File handling

```
import
import java.io.*;
import java.util.*;
```

1.FileInputStream (Reading from a file): file to console

Syntax: FIleInputStream fin = new FileInputStream (”filename”);

- FileInputStream is meant for reading streams of raw bytes such as image data.

- But, For reading streams of characters, consider using FileReader.

```
import java.io.FileInputStream;

public class FileInputStreamDemo {
    public static void main(String[] args) {
        try {
            FileInputStream fin = new FileInputStream( name: "file1.txt");
            int i;
            while ((i = fin.read()) != -1) {
                System.out.print((char) i);
            }
            fin.close();
        } catch (Exception e) {
            System.out.println("error");
        }
    }
```

Based on the image shown, here are the steps for reading from a file using FileInputStream in Java:

1. Create a FileInputStream object with the filename as parameter:

`FileInputStream fin = new FileInputStream("filename.txt");`

2. Use a while loop with fin.read() to read characters until end of file (-1 is reached)
3. Cast the read integer to char to get the actual character
4. Print or process the character as needed
5. Close the FileInputStream using fin.close() when done

Remember to handle exceptions using try-catch blocks as file operations can throw IOException.

2.FileOutputStream(Writing to a file): console to file

Syntax: FIleOutputStream fin = new FileOutputStream (”filename”);

- FileOutputStream is an outputstream for writing data/streams of raw bytes to file or storing data to file.
- FileOutputStream is a subclass of OutputStream.
- To write primitive values into a file, we use FileOutputStream class.
- For writing byte-oriented and character-oriented data, we can use FileOutputStream
- But for writing character-oriented data, FileWriter is more preferred.

```java
public class FileOutputStreamDemo {
    public static void main(String[] args) {
        try {
            FileOutputStream fout = new FileOutputStream( name: "file1.txt");
            String s = "Hello World.";
            byte[] b = s.getBytes();
            fout.write(b);
            fout.close();
            System.out.println("success...");
        } catch (IOException e) {
            System.out.println("Error..");
        }
    }
}
```

Based on the attached image and context, here are the steps to write to a file using FileOutputStream in Java:

1. Create a FileOutputStream object with the filename as parameter:

   ```
   FileOutputStream fout = new FileOutputStream("file1.txt");
   ```

2. Convert your data to bytes using getBytes() if working with strings
3. Use write() method to write the bytes to the file
4. Close the FileOutputStream using close() method
5. Handle any IOExceptions using try-catch blocks.

Character stream

1.FileWriter(Writing to a file): console to file

Syntax:FileWriter writer = new FileWriter(String fileName);

- CharacterStream classes are mainly used to read characters from the source and write them to the destination.

- CharacterStream classes are used to work with 16-bit Unicode characters. They can perform operations on characters, char arrays and Strings.
- CharacterStream classes are divided into two types of classes:
- Reader class
- Writer class

```java
import java.io.*;

public class FileWriterDemo {
    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter( fileName: "file2.txt");
            fw.write( str: "Hello World !!!.");
            fw.close();
            System.out.println("Success...");
        } catch (IOException e) {
            System.out.println("error");
        }
    }
}
```

The String doesn't have to be changed to byte , we can directly send the string ,

2.FileReader (Reading from a file): file to console

Syntax:FileReader reader = new FileReader(File name);

```java
import java.io.*;

public class FileReaderDemo {
    public static void main(String[] args){
        try {
            FileReader fr = new FileReader( fileName: "file2.txt");
            int i;
            while ((i = fr.read()) != -1)
                System.out.print((char) i);
            fr.close();
        } catch (IOException e) {
            System.out.println("error");
        }
    }
}
```

same as FIleInputStream

3.BufferedWriter :

```java
import java.io.*;
public class BufferedWriter{
 public static void main (String [] args ){
  try{
   FileWriter fw= new FileWriter (file.txt);
   BufferedWriter writer = new BufferedWriter (fw);
   String str ="Hello World";
   writer.write(str);
   writer.close();
   fw.close();
  }catch(Exception e){
   System.out.println("Error");
   }
  }
```

```
}
```

4.BufferedOutputStream:

```
import java.io.*;

public class BufferedOutputStreamExample {
    public static void main(String[] args) {
        String data = "This is a sample text written using BufferedOutputStream.";
        String filePath = "output.txt";

        try (
            // Create a FileOutputStream and wrap it with a BufferedOutputStream
            FileOutputStream fos = new FileOutputStream(filePath);
            BufferedOutputStream bos = new BufferedOutputStream(fos)
        ) {
            // Convert the string to bytes and write it to the file
            bos.write(data.getBytes());
            System.out.println("Data has been written to the file: " + filePath);
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

5.BufferedReader

```
package Filehandling;
import java.io.*;
public class BufferedReaderExample {
    public static void main(String[] args) {
        try (
                // Create a FileReader and wrap it with a BufferedReader
                FileReader fr = new FileReader("file.txt");
                BufferedReader br = new BufferedReader(fr)
```

```
        ) {
            String line;
            // Read and print each line from the file
            while ((line= br.readLine()) != null) {
                System.out.println(line);
            }
        } catch (FileNotFoundException e) {
            System.out.println("The file does not exist.");
        } catch (IOException e) {
            System.out.println("An error occurred while reading the file: " + e.getMes
        }
    }
}
```

6.BufferedInputStream:

```
import java.io.*;

public class BufferedInputStreamExample {
    public static void main(String[] args) {
        try {
            // Open a file input stream
            FileInputStream fileInput = new FileInputStream("example.txt");

            // Wrap the FileInputStream with a BufferedInputStream
            BufferedInputStream bufferedInput = new BufferedInputStream(fileInput);

            // Read data from the file
            System.out.println("Reading file content:");
            int data;
            while ((data = bufferedInput.read()) != -1) {
                System.out.print((char) data); // Convert byte to character and print
            }
```

```
            // Close the stream
            bufferedInput.close();
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

BufferedReader and **BufferedWriter** in Java are used when dealing with text data, especially when reading or writing large amounts of data efficiently.

Q.Copy content of one file to another

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) {

        try (
            FileInputStream inputStream = new FileInputStream("source.txt"); // To re
            FileOutputStream outputStream = new FileOutputStream("destination.txt"
        ) {
            int i;

            // Read each byte from the source file and write it to the destination file
            while ((i = inputStream.read()) != -1) {
                outputStream.write(i);
            }
            System.out.println("File copied successfully !");
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

Q.WAP to read and write 5 records to a file

1.Create a class implementing Serializable.

- Inside the class write the required property and functions/methods.

2.Create a public class with main methods and try and catch .

For writing to a file

- Make the object of FileOutputStream passing file path.
- Make a object of ObjectOutputStream passing above fileobject as parameter .
- Enter a loop
- Inside the loop Make the object of the Class ,Use this object to access the function.
- Write with ObjectOutputStream obj ex -out.write(obj) and pass class object as parameter.

For Reading from a file

- Make the object of FileInputStream passing file path.
- Make a object of ObjectInputStream passing above fileobject as parameter .
- Enter a loop
- Inside the loop Make the object of the Class , and type cast the obj to Class after reading using ObjectInputStream obj ex -in.read() Use the class object to access the function.

```
package File;


import java.io.*;
import java.util.*;
class Students implements Serializable{
   int id;
```

```
    String name;
    void input(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your id : ");
        int id = sc.nextInt();
        this.id = id;
        sc.nextLine();
        System.out.println("Enter your name : ");
        String name = sc.nextLine();
        this.name=name;
    }
    void display(){
        System.out.println("Id : "+id+" "+"Name : "+name);
    }
}
public class ReadWriteDemo{
    public static void main(String[] args) {
        try(
                FileOutputStream fout = new FileOutputStream("file1.txt");
                ObjectOutputStream out = new ObjectOutputStream(fout)){
            for(int i=0; i<5;i++){
                Students obj = new Students();
                obj.input();
                out.writeObject(obj);
            }
        }catch(Exception e){
            System.out.println("Error");
        }

        try(
                FileInputStream fin = new FileInputStream("file1.txt");
                ObjectInputStream in = new ObjectInputStream(fin)){
            for(int i=0;i<5;i++){
                Students obj1 = (Students) in.readObject();
                obj1.display();
            }
```

```
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

```
Id : 1 Name : ram
Id : 2 Name : shyam
Id : 3 Name : hari
Id : 4 Name : gita
Id : 5 Name : sita
```

Reading and writing object to file

```
import java.io.*;

class ObjectOutputStreamDemo {
    public static void main(String[] args) {
        Dog dog1 = new Dog( name: "Tyson",  breed: "Labrador");
        try {
            FileOutputStream fileOut = new FileOutputStream( name: "file.txt");
            ObjectOutputStream objOut = new ObjectOutputStream(fileOut);
            objOut.writeObject(dog1);
            objOut.close();
        } catch (Exception e) {
            System.out.println("Error...");
        }
    }
}
```

```java
import java.io.*;

public class ObjectInputStreamDemo {
    public static void main(String[] args) {
        try {
            FileInputStream fin = new FileInputStream( name: "file1.txt");
            ObjectInputStream oin = new ObjectInputStream(fin);
            Dog dog = (Dog) oin.readObject();
            System.out.println(dog.name + "   " + dog.breed);
        } catch (ClassNotFoundException | IOException e) {
            System.out.println("Error");
        }
    }
}
```

Q.Write a program to store the data into the database

```java
package File;

import java.io.*;
import java.sql.*;
import java.util.*;

class Students implements Serializable {
    int id;
    String name;

    void input() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your id : ");
        int id = sc.nextInt();
        this.id = id;
        sc.nextLine();
        System.out.println("Enter your name : ");
        String name = sc.nextLine();
        this.name = name;
```

```
    }

    void display() {
        System.out.println("Id : " + id + " " + "Name : " + name);
    }
}

public class ReadWriteDemo {
    public static void main(String[] args) {
        try (
                FileOutputStream fout = new FileOutputStream("file1.txt");
                ObjectOutputStream out = new ObjectOutputStream(fout)) {
            for (int i = 0; i < 5; i++) {
                Students obj = new Students();
                obj.input();
                out.writeObject(obj);
            }
        } catch (Exception e) {
            System.out.println("Error");
        }

        try (
                FileInputStream fin = new FileInputStream("file1.txt");
                ObjectInputStream in = new ObjectInputStream(fin)) {
            for (int i = 0; i < 5; i++) {
                Students obj1 = (Students) in.readObject();
                obj1.display();
            }
            storeToDatabase(); // Call the function to store data into the database
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    static void storeToDatabase() {
        try (
```

```
            FileInputStream fin = new FileInputStream("file1.txt");
            ObjectInputStream in = new ObjectInputStream(fin);
             String url = "jdbc:mysql://localhost:3306/students";
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, "root", "Bri$ti1010");
        ) {
            String query = "INSERT INTO student (id, name) VALUES (?, ?)";
            PreparedStatement preparedStatement = con.prepareStatement(query);

            for (int i = 0; i < 5; i++) {
                Students obj = (Students) in.readObject();
                preparedStatement.setInt(1, obj.id);
                preparedStatement.setString(2, obj.name);
                preparedStatement.executeUpdate();
            }
            System.out.println("Data stored in the database successfully!");
        } catch (Exception e) {
            System.out.println("Error while storing to database: " + e.getMessage());
        }
    }
```

```
package File;

import java.sql.*;
import java.util.*;

class Students {
    int id;
    String name;

    void input() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your id : ");
```

```
        int id = sc.nextInt();
        this.id = id;
        sc.nextLine();
        System.out.println("Enter your name : ");
        String name = sc.nextLine();
        this.name = name;
    }
}

public class ReadWriteDemo {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            Students obj = new Students();
            obj.input();
            saveToDb(obj.id, obj.name);
        }
    }

    static void saveToDb(int id, String name) {
        try {
            String url = "jdbc:mysql://localhost:3306/students";
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(url, "root", "Bri$ti1010");
            String query = "INSERT INTO student (id, name) VALUES (?, ?)";
            PreparedStatement preparedStatement = con.prepareStatement(query);
            preparedStatement.setInt(1, id);
            preparedStatement.setString(2, name);

            preparedStatement.executeUpdate();
            System.out.println("Data inserted successfully: ID = " + id + ", Name = " +
        } catch (Exception e) {
            System.out.println("Error while inserting data: " + e.getMessage());
        }
    }
}
```

Yes, you can directly use fout.write(s.getBytes()) instead of creating a separate byte array variable. This will achieve the same result more concisely. Looking at the image, you can modify the code to:

```
try {
    FileOutputStream fout = new FileOutputStream("file1.txt");
    String s = "Hello World.";
    fout.write(s.getBytes());
    fout.close();
    System.out.println("success...");
} catch (IOException e) {
    System.out.println("Error..");
}
```

```
import java.io.*;
import java.util.Scanner;

public class PrimeCompositeFileWriter {

    // Method to check if a number is prime using a divisor counter
    public static boolean isPrime(int number) {
        if (number <= 1) return false;

        int divisorCount = 0;
        for (int i = 1; i <= number; i++) {
            if (number % i == 0) {
                divisorCount++;
            }
        }

        return divisorCount == 2;  // A prime number has exactly two divisors: 1 and
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
        int[] numbers = new int[10];

        // Reading 10 numbers from the user
        System.out.println("Enter 10 numbers:");
        for (int i = 0; i < 10; i++) {
            numbers[i] = scanner.nextInt();
        }

        // Writing prime and composite numbers to respective files using FileOutput
        try (FileOutputStream primeStream = new FileOutputStream("prime.txt");
             FileOutputStream compositeStream = new FileOutputStream("composite
             BufferedWriter primeWriter = new BufferedWriter(new OutputStreamWrite
             BufferedWriter compositeWriter = new BufferedWriter(new OutputStream

            for (int number : numbers) {
                if (isPrime(number)) {
                    primeWriter.write(number + "\n");  // Write prime numbers to prime.tx
                } else {
                    compositeWriter.write(number + "\n");  // Write composite numbers t
                }
            }

            System.out.println("Numbers have been successfully written to prime.txt

        } catch (IOException e) {
            System.out.println("An error occurred while writing to the files.");
            e.printStackTrace();
        }
    }
}
```

Difference Between Byte Stream and Character Stream:

Byte Stream	Character Stream
Works with binary data (8-bit bytes).	Works with character data (16-bit Unicode characters).
Used for handling raw data, such as reading/writing images, files, and audio.	Used for reading/writing text files.
Classes: `InputStream` and `OutputStream`.	Classes: `Reader` and `Writer`.
No need for encoding/decoding as it deals with bytes.	Requires encoding/decoding (e.g., UTF-8) to handle characters.