A. Codice del client e del server per trasferire un file attraverso un socket java:

1: Codice del client (da inserire come classe in un progetto SocketFileClient)

```
import java.net.*;
import java.io.*;

public class SocketFileClient {

       public static void main(String[] args) throws IOException {
         Socket socket = null;
         String host = "127.0.0.1";

         socket = new Socket(host, 4444);

         File file = new File("C:\\Users\\Administrator\\Desktop\\prova.txt");

         byte[] bytes = new byte[16 * 1024];
         InputStream in = new FileInputStream(file);
         OutputStream out = socket.getOutputStream();

         int count;
         while ((count = in.read(bytes)) > 0) {
             out.write(bytes, 0, count);
         }

         out.close();
         in.close();
         socket.close();
    }
}
```

2: Codice del server (da inserire come classe in un progetto SocketFileServer)

```
import java.io.*;
import java.net.*;


public class SocketFileServer {
public static void main(String[] args) throws IOException {
    ServerSocket serverSocket = null;

    try {
        serverSocket = new ServerSocket(4444);
    } catch (IOException ex) {
        System.out.println("Can't setup server on this port number. ");
    }

    Socket socket = null;
    InputStream in = null;
    OutputStream out = null;

    try {
        socket = serverSocket.accept();
    } catch (IOException ex) {
        System.out.println("Can't accept client connection. ");
    }
```

```
    try {
        in = socket.getInputStream();
    } catch (IOException ex) {
        System.out.println("Can't get socket input stream. ");
    }

    try {
        out = new FileOutputStream("C:\\Users\\Administrator\\Desktop\\prova1.txt");
    } catch (FileNotFoundException ex) {
        System.out.println("File not found. ");
    }

    byte[] bytes = new byte[16*1024];

    int count;
    while ((count = in.read(bytes)) > 0) {
        out.write(bytes, 0, count);
    }

    out.close();
    in.close();
    socket.close();
    serverSocket.close();
}
```

B. Codice del client e del server per trasferire un array di oggetti attraverso un socket java:

1: Codice del client (da inserire come classe in un progetto `SocketObjectClient`)

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

class Message implements Serializable{
    private final String text;

    public Message(String text) {
        this.text = text;
    }

    public String getText() {
        return text;
    }
}

public class SocketFileClient {
      public static void main(String[] args) throws IOException {
        // need host and port, we want to connect to the ServerSocket at port 7777
        Socket socket = new Socket("localhost", 7777);
        System.out.println("Connected!");

        // get the output stream from the socket.
        OutputStream outputStream = socket.getOutputStream();
        // create an object output stream from the output stream so we can send an
object through it
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);

        // make a bunch of messages to send.
        List<Message> messages = new ArrayList<>();
        messages.add(new Message("Hello from the other side!"));
        messages.add(new Message("How are you doing?"));
        messages.add(new Message("What time is it?"));
        messages.add(new Message("Hi hi hi hi."));

        System.out.println("Sending messages to the ServerSocket");
        objectOutputStream.writeObject(messages);

        System.out.println("Closing socket and terminating program.");
        socket.close();
    }
```

2: Codice del server (da inserire come classe in un progetto `SocketObjectServer`)

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

class Message implements Serializable{
    private final String text;
```

```
    public Message(String text) {
        this.text = text;
    }

    public String getText() {
        return text;
    }
}



public class SocketFileServer {
      public static void main(String[] args) throws IOException, ClassNotFoundException
{
        // don't need to specify a hostname, it will be the current machine
        ServerSocket ss = new ServerSocket(7777);
        System.out.println("ServerSocket awaiting connections...");
        Socket socket = ss.accept(); // blocking call, this will wait until a
connection is attempted on this port.
        System.out.println("Connection from " + socket + "!");

        // get the input stream from the connected socket
        InputStream inputStream = socket.getInputStream();
        // create a DataInputStream so we can read data from it.
        ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);

        // read the list of messages from the socket
        List<Message> listOfMessages = (List<Message>) objectInputStream.readObject();
        System.out.println("Received [" + listOfMessages.size() + "] messages from: " +
socket);
        // print out the text of every message
        System.out.println("All messages:");
        listOfMessages.forEach((msg)-> System.out.println(msg.getText()));

        System.out.println("Closing sockets.");
        ss.close();
        socket.close();
    }
}
```