DAY 11

File Handling

File handling in Java refers to the process of reading from and writing to files on a computer's file system. Java provides various classes and methods to perform file operations, such as creating, reading, writing, and deleting files.

Key Concepts:

- File Input/Output Streams: Java uses streams to read from and write to files. Input streams are used to read data from a file, while output streams are used to write data to a file.

- File Classes: Java provides several classes for file handling, including File, FileInputStream, FileOutputStream, FileReader, and FileWriter.

Common File Operations:

- Creating a File: You can create a new file using the File class and the createNewFile() method.

- Reading from a File: You can read from a file using FileInputStream, FileReader, or other reader classes.

- Writing to a File: You can write to a file using FileOutputStream, FileWriter, or other writer classes.

- Deleting a File: You can delete a file using the File class and the delete() method.

Java File Handling Classes:

- \*File Class:\* Represents a file or directory and provides methods for file operations.

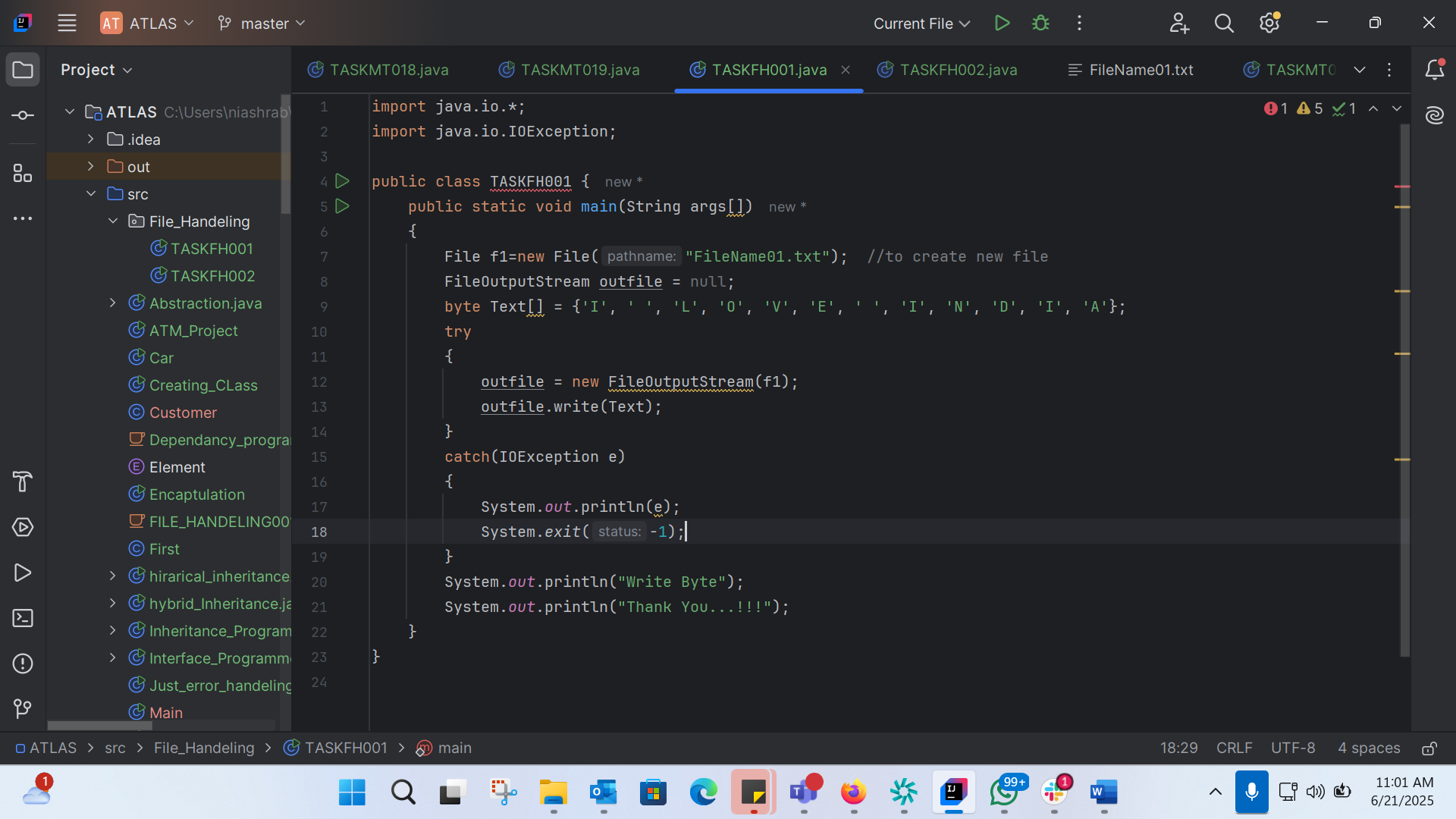
- \*FileInputStream Class:\* Used to read data from a file.

- \*FileOutputStream Class:\* Used to write data to a file.

- \*FileReader Class:\* Used to read characters from a file.

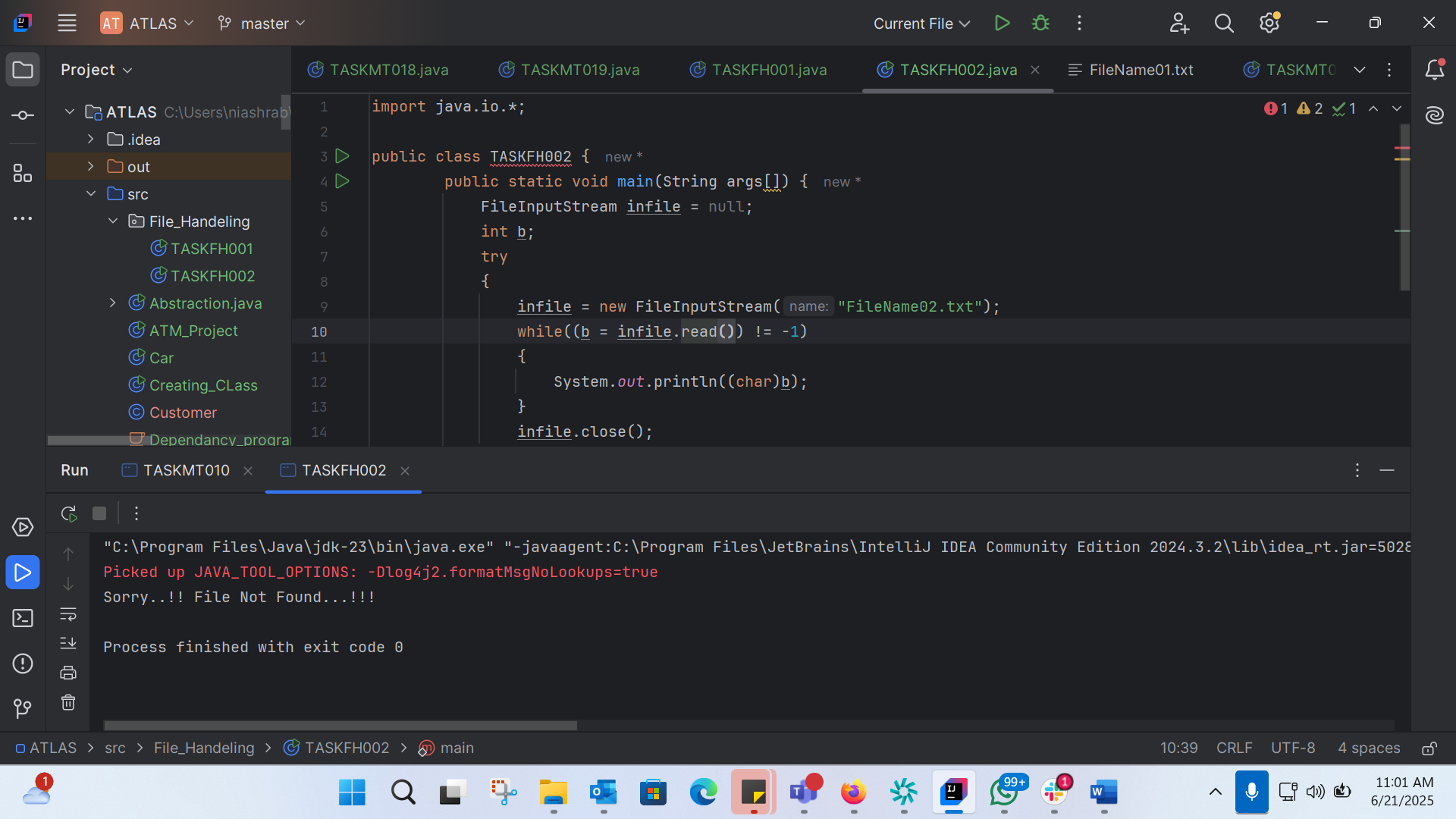
- \*FileWriter Class:\* Used to write characters to a file.

TASK 001:



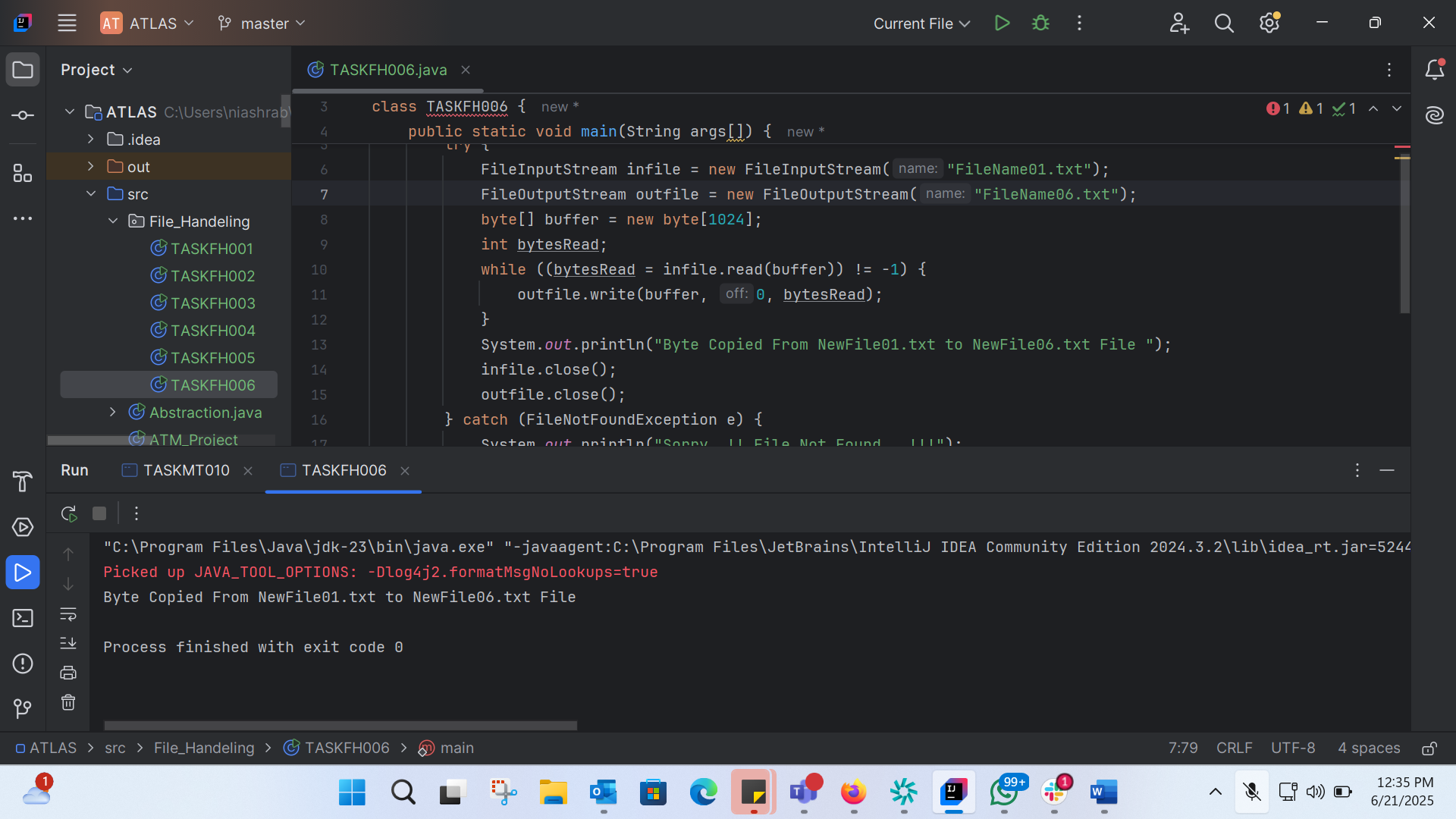
import java.io.\*;  
import java.io.IOException;  
  
public class TASKFH001 {  
 public static void main(String args[])  
 {  
 File f1=new File("FileName01.txt"); //to create new file  
 FileOutputStream outfile = null;  
 byte Text[] = {'I', ' ', 'L', 'O', 'V', 'E', ' ', 'I', 'N', 'D', 'I', 'A'};  
 try  
 {  
 outfile = new FileOutputStream(f1);  
 outfile.write(Text);  
 }  
 catch(IOException e)  
 {  
 System.*out*.println(e);  
 System.*exit*(-1);  
 }  
 System.*out*.println("Write Byte");  
 System.*out*.println("Thank You...!!!");  
 }  
}

TASK002:



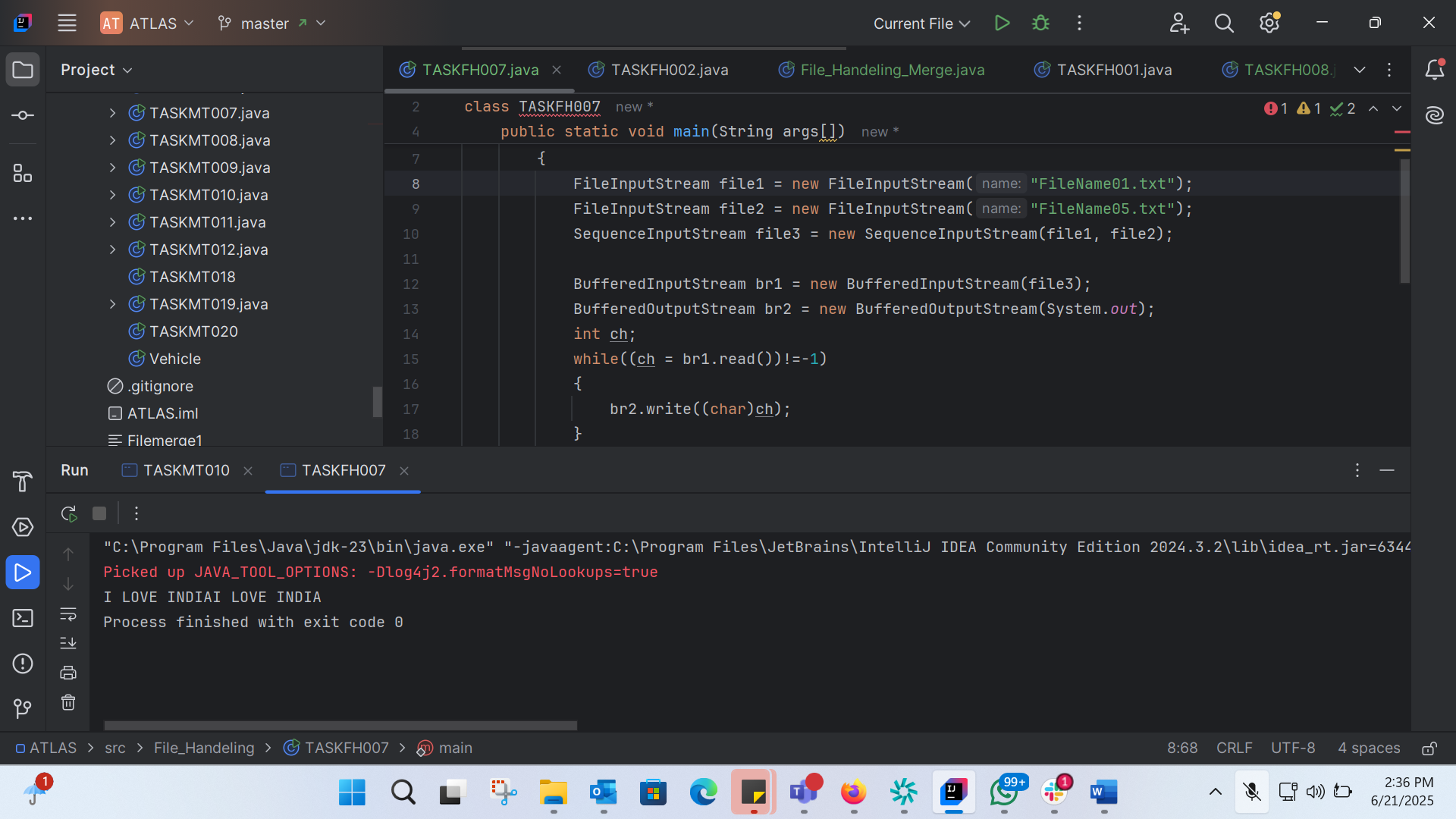
import java.io.\*;  
  
public class TASKFH002 {  
 public static void main(String args[]) {  
 FileInputStream infile = null;  
 int b;  
 try  
 {  
 infile = new FileInputStream("FileName02.txt");  
 while((b = infile.read()) != -1)  
 {  
 System.*out*.println((char)b);  
 }  
 infile.close();  
 }  
 catch(IOException e)  
 {  
 System.*out*.println("Sorry..!! File Not Found...!!!");  
 }  
 }  
}

TASK 6:



import java.io.\*;  
  
class TASKFH006 {  
 public static void main(String args[]) {  
 try {  
 FileInputStream infile = new FileInputStream("FileName01.txt");  
 FileOutputStream outfile = new FileOutputStream("FileName06.txt");  
 byte[] buffer = new byte[1024];  
 int bytesRead;  
 while ((bytesRead = infile.read(buffer)) != -1) {  
 outfile.write(buffer, 0, bytesRead);  
 }  
 System.*out*.println("Byte Copied From NewFile01.txt to NewFile06.txt File ");  
 infile.close();  
 outfile.close();  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Sorry..!! File Not Found...!!!");  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
}

Task7:



import java.io.\*;  
class TASKFH007  
{  
 public static void main(String args[])  
 {  
 try  
 {  
 FileInputStream file1 = new FileInputStream("FileName01.txt");  
 FileInputStream file2 = new FileInputStream("FileName05.txt");  
 SequenceInputStream file3 = new SequenceInputStream(file1, file2);  
  
 BufferedInputStream br1 = new BufferedInputStream(file3);  
 BufferedOutputStream br2 = new BufferedOutputStream(System.*out*);  
 int ch;  
 while((ch = br1.read())!=-1)  
 {  
 br2.write((char)ch);  
 }  
 br1.close();  
 br2.close();  
 file1.close();  
 file2.close();  
 System.*out*.println("Merge Two File Sucessfully ");  
 }  
 catch(IOException e)  
 {  
 System.*out*.println("Sorry..!! File Not Found...!!!");  
 }  
 }  
}

Task 8:

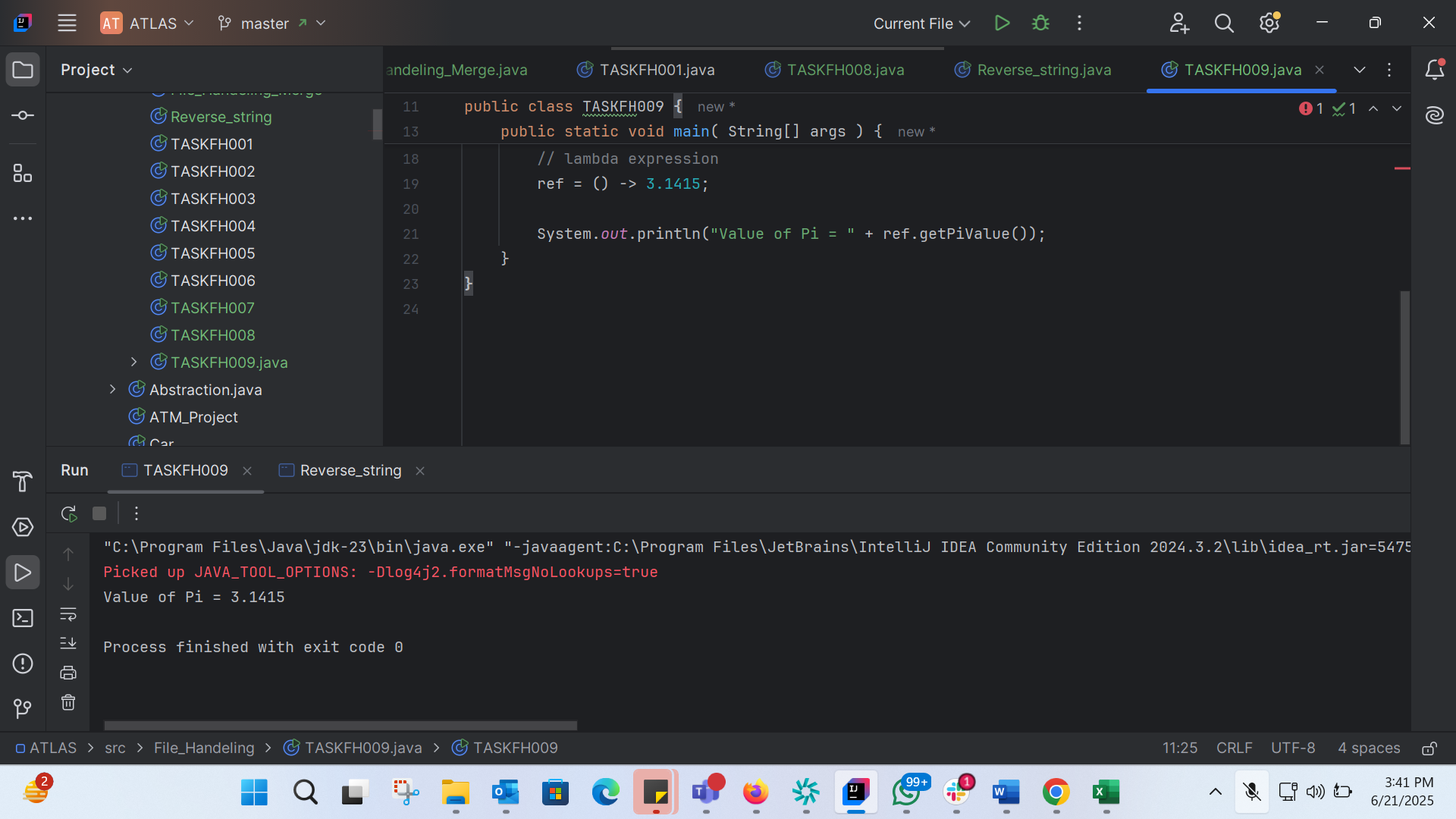


import java.io.\*;  
public class TASKFH008 {  
 public static void main(String[] args) {  
 File f1 = new File(args[0]);  
 File f2 = new File(args[1]);  
 f1.renameTo(f2);  
 System.*out*.println("Renamed file "+f1+" to "+f2);  
 }  
}

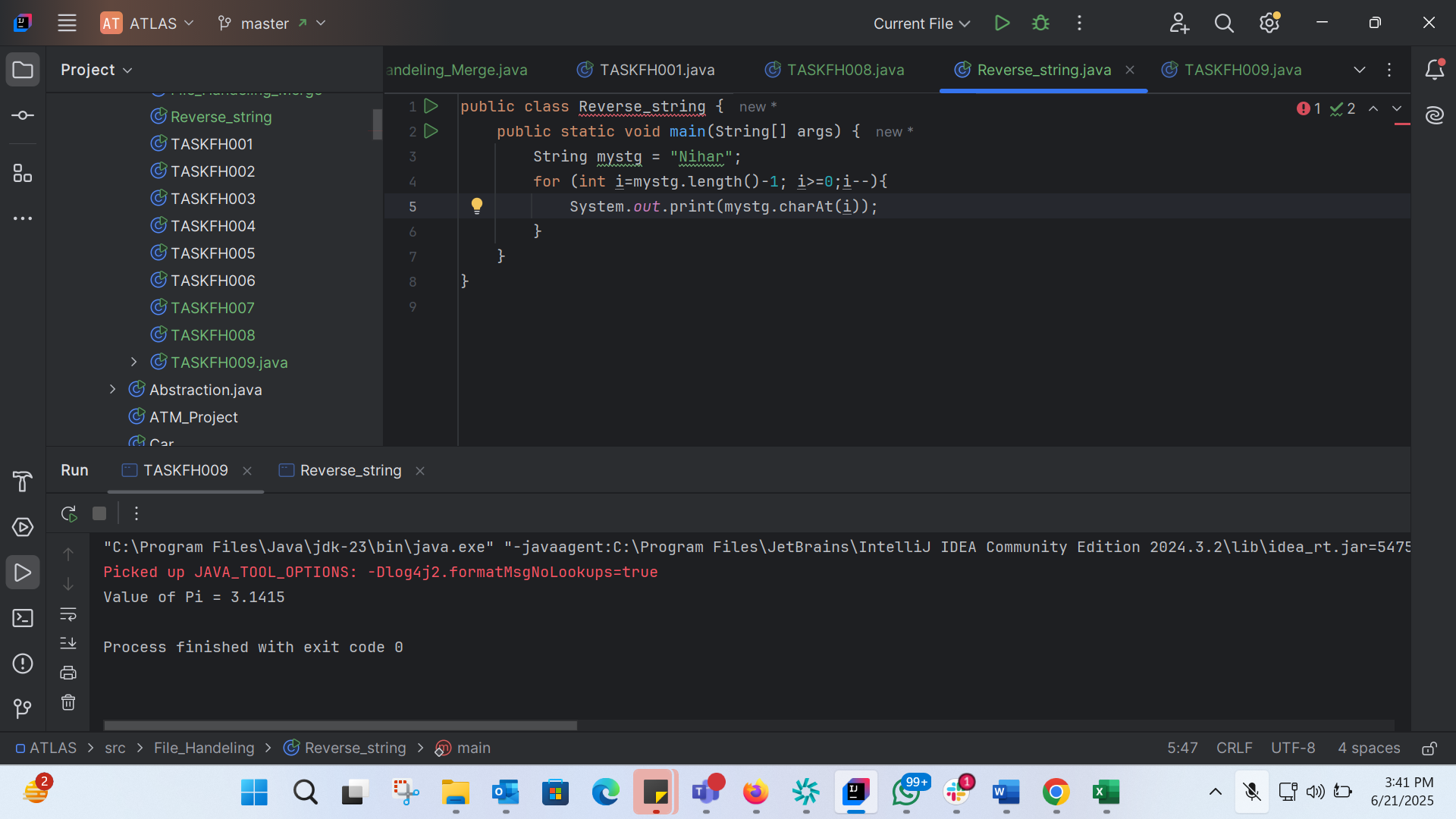
BIOLER PLATE CODE:

Boilerplate code refers to a section of code that is repeated in multiple places with little to no variation. It is often used to describe code that is necessary for a particular task or functionality, but does not provide any unique or interesting logic.

TASK 9:



import java.lang.FunctionalInterface;  
  
// this is functional interface  
@FunctionalInterface  
interface MyInterface{  
  
 // abstract method  
 double getPiValue();  
}  
  
public class TASKFH009 {  
  
 public static void main( String[] args ) {  
  
 // declare a reference to MyInterface  
 MyInterface ref;  
  
 // lambda expression  
 ref = () -> 3.1415;  
  
 System.*out*.println("Value of Pi = " + ref.getPiValue());  
 }  
}

TASK 10:

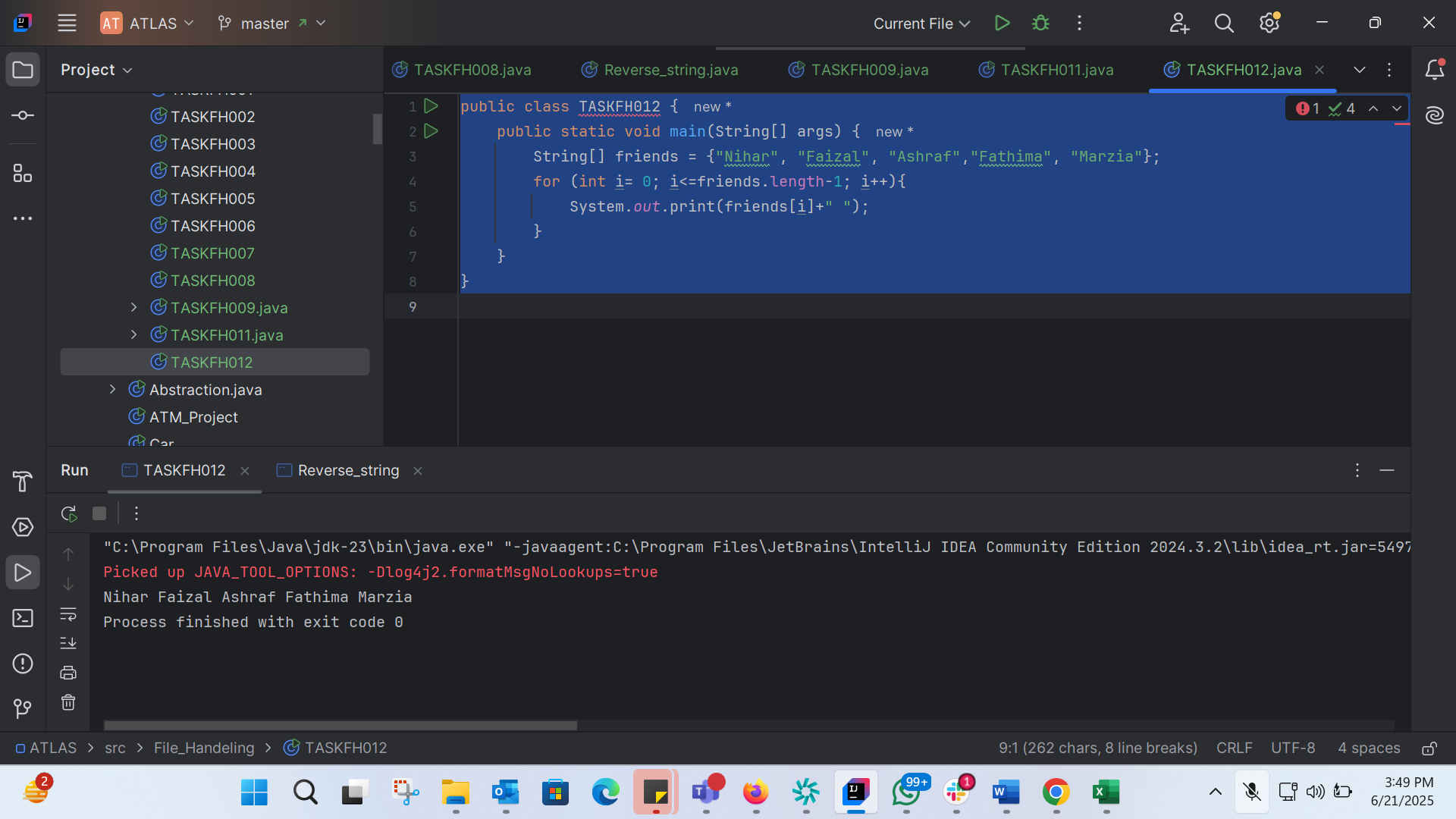
public class Reverse\_string {  
 public static void main(String[] args) {  
 String mystg = "Nihar";  
 for (int i=mystg.length()-1; i>=0;i--){  
 System.*out*.print(mystg.charAt(i));  
 }  
 }  
}

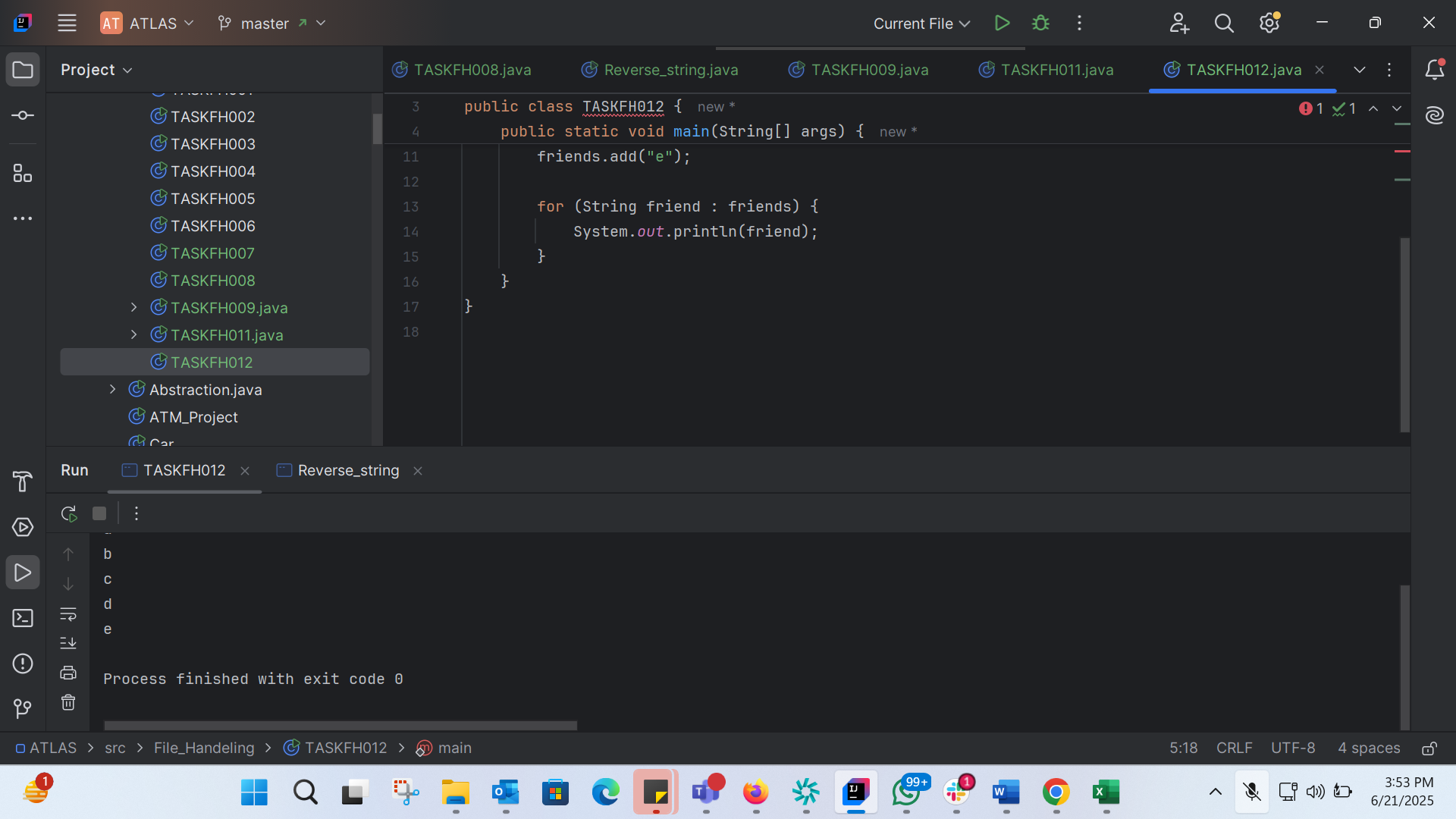
TASK 11:

@FunctionalInterface  
interface Myinterface {  
  
 // abstract method  
 String reverse(String n);  
}  
  
public class TASKFH011 {  
  
 public static void main( String[] args ) {  
  
 // declare a reference to MyInterface  
 // assign a lambda expression to the reference  
 Myinterface ref = (str) -> {  
  
 String result = "";  
 for (int i = str.length()-1; i >= 0 ; i--)  
 result += str.charAt(i);  
 return result;  
 };  
  
 // call the method of the interface  
 System.*out*.println("Lambda reversed = " + ref.reverse("Lambda"));  
 }  
  
}

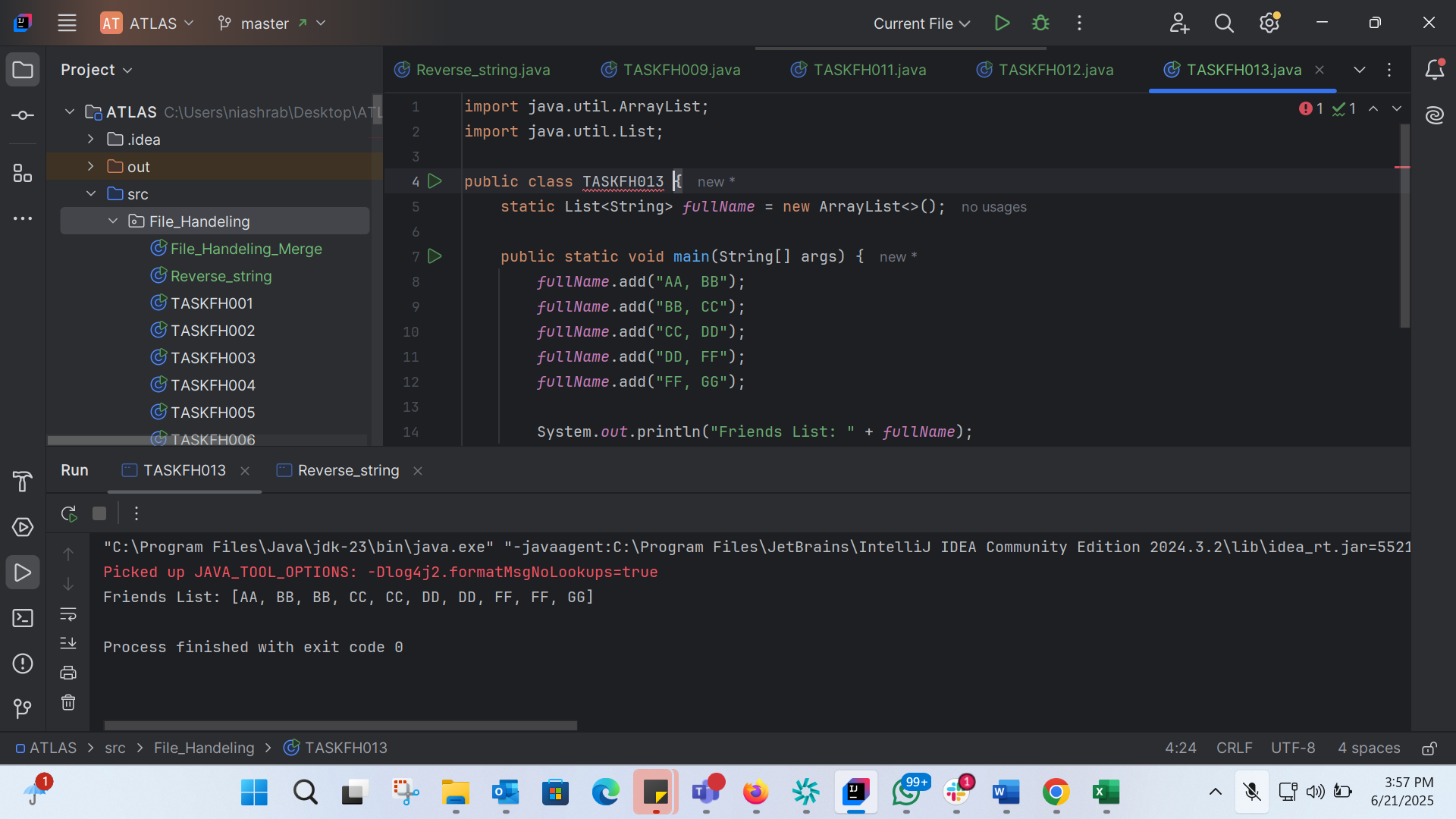
Task 12:

public class TASKFH012 {  
 public static void main(String[] args) {  
 String[] friends = {"Nihar", "Faizal", "Ashraf","Fathima", "Marzia"};  
 for (int i= 0; i<=friends.length-1; i++){  
 System.*out*.print(friends[i]+" ");  
 }  
 }  
}

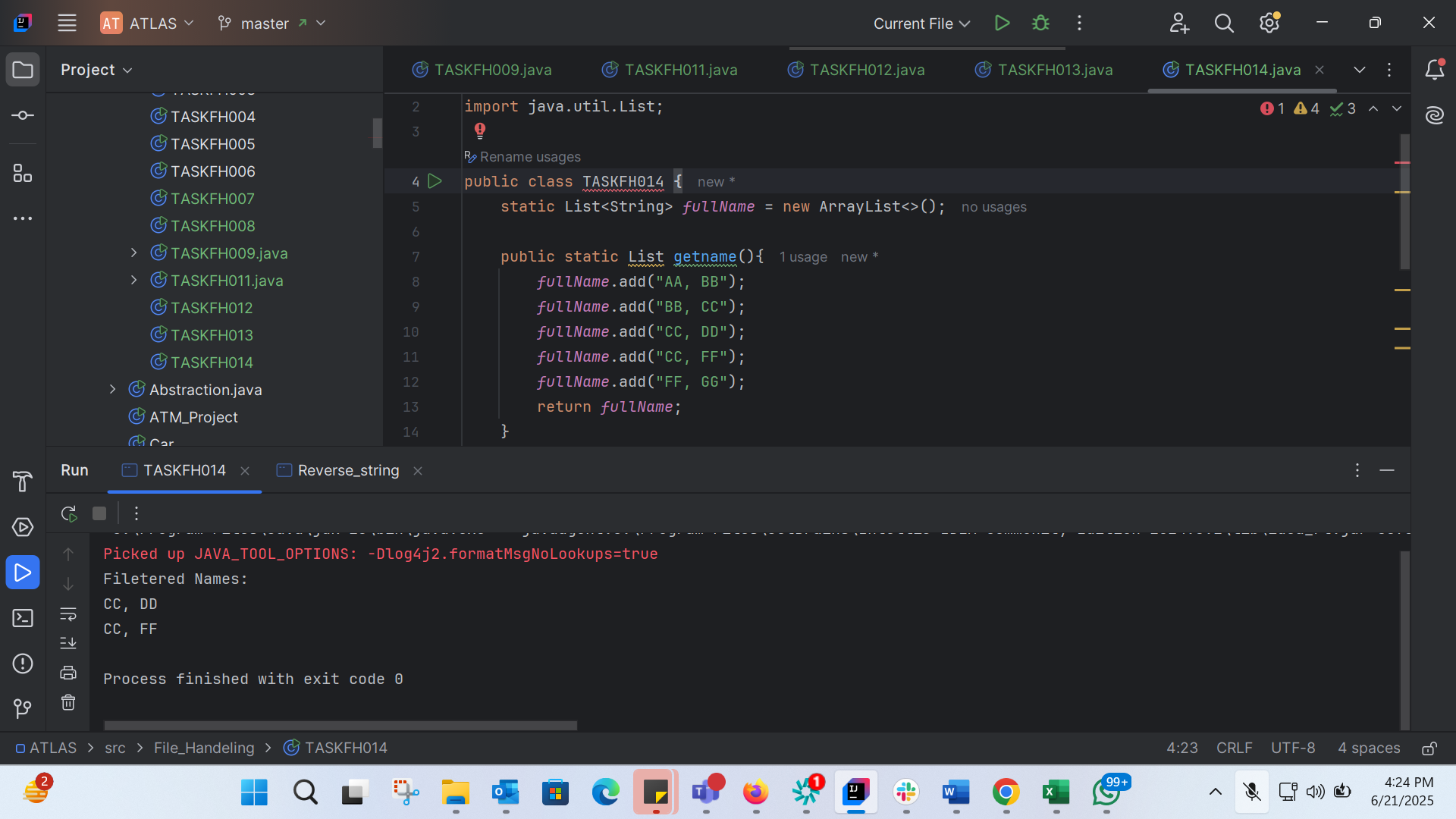




import java.util.ArrayList;  
  
public class TASKFH012 {  
 public static void main(String[] args) {  
 ArrayList<String> friends = new ArrayList<>();  
  
 friends.add("a");  
 friends.add("b");  
 friends.add("c");  
 friends.add("d");  
 friends.add("e");  
  
 for (String friend : friends) {  
 System.*out*.println(friend);  
 }  
 }  
}

Task 13:

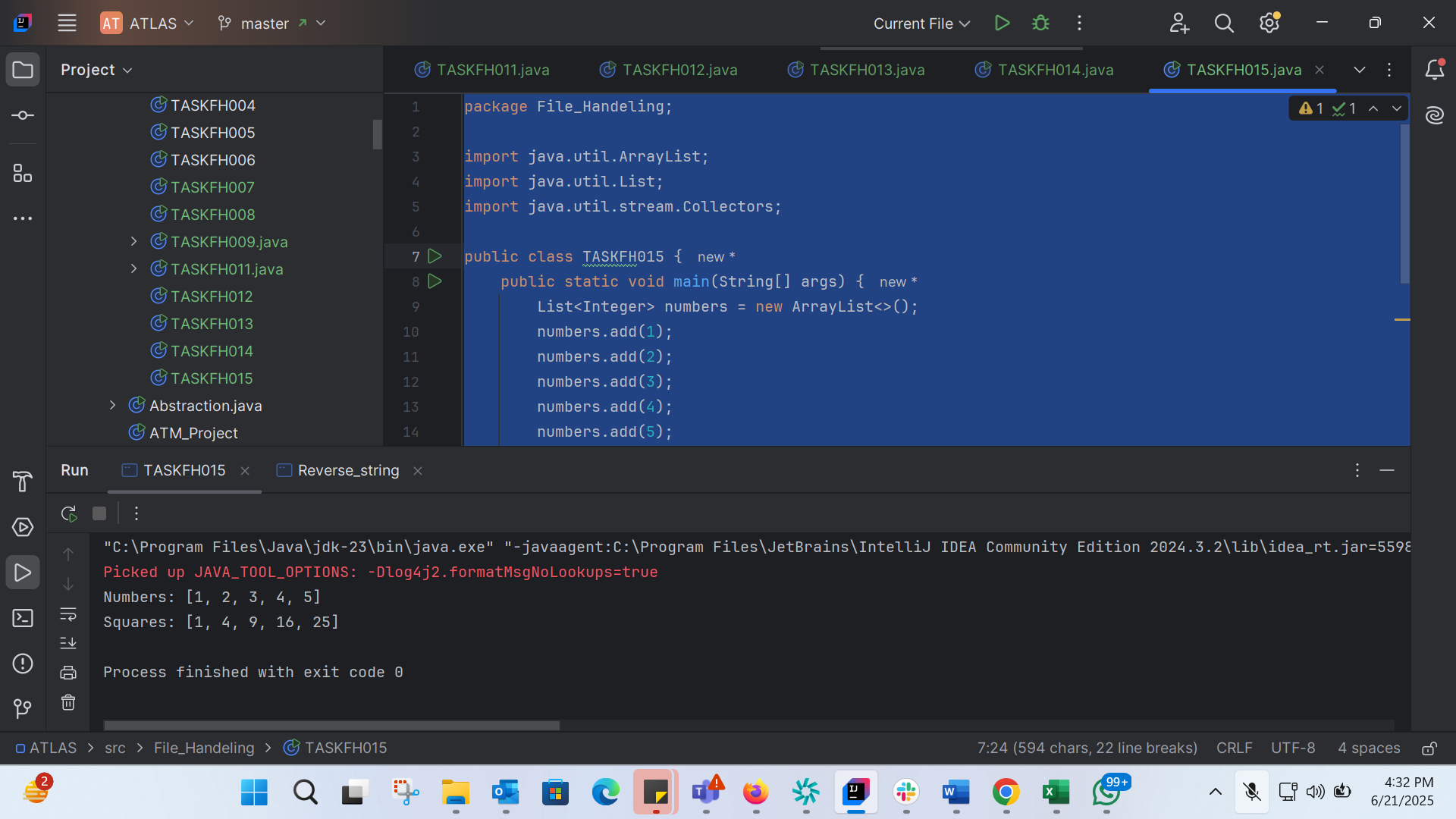
import java.util.ArrayList;  
import java.util.List;  
  
public class TASKFH013 {  
 static List<String> *fullName* = new ArrayList<>();  
  
 public static void main(String[] args) {  
 *fullName*.add("AA, BB");  
 *fullName*.add("BB, CC");  
 *fullName*.add("CC, DD");  
 *fullName*.add("DD, FF");  
 *fullName*.add("FF, GG");  
  
 System.*out*.println("Friends List: " + *fullName*);  
 }  
}

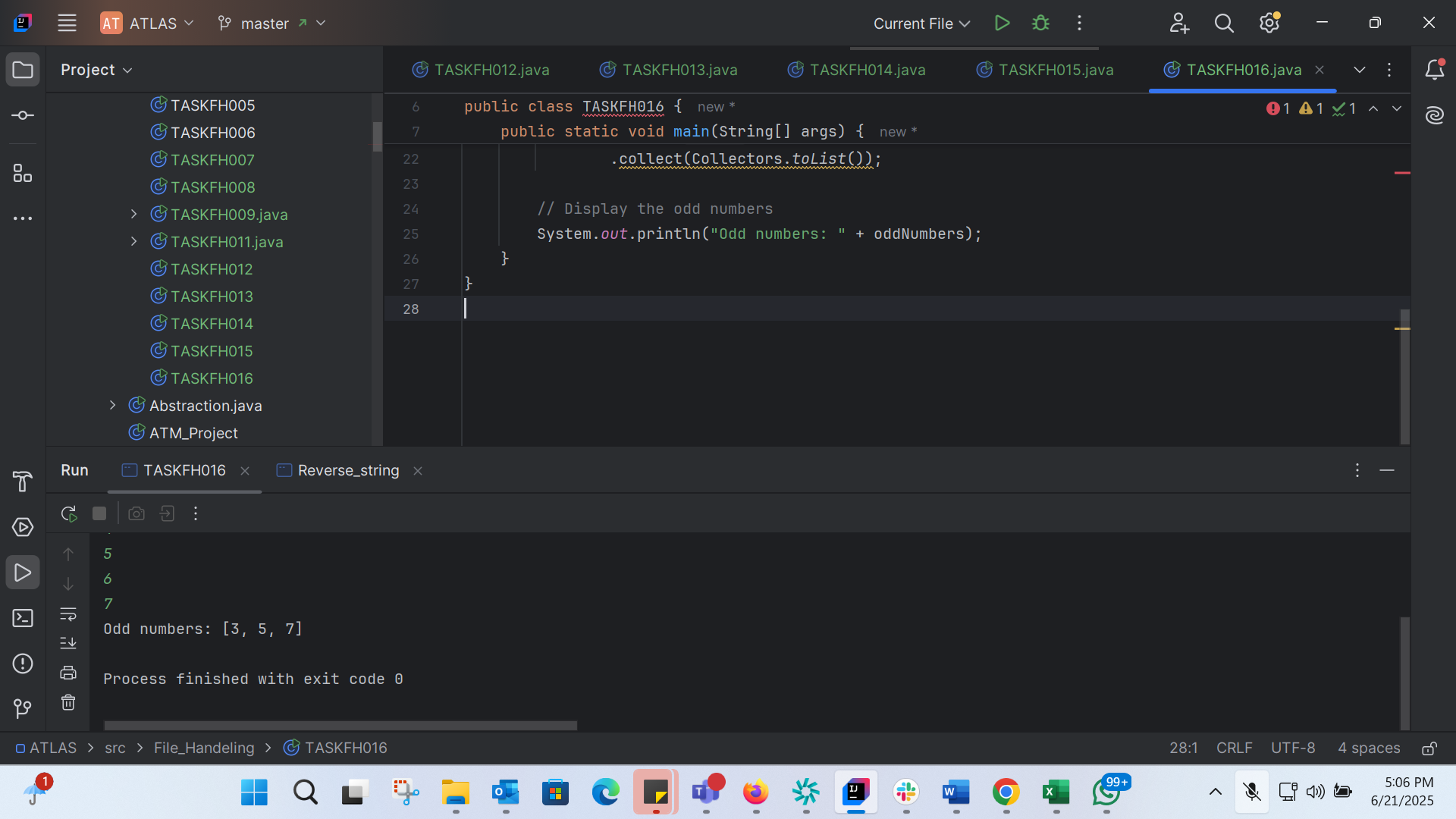
TASK 14:

import java.util.ArrayList;  
import java.util.List;  
  
public class TASKFH014 {  
 static List<String> *fullName* = new ArrayList<>();  
  
 public static List getname(){  
 *fullName*.add("AA, BB");  
 *fullName*.add("BB, CC");  
 *fullName*.add("CC, DD");  
 *fullName*.add("CC, FF");  
 *fullName*.add("FF, GG");  
 return *fullName*;  
 }  
  
 public static void main(String[] args) {  
 List<String> myNames = *getname*();  
 System.*out*.println("Filetered Names:");  
 myNames.stream()  
 .filter((p)->p.startsWith("CC"))  
 .map((p)->p.toUpperCase())  
 .sorted()  
 .forEach((p)-> System.*out*.println(p));  
  
 }  
}

TASK 15:

package File\_Handeling;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class TASKFH015 {  
 public static void main(String[] args) {  
 List<Integer> numbers = new ArrayList<>();  
 numbers.add(1);  
 numbers.add(2);  
 numbers.add(3);  
 numbers.add(4);  
 numbers.add(5);  
  
 List<Integer> squares = numbers.stream()  
 .map(n -> n \* n)  
 .collect(Collectors.*toList*());  
  
 System.*out*.println("Numbers: " + numbers);  
 System.*out*.println("Squares: " + squares);  
 }  
}

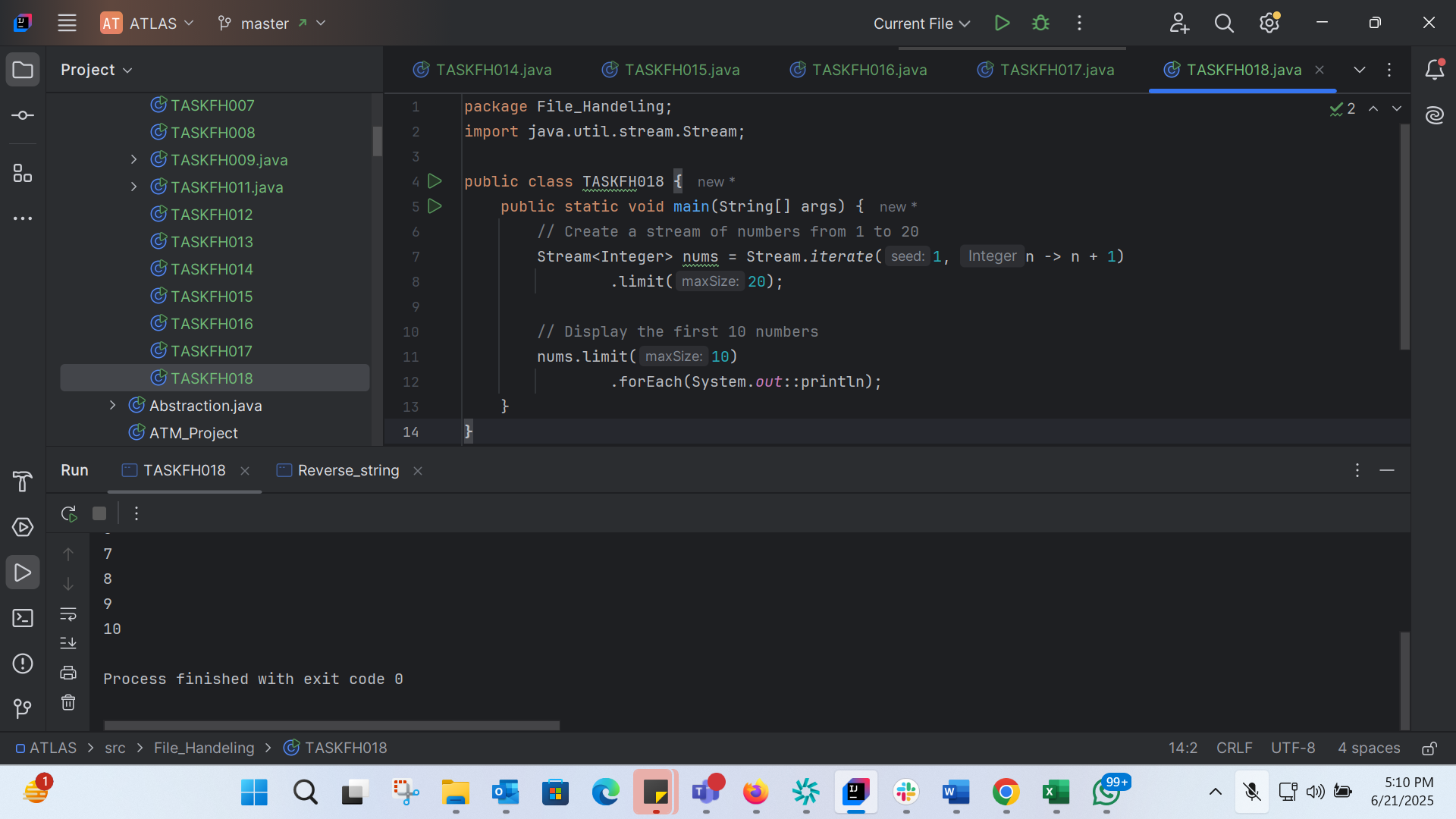


TASK 16:

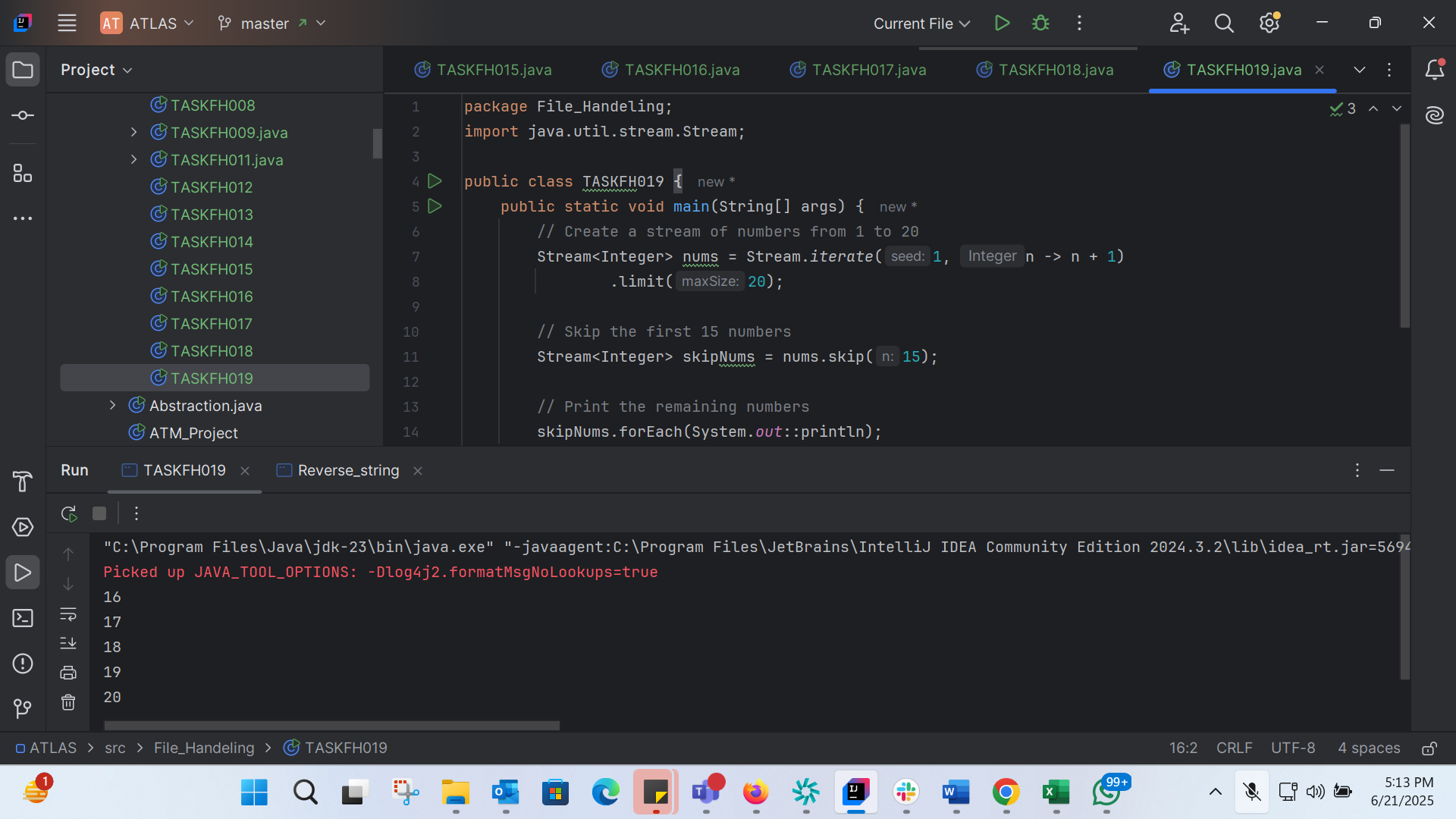
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
import java.util.stream.Collectors;  
  
public class TASKFH016 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 List<Integer> numbers = new ArrayList<>();  
  
 System.*out*.print("Enter the number of elements: ");  
 int n = scanner.nextInt();  
  
 System.*out*.println("Enter " + n + " numbers:");  
 for (int i = 0; i < n; i++) {  
 numbers.add(scanner.nextInt());  
 }  
  
 // Filter out the odd numbers  
 List<Integer> oddNumbers = numbers.stream()  
 .filter(num -> num % 2 != 0)  
 .collect(Collectors.*toList*());  
  
 // Display the odd numbers  
 System.*out*.println("Odd numbers: " + oddNumbers);  
 }  
}

TASK 17:

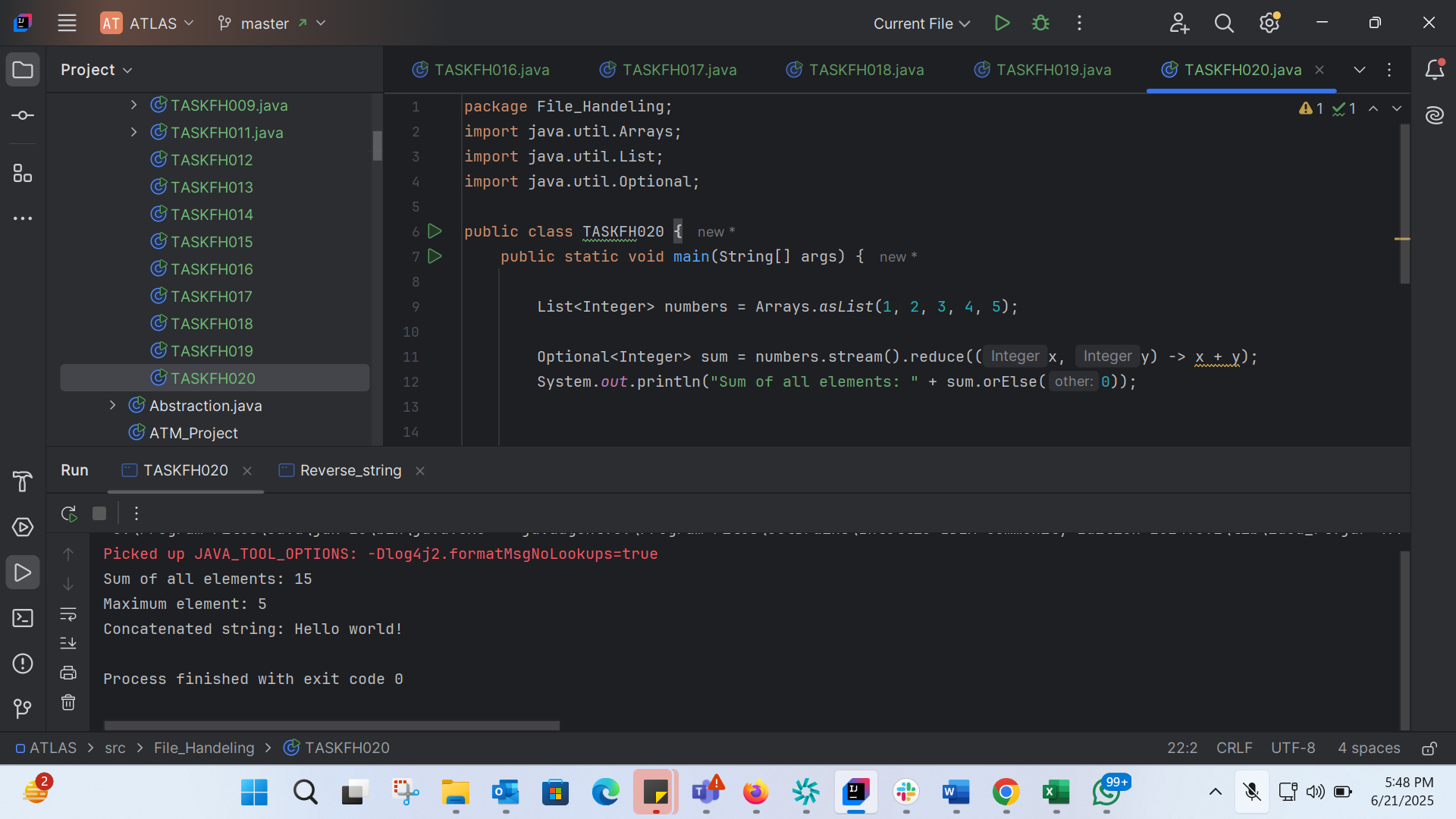
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class TASKFH017 {  
 public static void main(String[] args) {  
 // Create an ArrayList of integers  
 List<Integer> numbers = new ArrayList<>();  
 numbers.add(12);  
 numbers.add(44);  
 numbers.add(12);  
 numbers.add(66);  
 numbers.add(44);  
 numbers.add(77);  
 numbers.add(12);  
 numbers.add(66);  
  
 // Display the original list  
 System.*out*.println("Original list: " + numbers);  
  
 // Remove duplicates from the list  
 List<Integer> removeDups = numbers.stream()  
 .distinct()  
 .collect(Collectors.*toList*());  
  
 // Display the list after removing duplicates  
 System.*out*.println("List after removing duplicates: " + removeDups);  
 }  
}

TASK 18: 

package File\_Handeling;  
import java.util.stream.Stream;  
  
public class TASKFH018 {  
 public static void main(String[] args) {  
 // Create a stream of numbers from 1 to 20  
 Stream<Integer> nums = Stream.*iterate*(1, n -> n + 1)  
 .limit(20);  
  
 // Display the first 10 numbers  
 nums.limit(10)  
 .forEach(System.*out*::println);  
 }  
}

TASK 19: 

package File\_Handeling;  
import java.util.stream.Stream;  
  
public class TASKFH019 {  
 public static void main(String[] args) {  
 // Create a stream of numbers from 1 to 20  
 Stream<Integer> nums = Stream.*iterate*(1, n -> n + 1)  
 .limit(20);  
  
 // Skip the first 15 numbers  
 Stream<Integer> skipNums = nums.skip(15);  
  
 // Print the remaining numbers  
 skipNums.forEach(System.*out*::println);  
 }  
}

TASK 20: 

package File\_Handeling;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Optional;  
  
public class TASKFH020 {  
 public static void main(String[] args) {  
  
 List<Integer> numbers = Arrays.*asList*(1, 2, 3, 4, 5);  
  
 Optional<Integer> sum = numbers.stream().reduce((x, y) -> x + y);  
 System.*out*.println("Sum of all elements: " + sum.orElse(0));  
  
  
 Optional<Integer> max = numbers.stream().reduce(Integer::*max*);  
 System.*out*.println("Maximum element: " + max.orElse(0));  
  
 List<String> strings = Arrays.*asList*("Hello", " ", "world", "!");  
 Optional<String> concatenatedString = strings.stream().reduce((x, y) -> x + y);  
 System.*out*.println("Concatenated string: " + concatenatedString.orElse(""));  
 }  
}