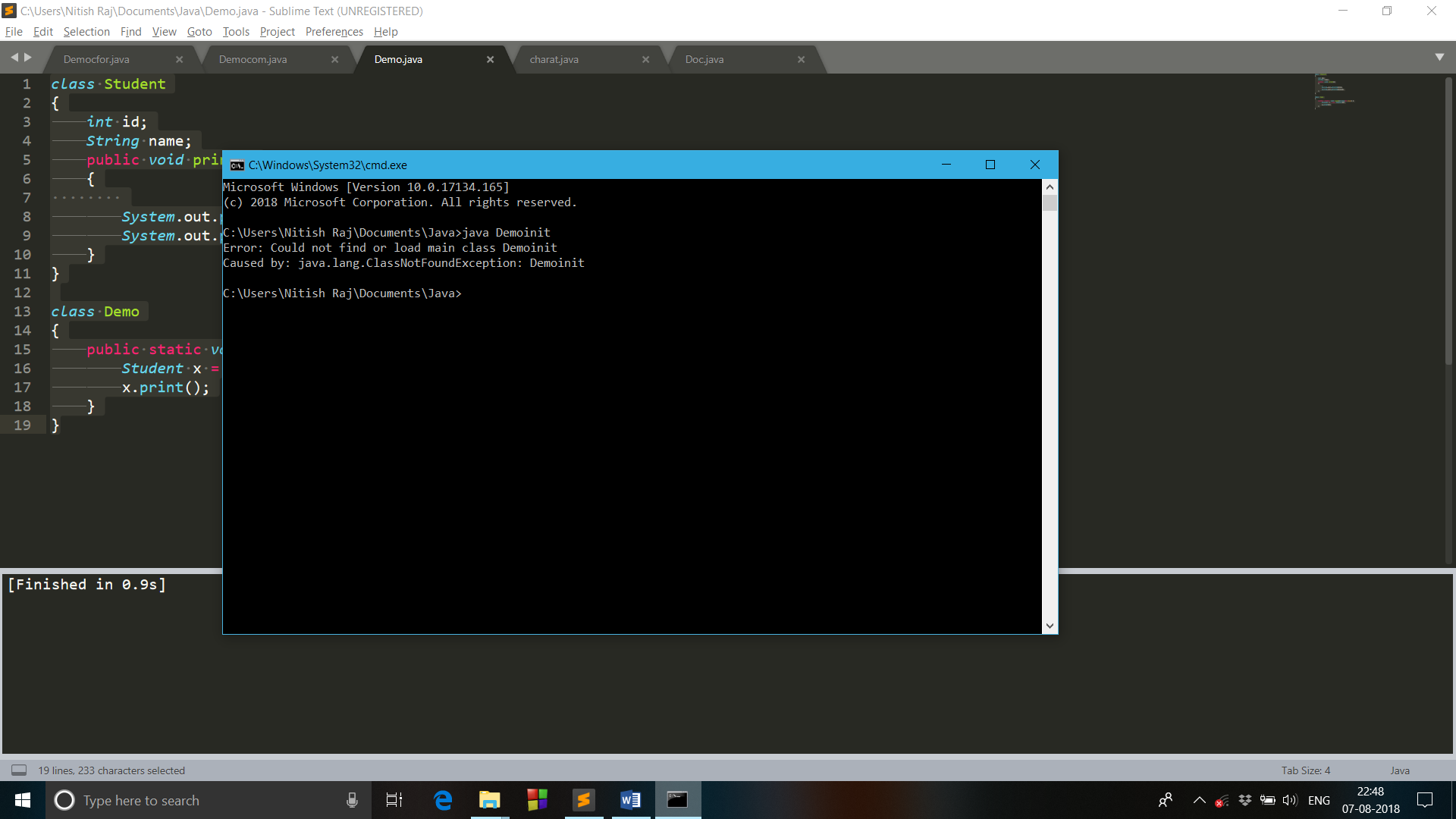
public static void main(String[] args) {

Student x = new Student();

x.print();

}

}



1. Prove or disprove the statement that nesting of multiline comments are allowed.

class Democom

{

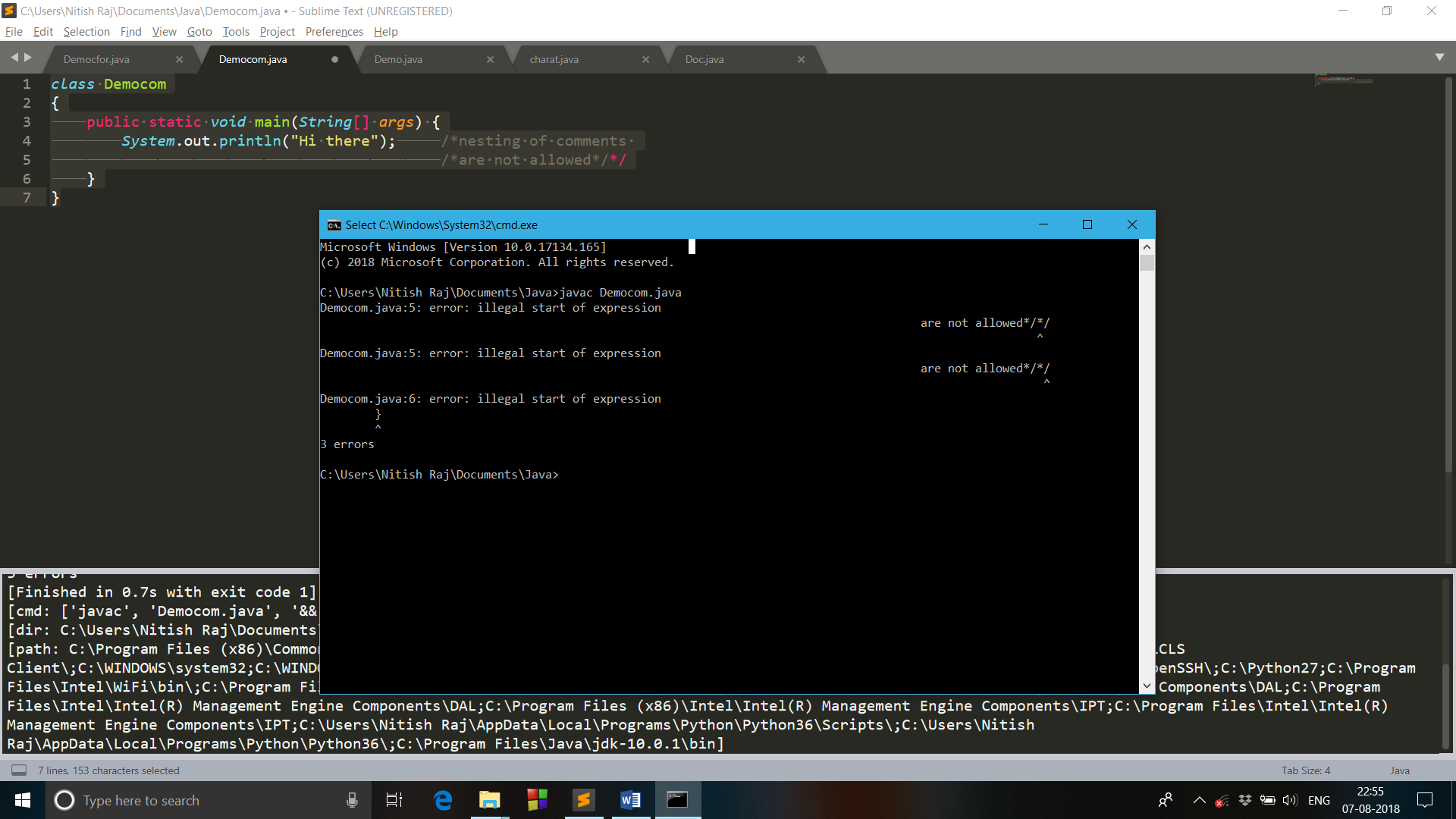
public static void main(String[] args) {

System.out.println("Hi there"); /\*nesting of comments

/\*are not allowed\*/\*/

}

}



1. Write a Java program for command line arguments.

class Democfor

{

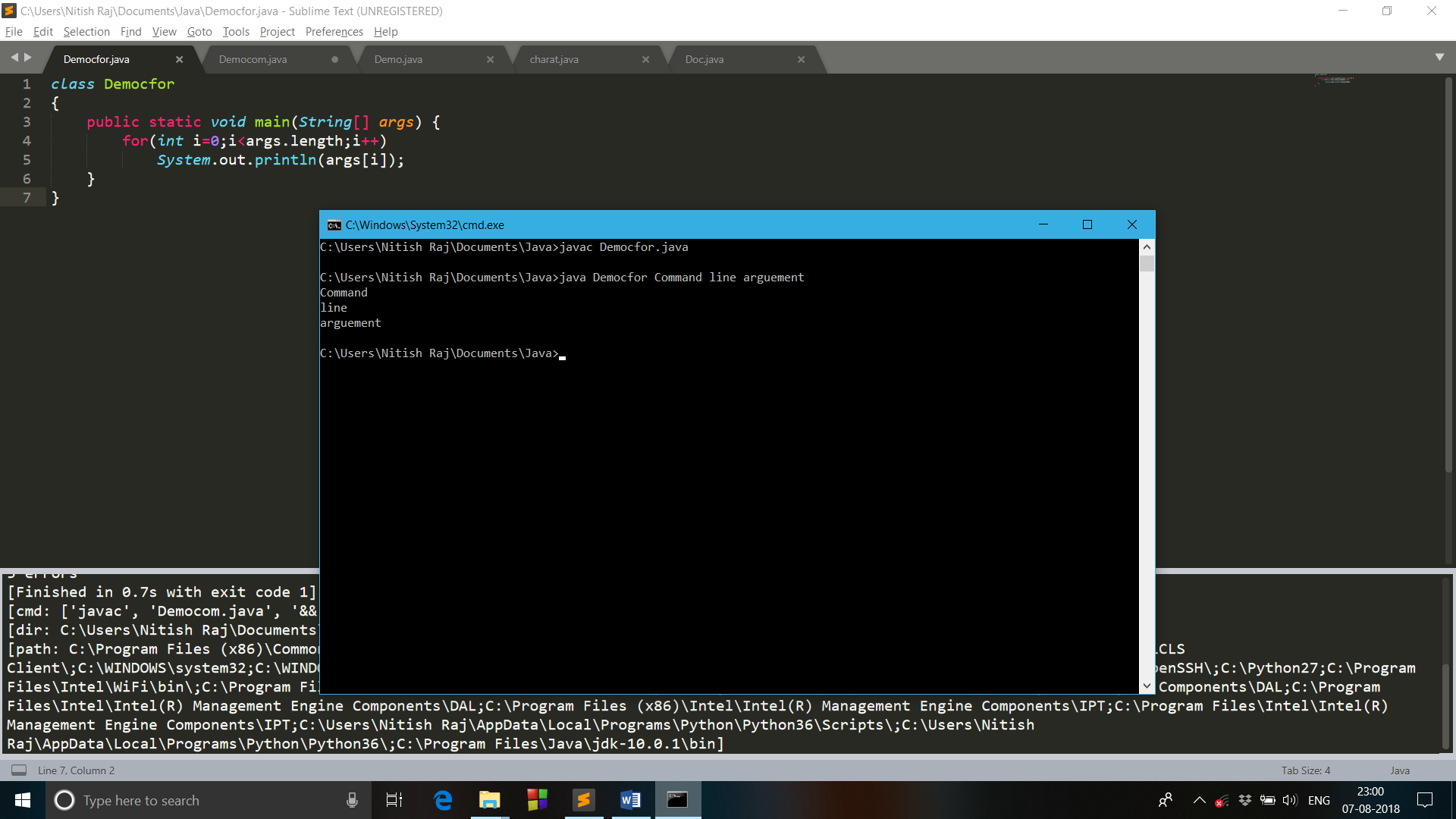
public static void main(String[] args) {

for(int i=0;i<args.length;i++)

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

}

}



1. Write a java program that demonstrate the use of Java Doc Tool wrt class declaration method and field declaration.

/\*\*hi there\*/

public class Doc

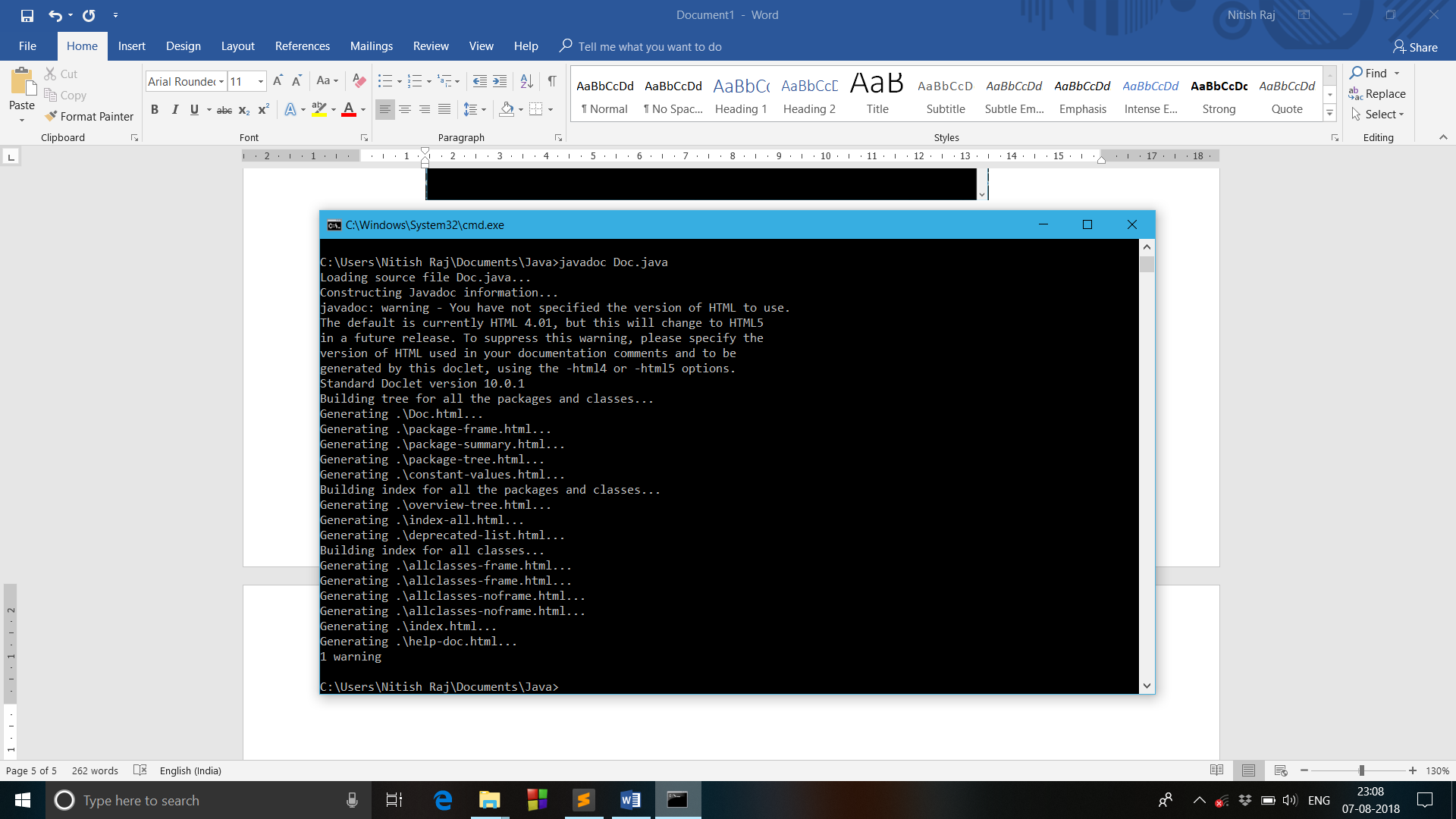
{

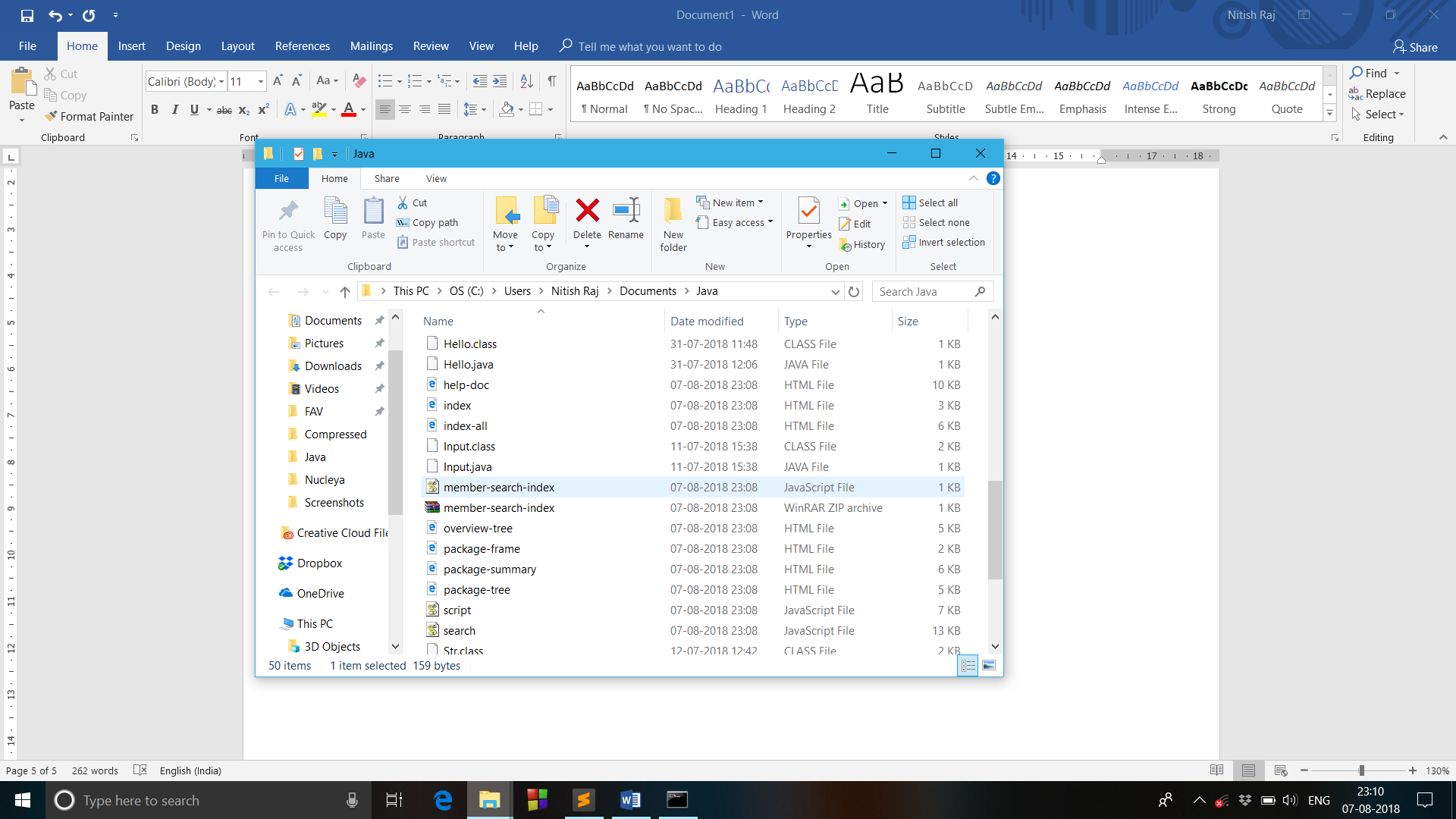
public static void main(String[] args) {

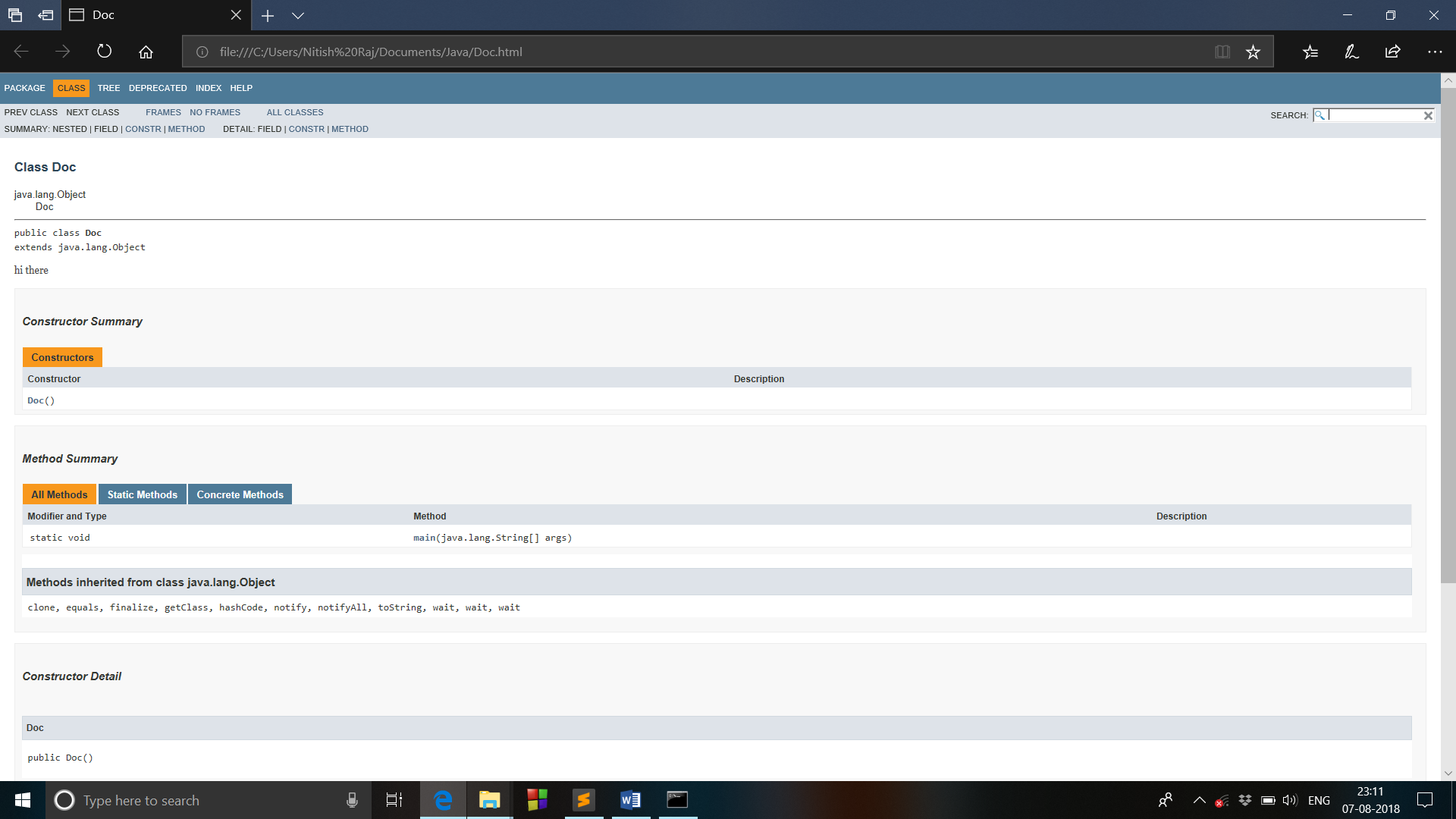
System.out.println("Java doc tool"); /\*\*this is it\*/

}

}







**EXPERIMENT 3**

1. Write a program to illustrate the use of object Aliasing.

class Aliasing

{

int roll;

String name;

public void print()

{

System.out.println(roll);

System.out.println(name);

}

public static void main(String[] args) {

Aliasing x = new Aliasing();

x.roll=11720068;

x.name="Nitish";

Aliasing y=x;

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

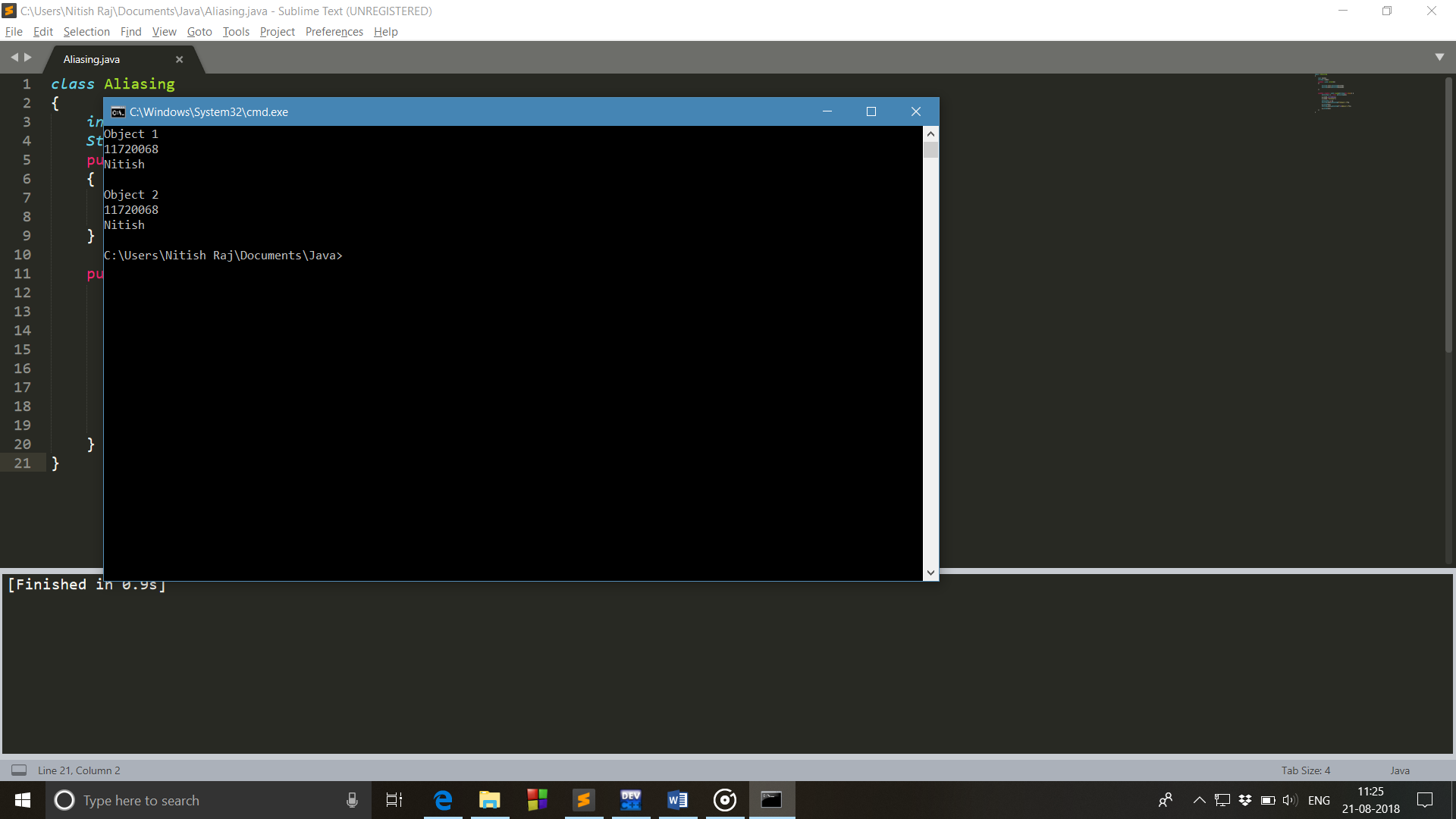
x.print();

System.out.println("\nObject 2");

y.print();

}

}



1. Static Field can be directly accessed in both static and non-static contexts.

class Staticfield

{

int roll;

String name;

static String section = "IT3";

Staticfield(int r,String n)

{

roll=r;

name=n;

}

void display()

{

System.out.println("Name is : "+name+" Roll no is : "+roll+" "+section);

}

public static void main(String[] args) {

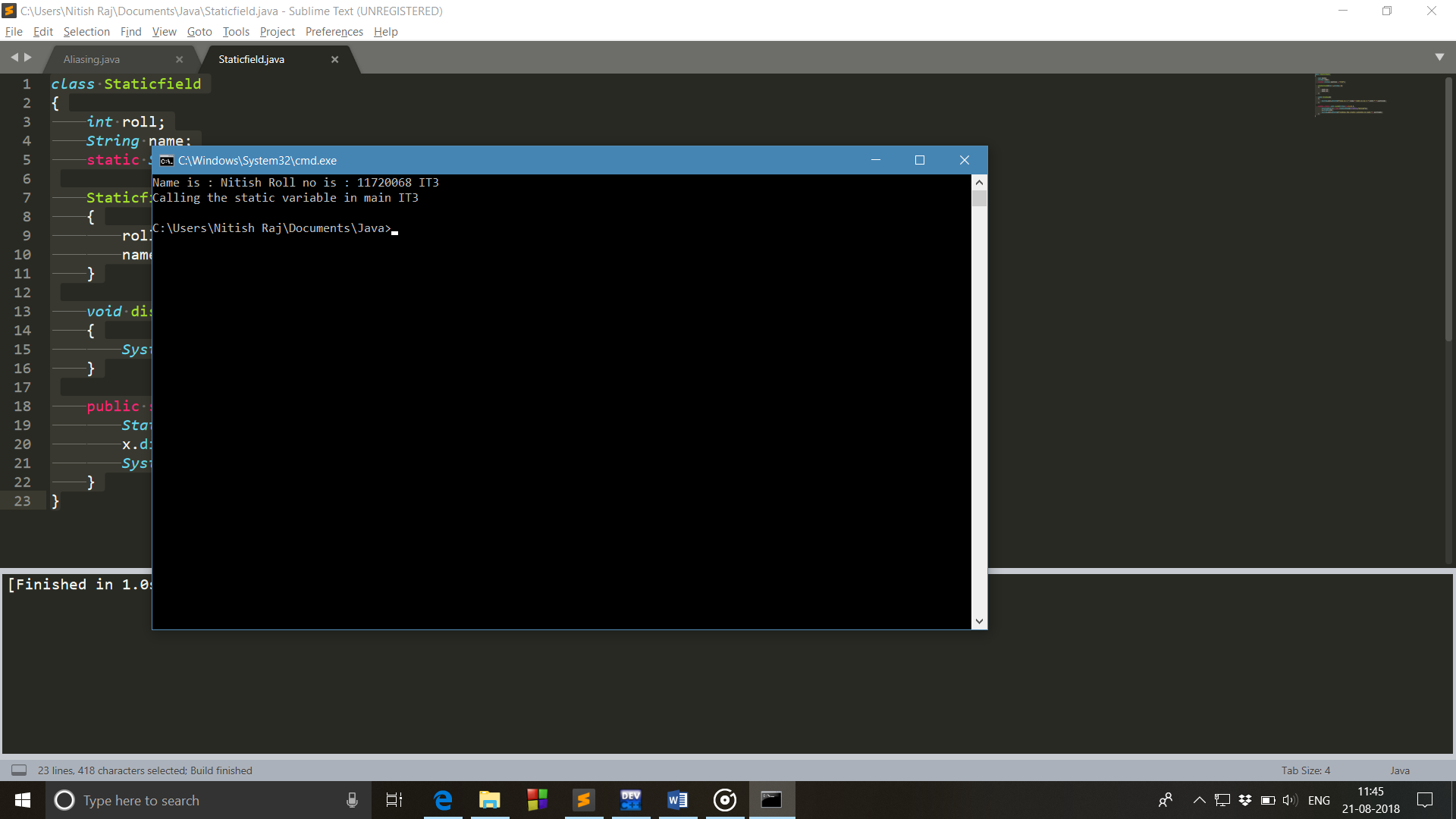
Staticfield x = new Staticfield(11720068,"Nitish");

x.display();

System.out.println("Calling the static variable in main "+ section);

}

}



1. Non-Static Field can be directly accessed in non-static contexts only.

class Nonstaticfield

{

int roll;

String name;

String section = "IT3";

Nonstaticfield(int r,String n)

{

roll=r;

name=n;

}

void display()

{

System.out.println("Name is : "+name+" Roll no is : "+roll+" "+section);

}

public static void main(String[] args) {

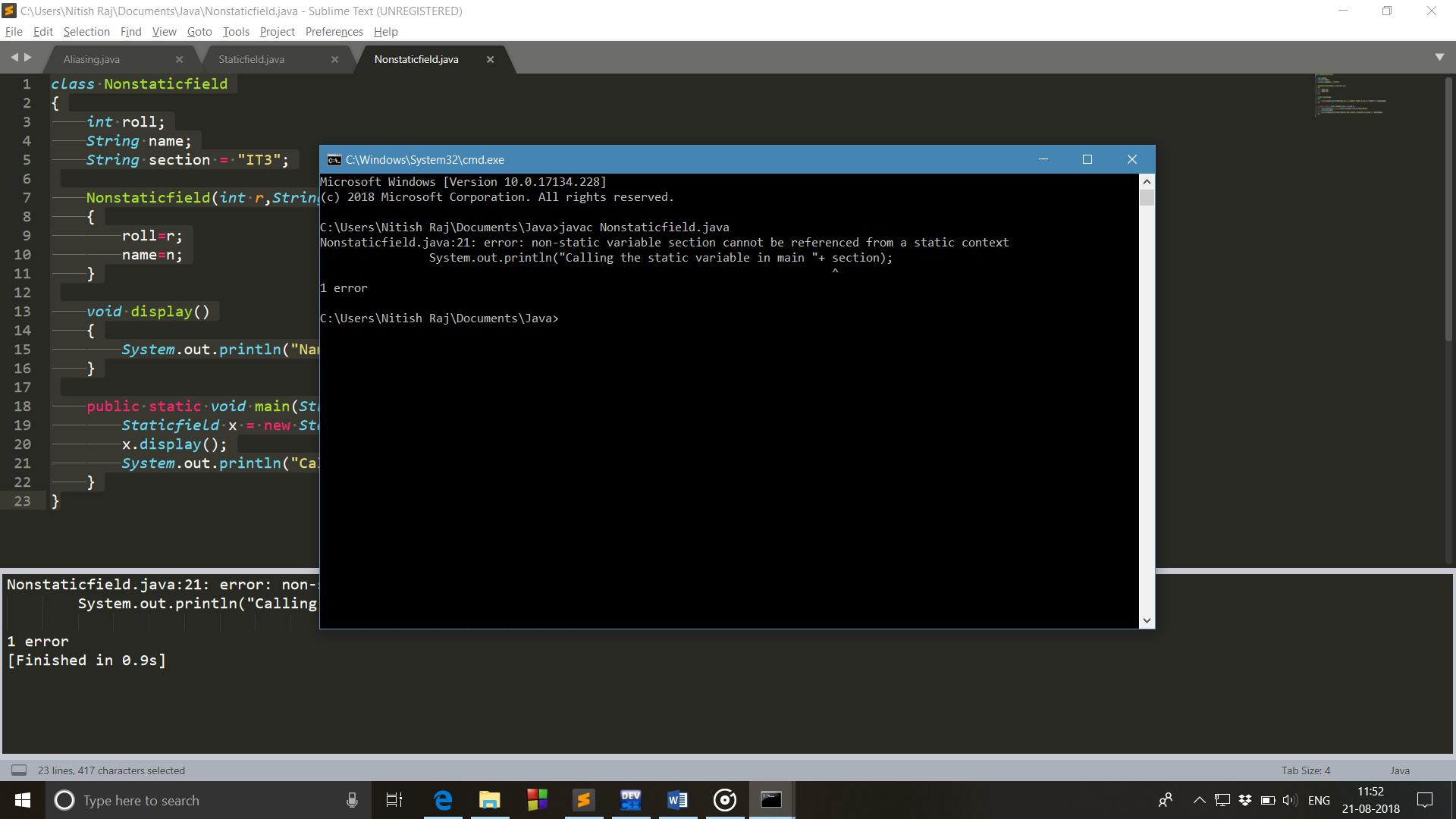
Staticfield x = new Staticfield(11720068,"Nitish");

x.display();

System.out.println("Calling the static variable in main "+ section);

}

}



1. WAP that demonstrate the class name and method name exists in different namespace.

class Mnameascname

{

String s;

public void Mnameascname()

{

s="Hi there";

}

public void print()

{

System.out.println(s);

}

public static void main(String[] args) {

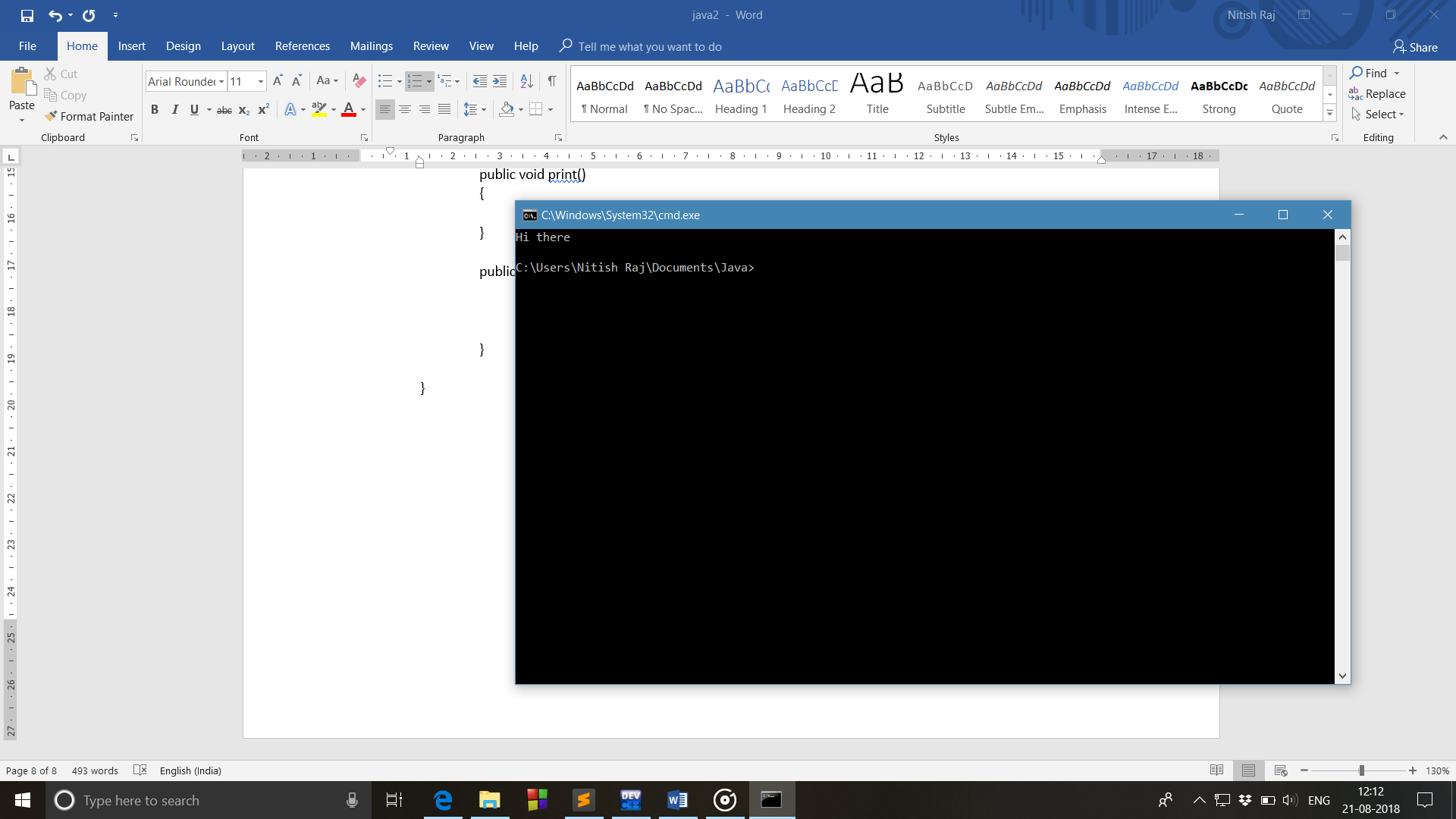
Mnameascname x = new Mnameascname();

x.Mnameascname();

x.print();

}

}



1. WAP to implement method overloading.