```
// Method to serialize (save) the list of users to a file
public void serialize(List<User> users, String filename) throws IOException;
public List<User> deserialize(String filename) throws IOException, ClassNotFoundException;

boolean userRemoved = users.removeIf(user -> user.getEmail().equals(email));

if (users.isEmpty()) {
        System.out.println("No users available.");
} else {
       for (User user : users) {
                System.out.println(user);
             }
}

Scanner scanner = new Scanner(System.in);
String option = scanner.nextLine();

 // Class to handle CSV serialization and deserialization
public class CSVSerializer implements UserSerializerInterface{

   @Override
   public void serialize(List<User> users, String filename) throws IOException {
      File file = new File(filename);
      if (!file.exists()) {
         file.createNewFile();
      }

      try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
         for (User user : users) {
            writer.write(String.format("%s;%s;%d;%s", user.getFirstName(), user.getLastName(), user.getAge(), user.getEmail()));
            writer.newLine();
         }
      }
   }

   @Override
   public List<User> deserialize(String filename) throws IOException, ClassNotFoundException {
      List<User> users = new ArrayList<>();
      File file = new File(filename);
      if (file.canRead()) {
         try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = reader.readLine()) != null) {
               String[] fields = line.split(";");
               if (fields.length == 4) {
                  String firstName = fields[0];
                  String lastName = fields[1];
                  int age = Integer.parseInt(fields[2]);
                  String email = fields[3];
                  users.add(new User(firstName, lastName, age, email));
               }
            }
         }
      }
      return users;
   }
```

```
}

// Method to sort users
users.sort(Comparator.comparing(User::getCreationDate));
System.out.println("Users sorted by creation date:");
users.sort(Comparator.comparing(User::getCreationDate).reversed());
users.sort(Comparator.comparingInt(User::getPriority));
System.out.println("Users sorted by priority:");

String[] indexArray = indices.split(",");
   try {
      for (String indexStr : indexArray) {
         int index = Integer.parseInt(indexStr.trim());
         if (index >= 0 && index < users.size()) {
            users.get(index).setFirstName(newFirstName);
            System.out.println("Updated user at index " + index + " to have first name: " + newFirstName);
         } else {
            System.out.println("Index " + index + " is out of bounds.");
         }
      }
   }

// Print the absolute path to the file where users were saved
System.out.println("Users saved to file: " + file.getAbsolutePath());
```