NANDINI SRIVASTAVA

17551A0598

**Assignment 4**

1. **Write a java program to create an user defined exception called PayOutOfBoundsException. This exception is thrown when basicpay is not in between 10000 and 30000.**

**package** com.rsc.stringpalindrome;

**import** java.util.\*;

**class** PayOutOfBoundsException **extends** Exception

{

**public** PayOutOfBoundsException(String s)

{

**super**(s);

}

}

**public** **class** Main

{

**public** **static** **void** main(String args[])

{

Scanner sc = **new** Scanner(System.***in***);

**int** basicpay = sc.nextInt();

**if**(basicpay<10000 || basicpay>30000)

**try**

{

**throw** **new** PayOutOfBoundsException("user defined error");

}

**catch** (PayOutOfBoundsException ex)

{

System.***out***.println("Caught");

System.***out***.println(ex.getMessage());

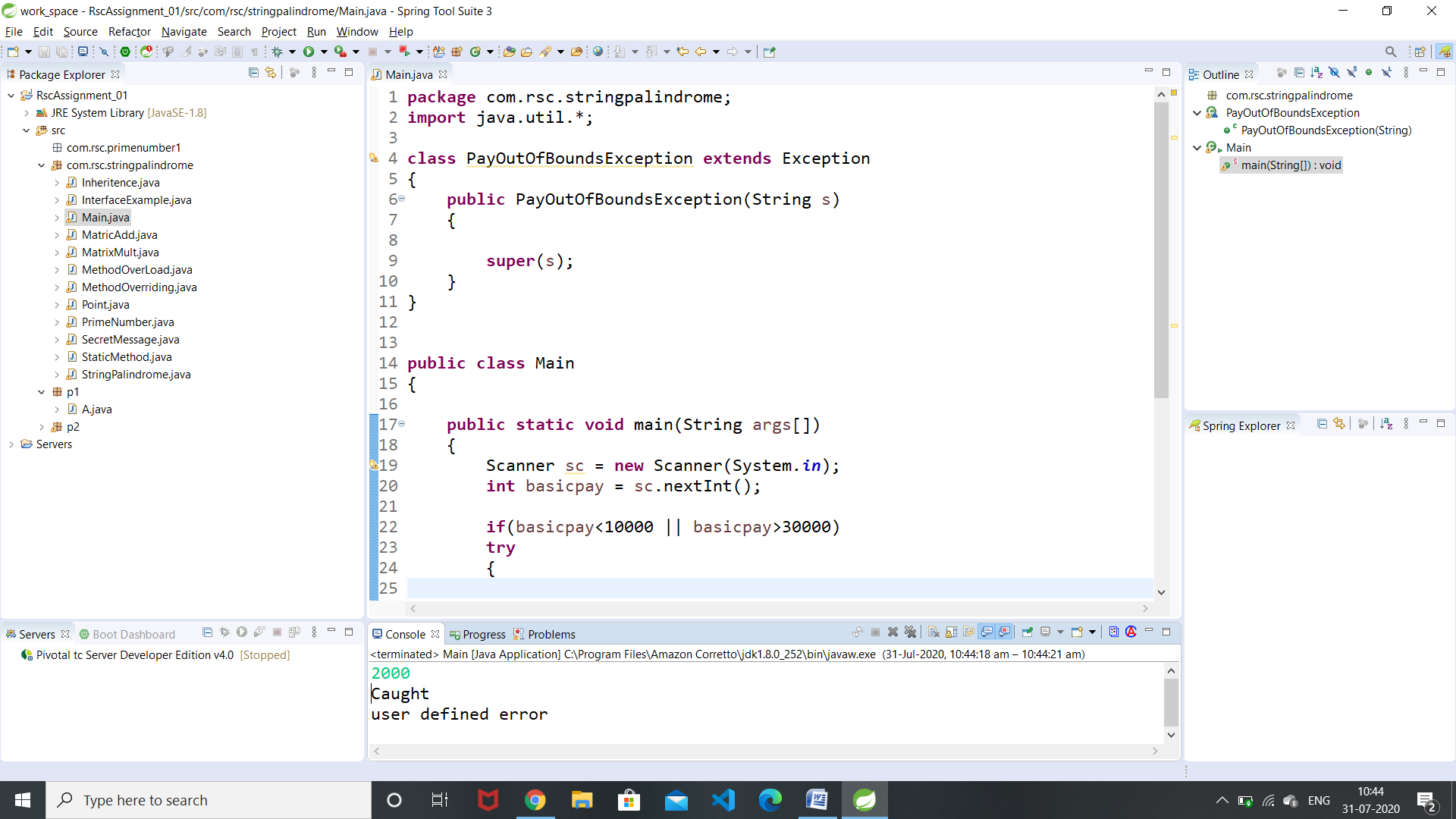
}

**else**

System.***out***.println("basicpay = "+ basicpay );

}

}



1. **Write a java program to create two threads which display a message every half second.**

**package** com.rsc.stringpalindrome;

**class** thread1 **extends** Thread

{

**public** **void** run()

{

**for**(**int** i=0;i<3;i++) {

String test = Thread.*currentThread*().getName();

System.***out***.println(test+" :- Hii");

**try**

{

Thread.*currentThread*().*sleep*(500);

}

**catch**(InterruptedException e)

{}

}

}

}

**class** thread2 **extends** Thread

{

**public** **void** run()

{

**for**(**int** i=0;i<3;i++) {

String test = Thread.*currentThread*().getName();

System.***out***.println(test+":- Hello..");

**try**

{

Thread.*currentThread*().*sleep*(500);

}

**catch**(InterruptedException e)

{}

}

}

}

**class** MainThread

{

**public** **static** **void** main(String args[])

{

System.***out***.println("\n");

thread1 obj1=**new** thread1();

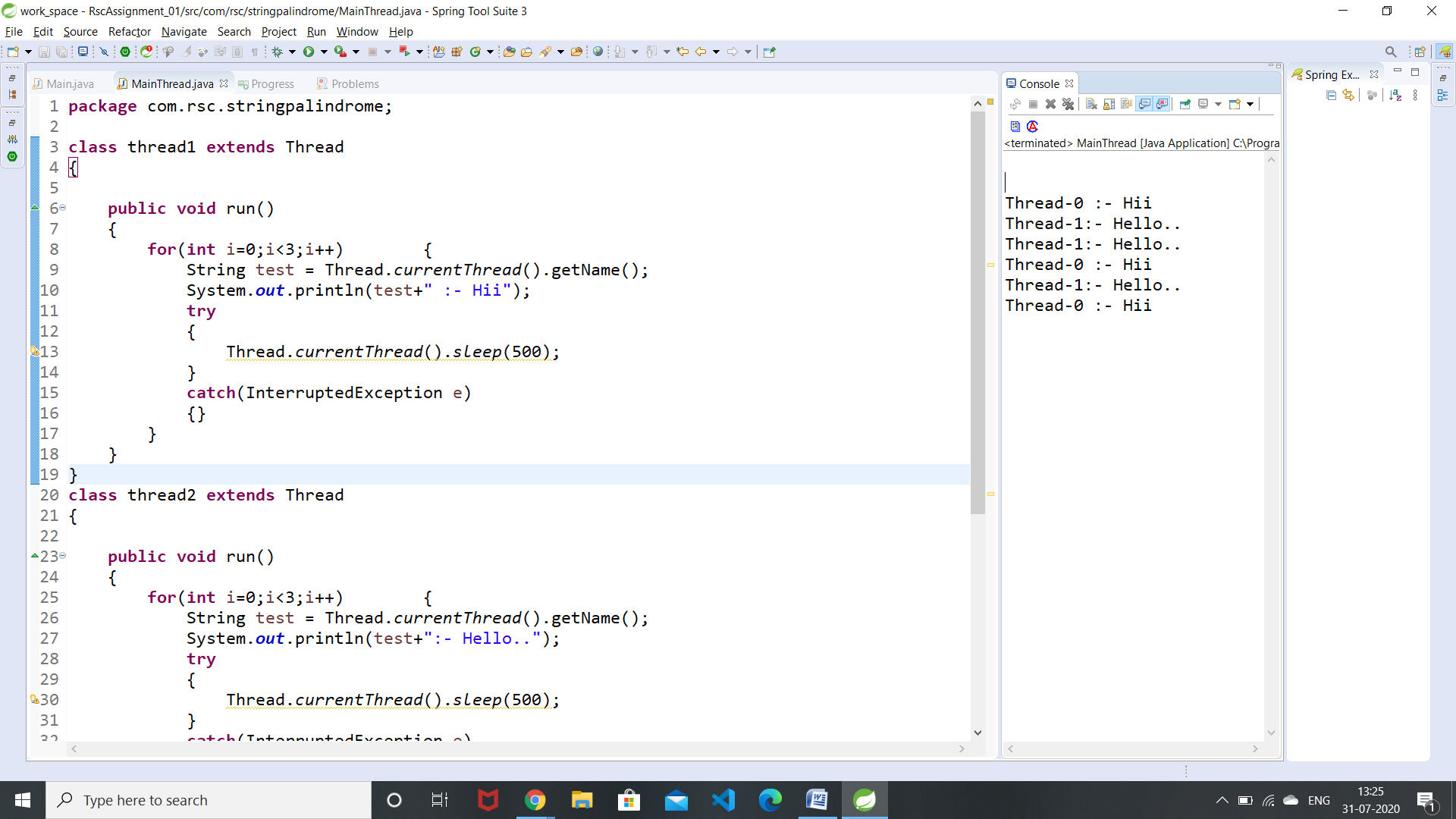
thread2 obj2=**new** thread2();

obj1.start();

obj2.start();

}

}



**3.Write a java program to implement interthread communication.**

**package** com.rsc.stringpalindrome;

**class** Customer **extends** Thread

{

**int** amount=10000;

**synchronized** **void** get(**int** amount)

{

System.***out***.println("consumer...");

**if**(**this**.amount<amount)

{

System.***out***.println("waiting for producer...");

**try**

{

wait();

}

**catch**(Exception e){}

}

**this**.amount-=amount;

System.***out***.println("consumed the item ...");

}

**synchronized** **void** put(**int** amount)

{

System.***out***.println("producer...");

**this**.amount+=amount;

System.***out***.println("produced the item... ");

notify();

}

}

**public** **class** ThreadCommunicaion {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Customer c=**new** Customer();

**new** Thread() {

**public** **void** run()

{

c.get(15000);

}

}.start();

**new** Thread(){

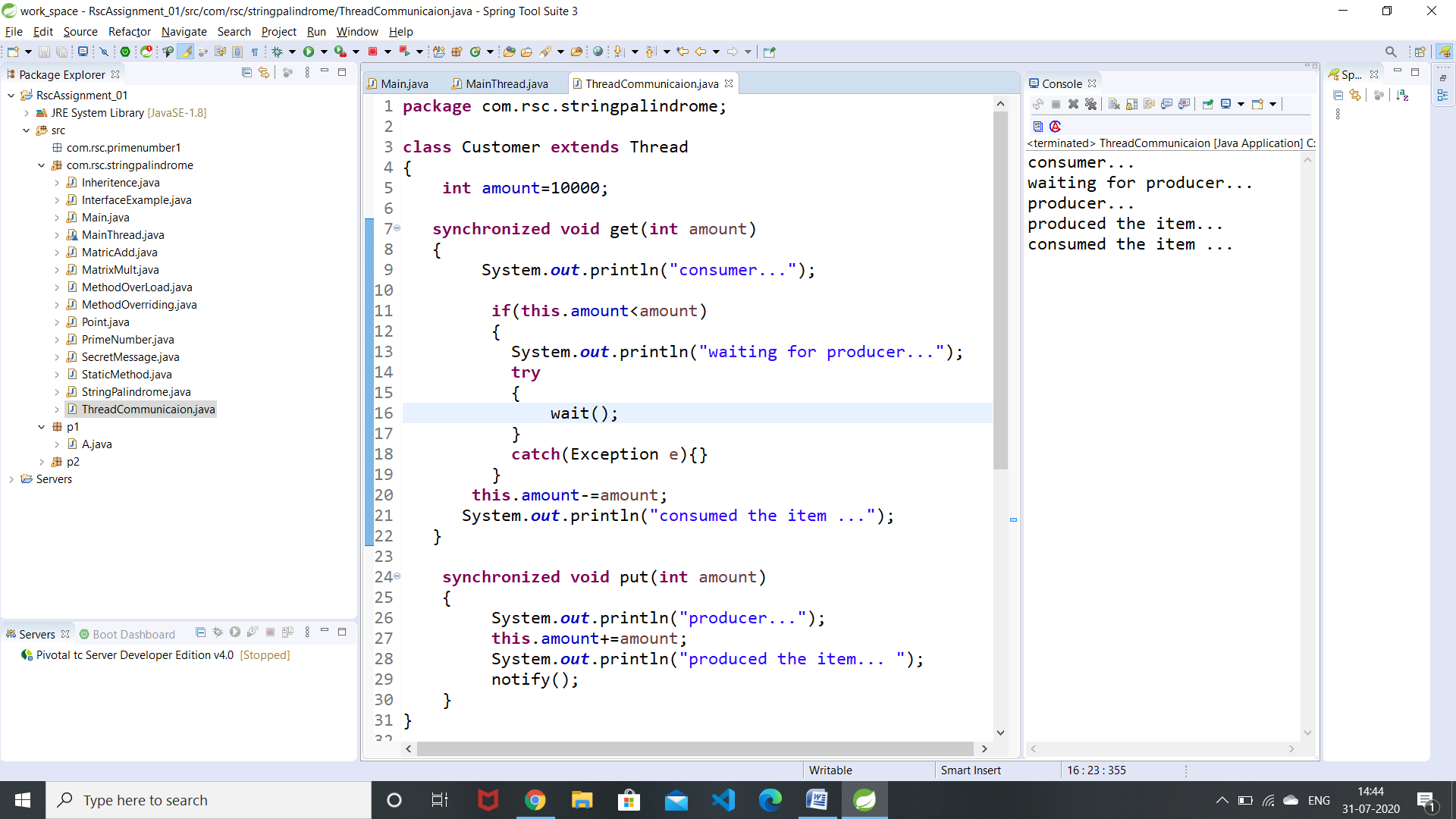
**public** **void** run()

{

c.put(10000);

}

}.start();

}

**4.Write a java program to implement Thread Synchronization.**

**package** com.rsc.stringpalindrome;

**class** Table{

**synchronized** **void** printTable(**int** n)

{

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

{

System.***out***.println(n\*i);

**try**

{

Thread.*sleep*(400);

}**catch**(Exception e)

{

System.***out***.println(e);

}

}

}

}

**public** **class** SynchThread {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Table obj = **new** Table();

Thread t1=**new** Thread()

{

**public** **void** run(){

obj.printTable(5);

}

};

Thread t2=**new** Thread()

{

**public** **void** run(){

obj.printTable(100);

}

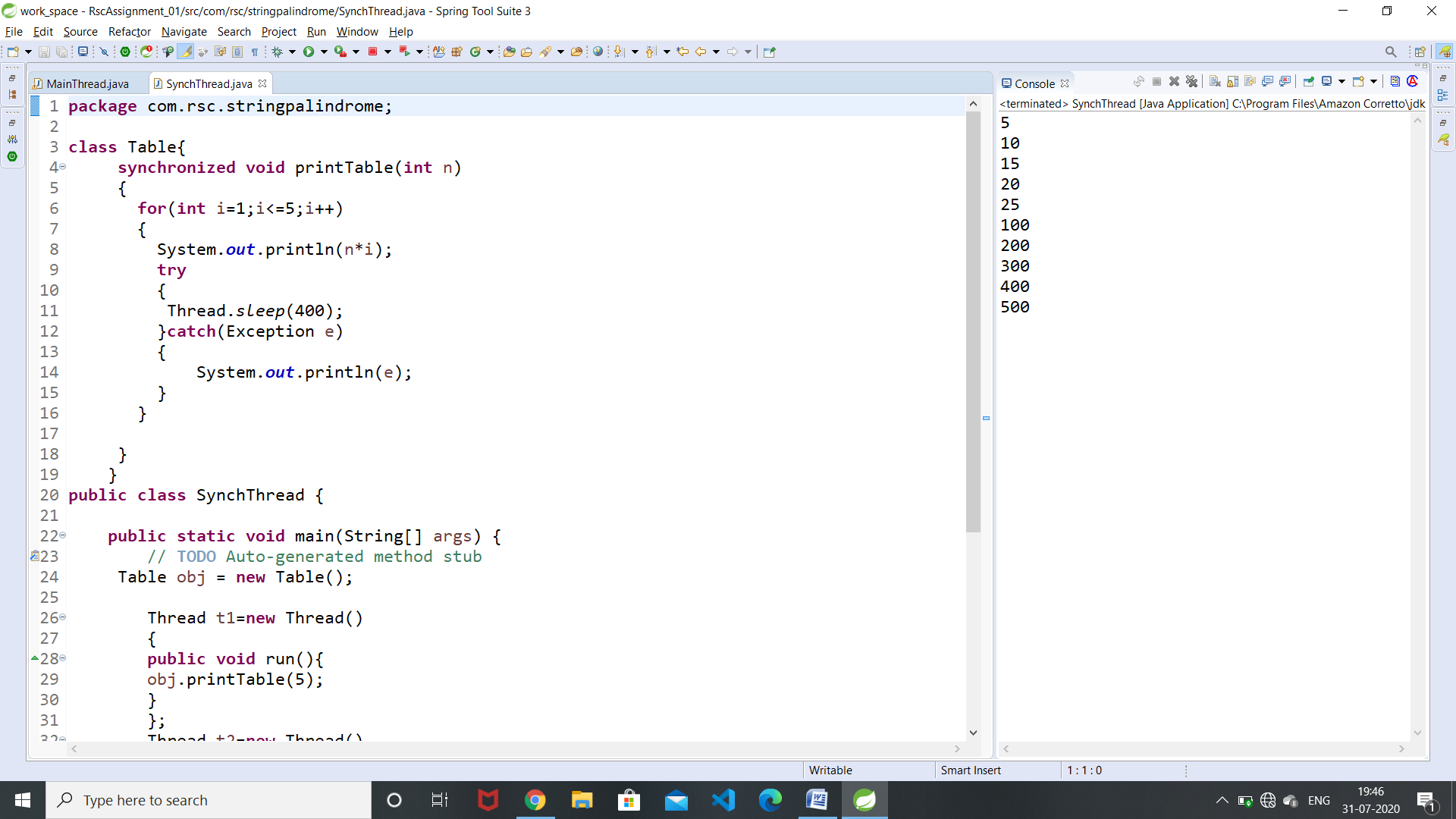
};

t1.start();

t2.start();

}

}



**5.Write a java program to implement Generic Class,Generic Method and Generic Constructor.**

**package** com.rsc.stringpalindrome;

**class** Test<Gen>

{

Gen n;

Test(Gen n)

{

**this**.n=n;

}

Gen show()

{

**return** n;

}

}

**public** **class** GenericExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Test<Integer> t1 = **new** Test<>(9852);

Test<String> t2 = **new** Test<>("NS");

**int** r1 = t1.show();

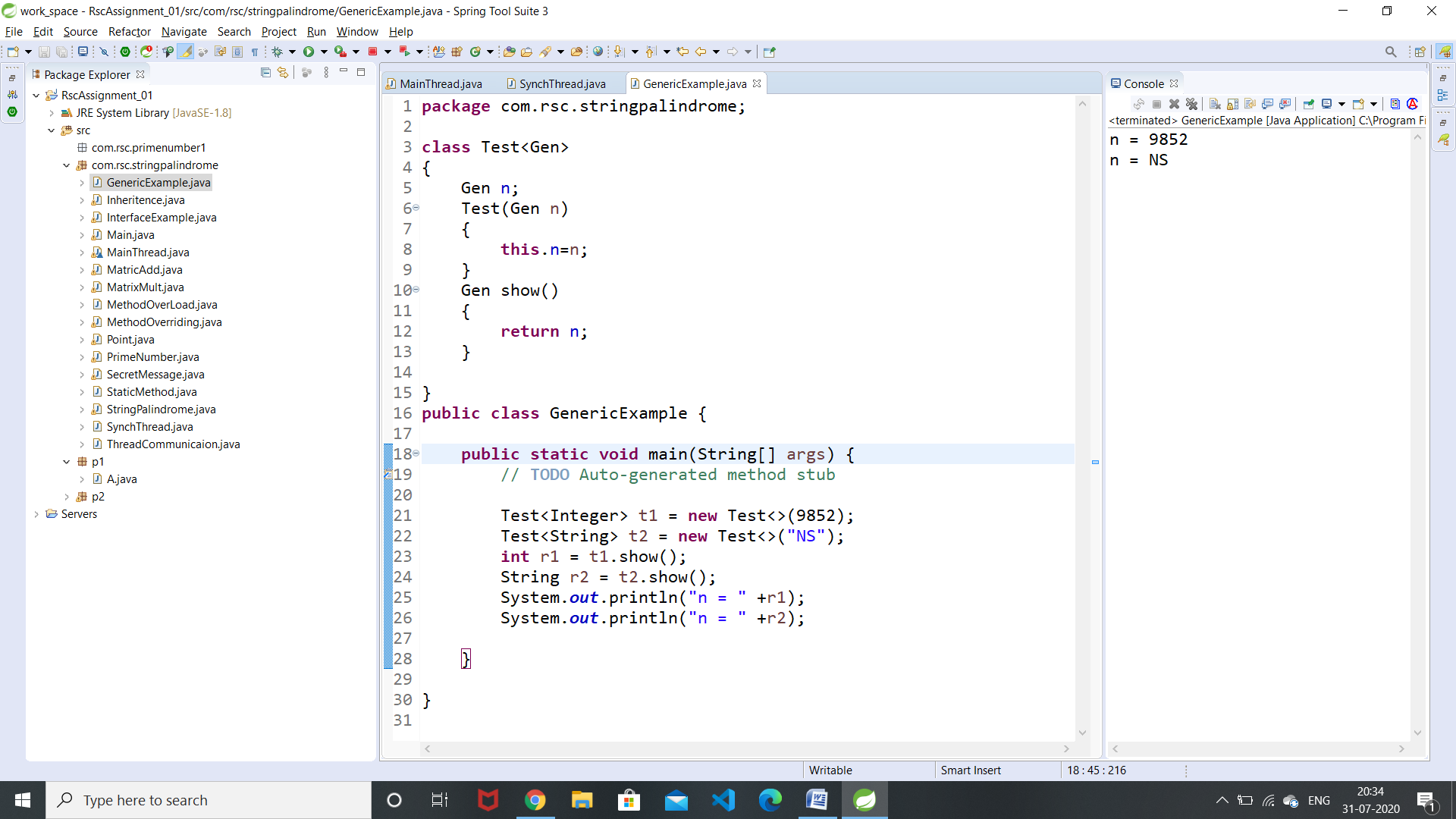
String r2 = t2.show();

System.***out***.println("n = " +r1);

System.***out***.println("n = " +r2);

}

}



**6.Write a java program to count no of vowels in a given file.**

**package** com.rsc.stringpalindrome;

**import** java.io.\*;

**public** **class** CountVowels {

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

// **TODO** Auto-generated method stub

File f1=**new** File("C:\\Users\\Nandini\\Desktop\\input.txt");

String[] words=**null**;

FileReader fr = **new** FileReader(f1);

BufferedReader br = **new** BufferedReader(fr);

String s;

**int** count=0;

**while**((s=br.readLine())!=**null**)

{

words=s.split(" ");

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

{

**for**(**int** j=0;j<words[i].length();j++)

{

**char** ch=words[i].charAt(j);

**if**(ch == 'a' || ch == 'e' || ch == 'i' ||ch == 'o' || ch == 'u')

{

count+=1;

}

}

System.***out***.println(words[i]);

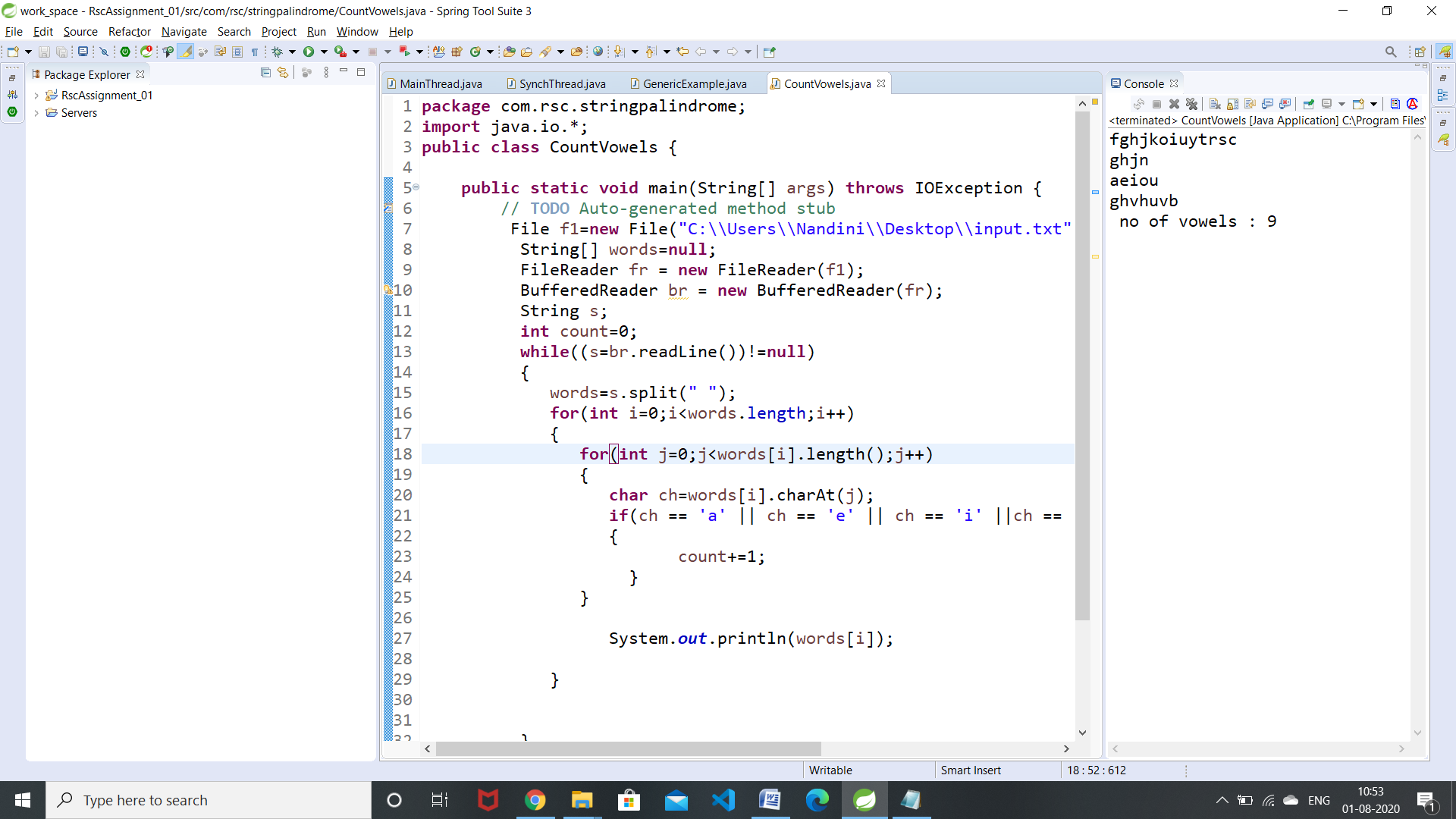
}

}

System.***out***.println("no of vowels : " +count);

}

}



**7.Write a java program to implement autoboxing and unboxing.**

**package** com.rsc.stringpalindrome;

**public** **class** Boxing {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** a = 98;

Integer i = **new** Integer(a); //AutoBoxing

Integer i1 =a;

**int** i2 = i; //unboxing

Character ch = 'a';

**char** ch1 = ch;

System.***out***.println("Value of i: " + i);

System.***out***.println("Value of i1: " + i1);

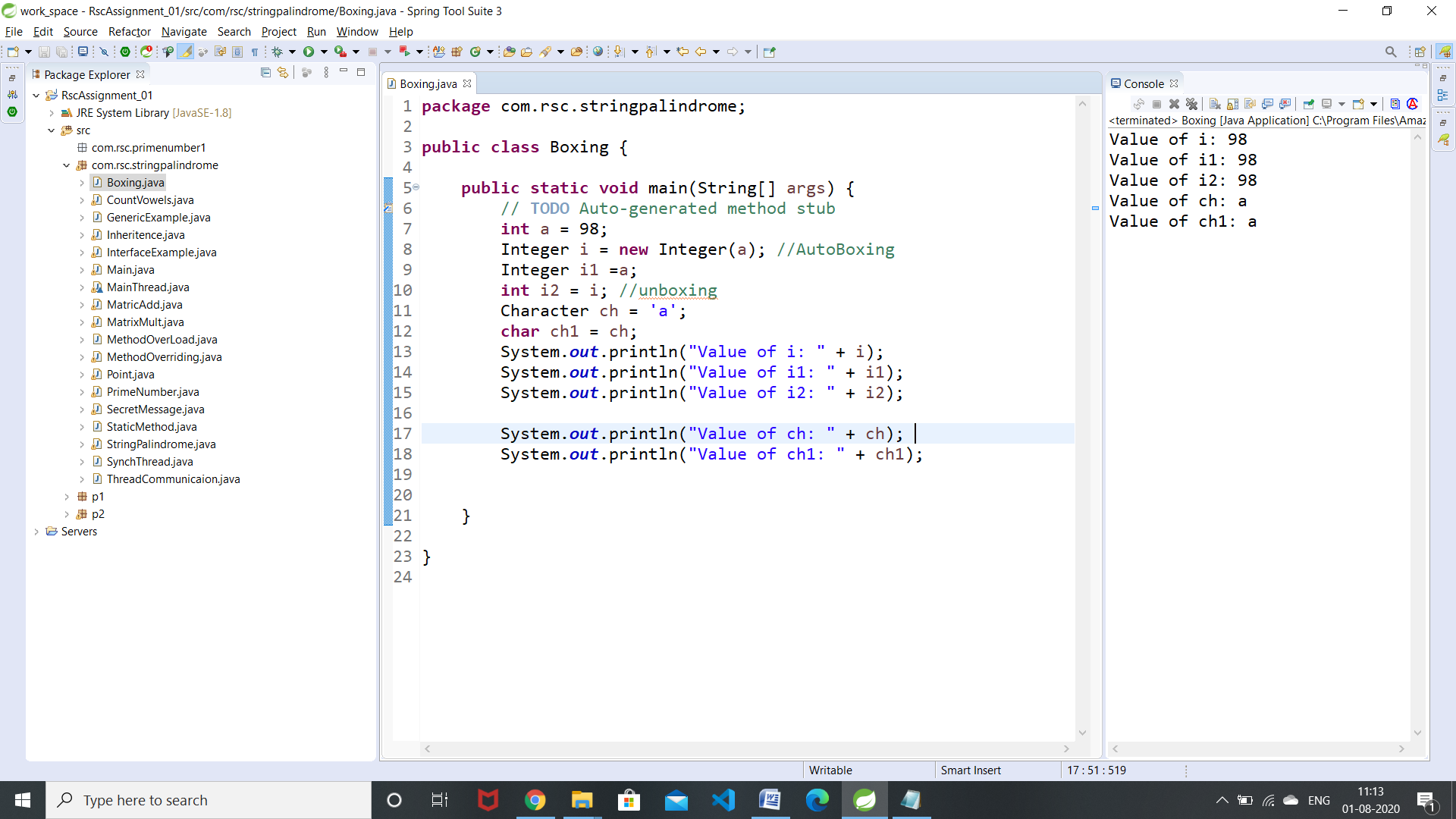
System.***out***.println("Value of i2: " + i2);

System.***out***.println("Value of ch: " + ch);

System.***out***.println("Value of ch1: " + ch1);

}

}



**8.Write a java program to copy a file.**

**package** com.rsc.stringpalindrome;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.\*;

**public** **class** CopyFile {

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

// **TODO** Auto-generated method stub

File f1=**new** File("C:\\Users\\Nandini\\Desktop\\input.txt");

FileReader fr = **new** FileReader(f1);

BufferedReader br = **new** BufferedReader(fr);

**int** s;

// FileInputStream fis = new FileInputStream("C:\\Users\\Nandini\\Desktop\\input.txt");

FileOutputStream fos = **new** FileOutputStream("C:\\Users\\Nandini\\Desktop\\newFile.txt");

**while**((s=br.read())!=-1)

{

fos.write(s);

}

}

}

**9.Write a java program to implement Stack using Generic class.**

**package** com.rsc.stringpalindrome;

**import** java.util.\*;

**public** **class** GenericStack<Gen> {

List<Gen> l = **new** ArrayList<>();

**void** push(Gen p)

{

l.add(p);

}

Gen pop()

{

**int** top=l.size();

**return** l.remove(--top);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

GenericStack<Integer> s = **new** GenericStack<>();

s.push(98);

s.push(52);

s.push(90);

**int** pop = s.pop();

System.***out***.print(pop);

}

}

**10. Write java program to swap two values using generic method.**

**package** com.rsc.stringpalindrome;

**public** **class** GenericSwap {

<Gen> **void** swap(Gen a1 , Gen a2)

{

Gen s;

s=a1;

a1=a2;

a2=s;

System.***out***.println("a1 = "+a1 + " a2 = " + a2);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

GenericSwap gs = **new** GenericSwap();

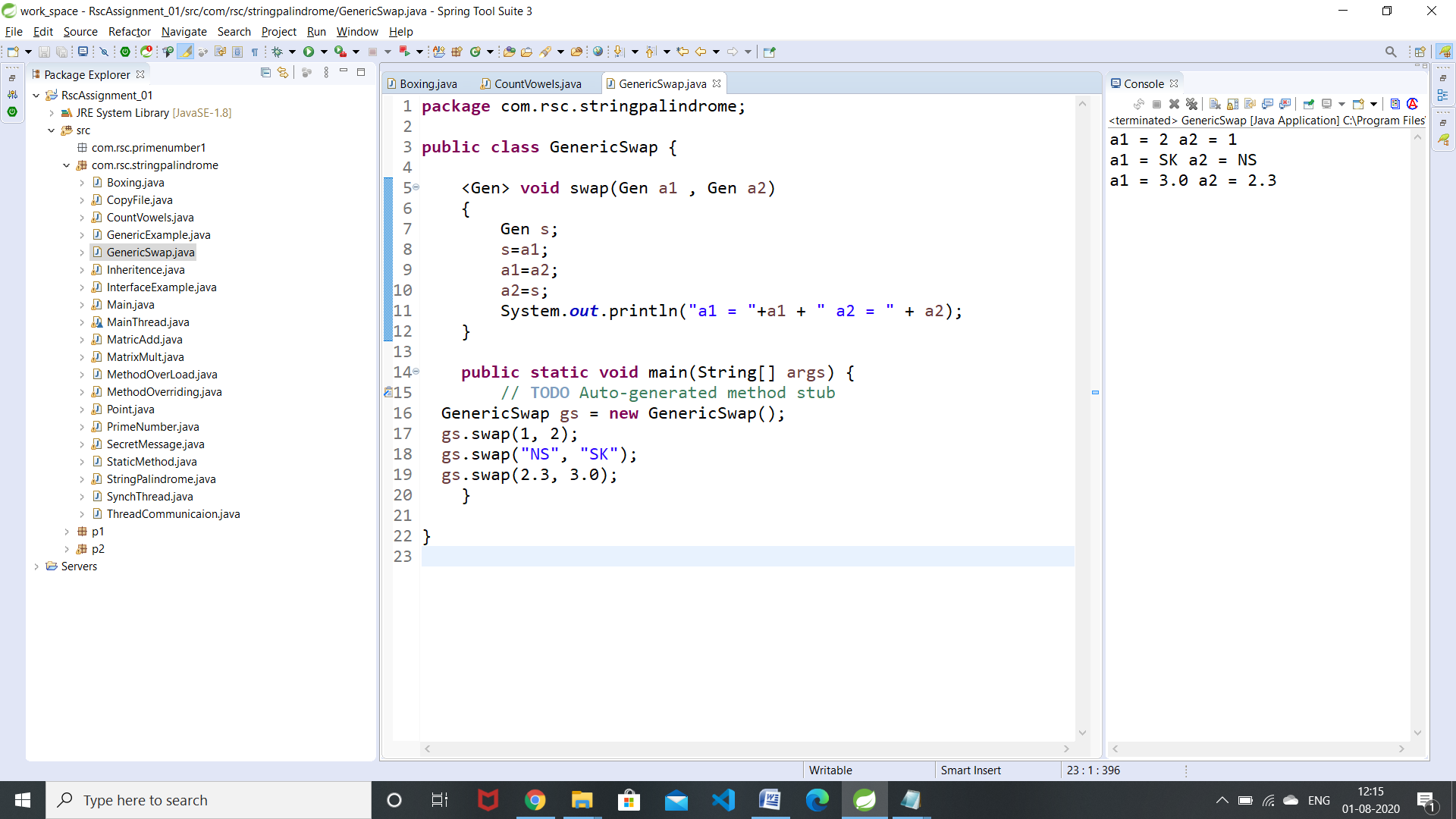
gs.swap(1, 2);

gs.swap("NS", "SK");

gs.swap(2.3, 3.0);

}

}



**1. What is thread?**

**Thread** is a light weight process which helps in running the tasks in parallel. The **threads** works independently and provides the maximum utilization of the CPU, thus enhancing the CPU performance.   Every **java** application has at least one **thread** – main **thread**. main is the first **java thread** and we can create multiple **threads** from it.

**2. Write the difference between multithreading and multitasking**

|  |  |  |
| --- | --- | --- |
|  | * **Multitasking** let CPU to execute multiple tasks at the same time. | * **Multithreading** let CPU to execute multiple threads of a process simultaneously. |
|  | * In multitasking CPU switches between programs frequently. | * In multithreading CPU switches between the threads frequently. |
|  | * In multitasking system has to allocate separate memory and resources to each program that CPU is executing. | * In multithreading system has to allocate memory to a process, multiple threads of that process shares the same memory and resources allocated to the process. |

**3. What is Enumeration?**The **Enum in Java** is a data type which contains a fixed set of constants. The **Java enum** constants are static and final implicitly. In **Java**, **enumeration** defines a class type. An **Enumeration** can have constructors, methods and instance variables. It is created using **enum** keyword. Each **enumeration** constant is public, static and final by default.

**4. What is autoboxing?   
Autoboxing**: Converting a primitive value into an object of the corresponding wrapper class is called **autoboxing**.

For example, converting int to Integer class.

**5. What is wrapper class?**

**Wrapper class** in java are the Object representation of eight primitive types in java. All the **wrapper classes** in java are immutable and **final**. While storing in data structures which support only objects, it is required to convert the primitive type to object first which we can do by using **wrapper** classes.

**6. what is transient modifier?**

**Transient** is defined as someone or something that is temporary or staying for a short amount of time. The **Java** transient keyword is used on class attributes/variables to indicate that serialization process of such class should ignore such variables while creating a persistent byte stream for any instance of that class. A transient variable is a variable that can not be serialized.

**7. What is Generic class?Write the syntax of generic class.**

A **generic type** is a **class** or interface that is parameterized over **types.**Rather than specifying obj to be of an int **type**, or a String **type**, or any other **type**, you define the Box class to accept a **type** parameter < T>.

Class BaseType<Type>

{

}

.**8. What is stream?**

The Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.

**9.What is predefined stream?**  
**Java** provides three **predefined stream** objects: in, out, and err, defined in the System class of the **java**. lang package. The out object refers to the standard output **stream** or console. The in object refers to standard input, which is the keyboard.

**10.What is multithreading?**

**Multithreading** allows the execution of multiple parts of a program at the same time. These parts are known as threads and are lightweight processes available within the process

**11. What is the use of toString()?**

A toString() is an in-built method in **Java** that returns the value given to it in string format. Hence, any object that this method is applied on, will then be returned as a string object.

**12.What is deadlock?**

A **deadlock** occurs when a process or thread enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another resource held by another waiting process.

**13. Write inter thread communication methods.**

The **wait()** method causes the current thread to **wait** until another thread invokes the **notify()** or notifyAll**()** methods for that object. The **notify()** method wakes up a single thread that is **waiting** on that object's monitor.

**14. Write the difference between Checked and Unchecked exception.**

The main difference between **checked** and **unchecked exception** is that the **checked exceptions** are **checked** at compile-time while **unchecked exceptions** are **checked** at runtime.

A **checked exception** is a type of **exception** that must be either caught or declared in the method in which it is thrown. For example, the java.io.IOException is a **checked exception**.

**Unchecked Exceptions** are subclasses of RuntimeException. **Example** of **unchecked exceptions** are : ArithmeticException , ArrayStoreException , ClassCastException and so on.

**15. What is thread synchronization?**

**Synchronization** in **java** is the capability to control the access of multiple **threads** to any shared resource. **Java Synchronization** is better option where we want to allow only one **thread** to access the shared resource.  
...  
**Thread Synchronization**

1. **Synchronized** method.
2. **Synchronized** block.
3. static **synchronization**.